Digital Communications and Networks 4 (2018) 161–175

Contents lists available at ScienceDirect

Digital Communications and Networks

journal homepage: www.keaipublishing.com/en/journals/digital-communications-and-networks/

Machine learning for internet of things data analysis: a survey

Mohammad Saeid Mahdavinejad[a,b,*], Mohammadreza Rezvan[a,b], Mohammadamin Barekatain[c], Peyman Adibi[a], Payam Barnaghi[d], Amit P. Sheth[b]

[a] University of Isfahan, Iran
[b] Kno.e.sis - Wright State University, USA
[c] Technische Universität München, Germany
[d] University of Surrey, UK

A R T I C L E I N F O

Keywords:
Machine learning
Internet of Things
Smart data
Smart City

A B S T R A C T

Rapid developments in hardware, software, and communication technologies have facilitated the emergence of Internet-connected sensory devices that provide observations and data measurements from the physical world. By 2020, it is estimated that the total number of Internet-connected devices being used will be between 25 and 50 billion. As these numbers grow and technologies become more mature, the volume of data being published will increase. The technology of Internet-connected devices, referred to as Internet of Things (IoT), continues to extend the current Internet by providing connectivity and interactions between the physical and cyber worlds. In addition to an increased volume, the IoT generates big data characterized by its velocity in terms of time and location dependency, with a variety of multiple modalities and varying data quality. Intelligent processing and analysis of this big data are the key to developing smart IoT applications. This article assesses the various machine learning methods that deal with the challenges presented by IoT data by considering smart cities as the main use case. The key contribution of this study is the presentation of a taxonomy of machine learning algorithms explaining how different techniques are applied to the data in order to extract higher level information. The potential and challenges of machine learning for IoT data analytics will also be discussed. A use case of applying a Support Vector Machine (SVM) to Aarhus smart city traffic data is presented for a more detailed exploration.

1. Introduction

Emerging technologies in recent years and major enhancements to Internet protocols and computing systems, have made communication between different devices easier than ever before. According to various forecasts, around 25–50 billion devices are expected to be connected to the Internet by 2020. This has given rise to the newly developed concept of Internet of Things (IoT). IoT is a combination of embedded technologies including wired and wireless communications, sensor and actuator devices, and the physical objects connected to the Internet [1,2]. One of the long-standing objectives of computing is to simplify and enrich human activities and experiences (e.g., see the visions associated with “The Computer for the 21st Century” [3] or “Computing for Human Experience” [4]). IoT requires data to either represent better services to users or enhance the IoT framework performance to accomplish this intelligently. In this manner, systems should be able to access raw data from different resources over the network and analyze this information in order to extract knowledge.

Since IoT will be among the most significant sources of new data, data science will provide a considerable contribution to making IoT applications more intelligent. Data science is the combination of different scientific fields that uses data mining, machine learning, and other techniques to find patterns and new insights from data. These techniques include a broad range of algorithms applicable in different domains. The process of applying data analytics methods to particular areas involves defining data types such as volume, variety, and velocity; data models such as neural networks, classification, and clustering methods, and applying efficient algorithms that match with the data characteristics. By following our reviews, the following is deduced: First, because data is generated from different sources with specific data types, it is important to adopt or develop algorithms that can handle the data characteristics. Second, the great number of resources that generate data in real-time are not without the problem of scale and velocity. Finally, finding the best data model that fits the data is one of the most important issues for pattern recognition and for better analysis of IoT data. These issues have opened a vast number of opportunities in expanding new developments.

* Corresponding author.
E-mail addresses: saeid@knoesis.org (M.S. Mahdavinejad), p.barnaghi@surrey.ac.uk (P. Barnaghi).

https://doi.org/10.1016/j.dcan.2017.10.002
Received 26 July 2017; Received in revised form 4 October 2017; Accepted 9 October 2017
Available online 12 October 2017

Big data is defined as high-volume, high-velocity, and high variety data that demands cost-effective, innovative forms of information processing that enable enhanced insight, decision making, and process automation [5].

With respect to the challenges posed by big data, it is necessary to introduce a new concept termed smart data, which means: ”realizing productivity, efficiency, and effectiveness gains by using semantics to transform raw data into Smart Data” [6]. A more recent definition of this concept is: ”Smart Data provides value from harnessing the challenges posed by volume, velocity, variety, and veracity of Big Data, and in turn providing actionable information and improving decision making.” [7]. Finally, smart data can act as a good representative for IoT data.

1.1. The Contributions of this paper

The objective here is to answer the following questions:

A) **How can machine learning algorithms be applied to IoT smart data?**
B) **What is the taxonomy of machine learning algorithms that can be adopted in IoT?**
C) **What are the characteristics of IoT data in the real world?**
D) **Why is the smart city a typical use case of IoT applications?**

A) To understand which algorithm is more appropriate for processing and decision-making on smart data generated from the things in IoT, it is essential to consider the following three concepts. First, the IoT application (Section 3). Second, the IoT data characteristics (Section 4.2), and third, the data-driven vision of machine learning algorithms (Section 5). We finally discuss the issues in Section 6.

B) Around 70 articles in the field of IoT data analysis are reviewed, revealing that there exist eight major groups of algorithms applicable to IoT data. These algorithms are categorized according to their structural similarities, types of data they can handle, and the amount of data they can process in a reasonable time.

C) Having reviewed the real-world perspective of how IoT data is analyzed by over 20 authors, many significant and insightful results have been revealed regarding data characteristics. We discuss the results in Section 6 and Table 1. To gain a deeper insight into IoT smart data, patterns must be extracted and the generated data interpreted. Cognitive algorithms undertake interpretation and matching, much as the human mind would do. Cognitive IoT systems previously learn from generated data and improve when performing repeated tasks. Cognitive computing acts as a prosthetic for human cognition by analyzing massive amount of data and responding to questions that humans might have when making certain decisions. Cognitive IoT plays an important role in enabling the extraction of meaningful patterns from generated IoT smart data [8].

Table 1
Characteristic of smart data in smart cities.

Smart city use cases	Type of data	Where data processed	References
Smart Traffic	Stream/Massive Data	Edge	[14,43]
Smart Health	Stream/Massive Data	Edge/Cloud	[44]
Smart Environment	Stream/Massive Data	Cloud	[45]
Smart Weather Prediction	Stream Data	Edge	[46]
Smart Citizen	Stream Data	Cloud	[47,48]
Smart Agriculture	Stream Data	Edge/Cloud	[49]
Smart Home	Massive/Historical Data	Cloud	[50]
Smart Air Controlling	Massive/Historical Data	Cloud	[38]
Smart Public Place Monitoring	Historical Data	Cloud	[51]
Smart Human Activity Control	Stream/Historical Data	Edge/Cloud	[52,53]

D) A smart city has been selected as our primary use case in IoT for three reasons: First, among all of the reviewed articles the focus of 60 percent is on the field of the smart cities. Second, smart cities include many of the other use cases in IoT. Third, there are many open datasets for smart city applications that are easily accessible for researchers. Furthermore, a Support Vector Machine (SVM) algorithm is implemented on the Aarhus City smart traffic data to predict traffic hours during one day. By answering the above questions about IoT smart data and machine learning algorithms, we would be able to choose the best machine learning algorithm that can handle IoT smart data characteristics. Unlike similar surveys regarding machine learning and IoT, readers of this article would be able to obtain a deep and technical understanding of machine learning algorithms, IoT applications, and IoT data characteristics along with both technical and simple implementations.

1.2. Organization

The rest of this paper is organized as follows. Related articles in this field are reviewed and reported in Section 2. IoT applications and communication protocols, computing frameworks, IoT architecture, and smart city segments are reviewed, explained, summarized, and illustrated in Section 3. The quality of data, big data generation, sensor data integration, and semantic data annotation are reviewed in Section 4. Machine learning algorithms in eight categories based on recent studies on IoT data and frequency of machine learning algorithms are reviewed and summarized in Section 5. The matching of the algorithms to particular smart city applications is carried out in Section 6, and the conclusion together with future research trends and open issues are presented in Section 7, Fig. 1 shows the structure of the survey.

2. Literature review

Since IoT represents a new concept for the Internet and smart data, it is a challenging area in the field of computer science. The important challenges for researchers with respect to IoT consist of preparing and processing data.

Ref. [9] proposed four data mining models for processing IoT data. The first is a *multi layer model*, based on a data collection layer, a data management layer, an event processing model, and a data mining service layer. The second model is a *distributed data mining model*, proposed for data deposition at different sites. The third is a *grid-based data mining model* where the authors seek to implement heterogeneous, large scale, and high performance applications. The final model is a *data mining model from a multi technology integration perspective*, where a corresponding framework for the future Internet is described.

Ref. [10] performed research into warehousing Radio Frequency Identification (RFID) data, with a focus specifically on managing and mining RFID stream data.

Ref. [11] introduced a systematic method for reviewing data mining knowledge and techniques in most common applications. In this study, they reviewed some data mining functions like classification, clustering, association analysis, time series analysis, and outline detection. They revealed that the data generated by data mining applications, such as e-commerce, industry, healthcare, and city governance are similar to that of the IoT data. Following their findings, they assigned the most popular data mining functionality to the application and determined which data mining functionality was the most appropriate for processing each specific application's data.

Ref. [12] ran a survey to respond to some of the challenges in preparing and processing data on the IoT through data mining techniques. The authors divided their research into three major sections. In the first and second sections, they explain IoT, the data, and the challenges that exist in this area, such as building a model for mining, and mining algorithms for IoT. In the third section, they discuss the potential and open

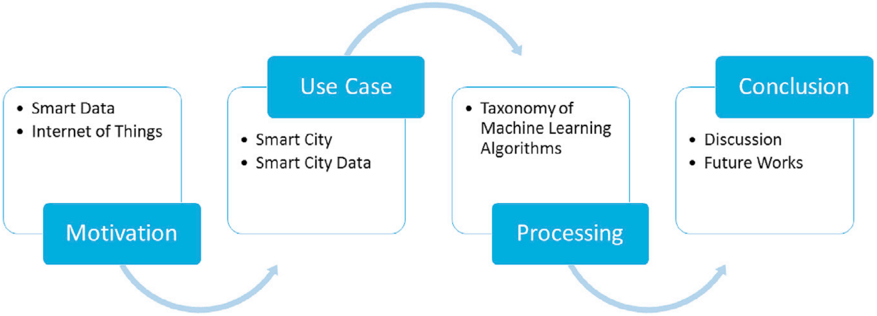

Fig. 1. Organization of survey.

issues that exist in this field. Here, it is summarized that data mining on IoT data involves three major concerns: First, it must be shown that processing the data will solve the chosen problems. Next, the data characteristics must be extracted from the generated data, and then, the appropriate algorithm is chosen according to the taxonomy of algorithms and data characteristics.

Ref. [13] attempted to explain the smart city infrastructure in IoT and discussed advanced communication to support added-value services for the administration of the city and citizens thereof. The authors provide a comprehensive review of enabling technologies, protocols, and architectures for smart city. In the technical part of the article, the authors reviewed the data of Padova Smart City.

3. Internet of Things

The purpose of *Internet of Things* (IoT) is to develop a smarter environment and a simplified life-style by saving time, energy, and money. Through this technology, expenses in different industries can be reduced. The enormous investments and many studies running on IoT have made IoT a growing trend in recent years. IoT consists of a set of connected devices that can transfer data among one another in order to optimize their performance; these actions occur automatically and without human awareness or input. IoT includes four main components: 1) sensors, 2) processing networks, 3) data analysis data, and 4) system monitoring. The most recent advances made in IoT began when RFID tags were put into use more frequently, lower cost sensors became more available, web technology developed, and communication protocols changed [14,15]. The IoT is integrated with various technologies, and connectivity is a necessary and sufficient condition for it to function. Therefore, communication protocols are constituents of this technology that should be enhanced [16,17]. In IoT, communication protocols can be divided into three major components:

(1) *Device to Device (D2D)*: this type of communication enables communication between nearby mobile phones. This represents the next generation of cellular networks.
(2) *Device to Server (D2S)*: in this type of communication device, all the data is sent to the servers, which can be close or far from the devices. This type of communication is mostly applied to cloud processing.
(3) *Server to Server (S2S)*: in this type of communication, servers transmit data between each other. This type of communication is mostly applied for cellular networks.

Processing and preparing data for these communications is a critical challenge. To respond to this challenge, different kinds of data processing, such as analytics at the edge, stream analysis, and IoT analysis at the database, must be applied. The decision to apply any of the mentioned processes depends on the particular application and its needs [18]. Fog and cloud processing are two analytical methods adopted for processing and preparing data before transferring it to other things. The whole task of IoT is summarized as follows: First, sensors and IoT devices collect information from the environment. Next, knowledge is extracted from the raw data. Then, data is ready for transferring to other objects, devices, or servers through the Internet.

3.1. Computing framework

Another important part of IoT is the *computing framework* of processing data, the most famous of which are fog and cloud computing. IoT applications use both frameworks depending on the application and process location. In some applications, data should be processed upon generation, while in other applications it is not necessary to process data immediately. The instant processing of data and the network architecture that supports it is known as fog computing. Collectively, these are applied for edge computing [19].

3.1.1. Fog computing

Here, the architecture of fog computing is applied to migrating information from a data center task to the edge of the servers. This architecture is built based on the edge servers. Fog computing provides limited computing, storage, and network services, also providing logical intelligence and filtering of data for data centers. This architecture has been and is being implemented in vital areas like eHealth and military applications [20,21].

3.1.2. Edge computing

In this architecture, processing is run at a distance from the core, toward the edge of the network. This type of processing enables data to be initially processed at edge devices. Devices at the edge may not be connected to the network in a continuous manner, and so they require a copy of the master data/reference data for offline processing. Edge devices have different features such as 1) enhancing security, 2) filtering and cleaning data, and 3) storing local data for local use [22].

3.1.3. Cloud computing

Here, data for processing is sent to data centers, and after being analyzed and processed, they become accessible.

This architecture has high latency and high load balancing, indicating that this architecture is not sufficient for processing IoT data because most processing should run at high speeds. The volume of this data is high, and big data processing will increase the CPU usage of the cloud servers [23]. There are various types of cloud computing:

(1) *Infrastructure as a Service (IaaS)*: where the company purchases all the equipment like hardware, servers, and networks.
(2) *Platform as a Service (PaaS)*: where all the equipment above is placed for rent on the Internet.
(3) *Software as a Service(SaaS)*: where a distributed software model is presented. In this model, all practical software will be hosted by a service provider and is accessible to the users through the Internet [24].

(4) *Mobile Backend as a Service (MBaaS)*: also known as Backend as a Service (BaaS), provides a web or mobile application with a path in order to connect the application to the backend cloud storage. MBaaS provides features like user management, push notifications, and integration with social network services. This cloud service benefits from an Application Programming Interface (API) and software Development Kits (SDK).

3.1.4. Distributed computing

This architecture is designed for processing high volumes of data. In IoT applications, because the sensors generate data in a repetitive manner, big data challenges are encountered [22,25]. To overcome this phenomenon, distributed computing is designed to divide data into packets, and assign the packets to different computers for processing. This distributed computing has different frameworks like Hadoop and Spark. When migrating from cloud to fog and distributed computing, the following phenomena occur: 1) a decrease in network loading, 2) an increase in data processing speed, 3) a reduction in CPU usage, 4) a reduction in energy consumption, and 5) an ability to process a higher volume of data.

Because the smart city is one of the primary applications of IoT, the most important use cases of the smart city and their data characteristics are discussed in the following sections.

4. Smart city

Cities always demand services to enhance the quality of life and make existing services more efficient. In the last few years, the concept of smart cities has played an important role in academia and industry [26]. With an increase in the population and complexity of city infrastructures, cities seek methods to handle large-scale urbanization problems. IoT plays a vital role in collecting data from the city environment. IoT enables cities to use live status reports and smart monitoring systems to react more intelligently to emerging situations such as earthquakes and volcanoes. By adopting IoT technologies in city, the majority of the city's assets can be connected to one another, making them more readily observable, and consequently, more easy to monitor and manage. The purpose of building smart cities is to improve services like traffic management, water management, and energy consumption, as well as improving the quality of life for the citizens. The objective of smart cities is to transform rural and urban areas into places of democratic innovation [27]. Such smart cities seek to decrease the expenses in public health, safety, transportation, and resource management, thus assisting their economies [28].

In Ref. [29], the authors believe that in the long term, the vision for a smart city would be that all the cities' systems and structures will monitor their own conditions and carry out self-repair upon need.

4.1. Use case

A city has an important effect on society because the city touches all aspects of human life. A smart city can assist in having a comfortable life. The use cases for smart cities consist of smart energy, smart mobility, smart citizens, and urban planning. This division is based on a review of the latest studies in this field and the most recent reports released by McKinsey and Company.

4.1.1. Smart energy

Smart energy is one of the most important research areas of IoT because it is essential to reduce overall power consumption [30]. It offers high-quality and affordable environmentally friendly energy. Smart energy includes a variety of operational and energy measures, including Smart energy applications, smart leak monitoring, and renewable energy resources, etc. Using smart energy (i.e., deployment of a smart grid) implies a fundamental re-engineering of the electricity services [31]. The smart grid is one of the most important applications of smart energy. It includes many high-speed time series data to monitor key devices. For managing this kind of data, [32] introduced a method to manage and analyze time series data in order to make them organized on demand. Moreover, the smart energy infrastructure will become more complex in the future, and therefore [33] proposed a simulation system to test new concepts and optimization approaches and forecast future consumption. Another important application of smart energy is leak monitoring systems. The objective of such a system is to model a water or gas management system that would optimize energy resource consumption [34,35].

4.1.2. Smart mobility

Mobility is another important aspect of any city. Through IoT, city officials can improve the quality of life in the city. Smart mobility can be divided into the following three major components:

(1) *Autonomous cars*: IoT will have a broad range of effects on how vehicles are run. The most important question is about how IoT can improve vehicle services. IoT sensors and wireless connections make it possible to create self-driving cars and monitor vehicle performance. With the data collected from vehicles, the most popular/congested routes can be predicted, and decisions can be made to decrease traffic congestion. Self-driving cars can improve passenger safety because they have the ability to monitor the driving of the other cars.

(2) *Traffic control*: Optimizing the traffic flow by analyzing sensor data is another part of mobility in the city. For traffic control, traffic data will be collected from cars, road cameras, and counter sensors installed on roads.

(3) *Public transportation*: IoT can improve the public transportation system management by providing accurate location and routing information to a smart transportation system. It can assist passengers in making better decisions in their schedules as well as decrease the amount of wasted time. There exist different perspectives on how to build smart public transportation systems. These systems need to manage different kinds of data like vehicle location data and traffic data. Smart public transportation systems should be real-time oriented in order to make proper decisions in real-time as well as use historical data analysis [36]. For instance, [37] proposed a mechanism that considers smart city devices as graph nodes, and the authors used big data solutions to solve these issues.

4.1.3. Smart citizens

This use case for smart cities covers a broad range of areas in human lives, such as environmental monitoring, crime monitoring, and social health. The environment with all its components is fundamental and vital for life. Consequently, making progress in technology is guaranteed to enhance security. Close monitoring devoted to crime would also contribute to overall social health.

4.1.4. Urban planning

Another important aspect in use cases for the smart city is reaching long-term decisions. Because the city and environment both play major roles in human life, reaching decisions in this context is critical. By collecting data from different sources, it is possible to make a decision for the future of the city. Making decisions affecting the city infrastructure, design, and functionality is called urban planning. IoT is beneficial in this area because through smart city data analysis, the authorities can predict which part of the city will be more crowded in the future and find solutions for the potential problems. A combination of IoT and urban planning would have a major effect on scheduling future infrastructure improvements.

4.2. Smart city data characteristics

Smart city devices generate data in a continuous manner, indicating that the data gathered from traffic, health, and energy management applications would generate a sizable volume. In addition, because the data generation rate varies for different devices, processing data with different generation rates is a challenge. For example, the frequency of GPS sensor updates is measured in seconds, while the frequency of updates for temperature sensors may be measured hourly. Whether the data generation rate is high or low, there always exists a danger of losing important information. To integrate the sensory data collected from heterogeneous sources is challenging [14,38]. Ref. [39] applied big data analytic methods to distinguish the correlation between the temperature and traffic data in Santander, Spain.

Ref. [40] proposed a new framework integrating big data analysis and Industrial Internet of Things (IIoT) technologies for Offshore Support Vessels (OSV) based on a hybrid CPU/GPU high-performance computing platform.

Another characteristic is the dynamic nature of the data. Data for autonomous cars is an example of dynamic data because the sensor results will change based on different locations and times.

The quality of the collected data is important, particularly for smart city data, which have different qualities due to the fact that they are generated from heterogeneous sources. According to Ref. [41], the quality of information from each data source depends on three factors:

1) Error in measurements or precision of data collection.
2) Devices' noise in the environment.
3) Discrete observation and measurements.

To achieve a better Quality of Information (QoI), it is necessary to extract higher levels of abstraction and provide actionable information to other services. QoI in smart data depends on the applications and characteristics of data. There exist different solutions to improve QoI. For example, to improve the accuracy of data, selecting trustworthy sources and combining the data from multiple resources is of key importance. By increasing the frequency and density of sampling, the precision of the observations and measurements will be improved, which would lead to a reduction in environmental noise. The data characteristics in both IoT and the smart city are illustrated in Fig. 2. Semantic data annotation is another prime solution to enhancing data quality. Smart devices generate raw data with low-level abstractions. For this reason, the semantic models provide interpretable descriptions of data, its quality, and its original attributes [28]. Semantic annotation is beneficial in interpretable and knowledge-based information fusion [42]. Smart data characteristics in smart cities are tabulated in brief, in Table 1.

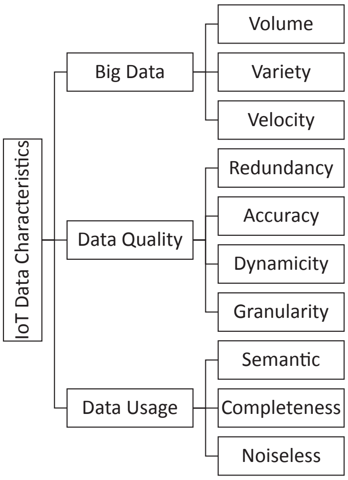

Fig. 2. Data characteristics.

5. Taxonomy of machine learning algorithms

Machine learning is a subfield of computer science, and is a type of Artificial Intelligence (AI) that provides machines with the ability to learn without explicit programming. Machine learning evolved from pattern recognition and computational learning theory. Here, some essential concepts of machine learning are discussed as well as the frequently applied machine learning algorithms for smart data analysis.

A learning algorithm takes a set of samples as an input named a *training set*. In general, there exist three main categories of learning: *supervised*, *unsupervised*, and *reinforcement* [54–56]. In an informal sense, in supervised learning, the training set consists of samples of input vectors together with their corresponding appropriate target vectors, also known as *labels*. In unsupervised learning, no labels are required for the training set. Reinforcement learning deals with the problem of learning the appropriate action or sequence of actions to be taken for a given situation in order to maximize payoff. This article focuses on supervised and unsupervised learning since they have been and are still widely applied in IoT smart data analysis. The objective of supervised learning is to learn how to predict the appropriate output vector for a given input vector. Applications where the target labels consist of a finite number of discrete categories are known as *classification* tasks. Cases where the target labels are composed of one or more continuous variables are known as *regression* tasks [57].

Defining the objective of unsupervised learning is difficult. One of the major objectives is to identify sensible clusters of similar samples within the input data, known as *clustering*. Moreover, the objective may be the discovery of a useful internal representation of the input data by *preprocessing* the original input variable in order to transfer it into a new variable space. This preprocessing stage can significantly improve the result of the subsequent machine learning algorithm and is named *feature extraction* [55].

The frequently applied machine learning algorithms for smart data analysis and IoT use cases are shown in Table 2 and Table 3 accordingly.

In the following subsections, we assume that we are given a training set containing N training samples denoted as $\{(x_i, y_i)\}_{i=1}^{N}$, where x_i is the ith M-dimensional training input vector and y_i is the corresponding

Table 2
Overview of frequently used machine learning algorithms for smart data analysis.

Machine learning algorithm	Data processing tasks	Section	Representative references
K-Nearest Neighbors	Classification	5.1.1	[58,59]
Naive Bayes	Classification	5.1.2	[60,61]
Support Vector Machine	Classification	5.1.3	[62–65]
Linear Regression	Regression	5.2.1	[66–68]
Support Vector Regression	Regression	5.2.2	[69,70]
Classification and Regression Trees	Classification/ Regression	5.3.1	[71–73]
Random Forests	Classification/ Regression	5.3.2	[74]
Bagging	Classification/ Regression	5.3.3	[75]
K-Means	Clustering	5.4.1	[76–78]
Density-Based Spatial Clustering of Applications with Noise	Clustering	5.4.2	[79–81]
Principal Component Analysis	Feature extraction	5.5.1	[82–86]
Canonical Correlation Analysis	Feature extraction	5.5.2	[87,88]
Feed Forward Neural Network	Regression/ Classification/ Clustering/Feature extraction	5.6.1	[89–93,57]
One-class Support Vector Machines	Anomaly detection	5.8.1	[94,95]

Table 3
Overview of applying machine learning algorithms to Internet of Things use cases.

Machine learning Algorithm	IoT, Smart City use cases	Metric to Optimize	References
Classification	Smart Traffic	Traffic Prediction, Increase Data Abbreviation	[43,14]
Clustering	Smart Traffic, Smart Health	Traffic Prediction, Increase Data Abbreviation	[43,14,44]
Anomaly Detection	Smart Traffic, Smart Environment	Traffic Prediction, Increase Data Abbreviation, Finding Anomalies in Power Dataset	[43,14,45]
Support Vector Regression	Smart Weather Prediction	Forecasting	[46]
Linear Regression	Economics, Market analysis, Energy usage	Real Time Prediction, Reducing Amount of Data	[48,148]
Classification and Regression Trees	Smart Citizens	Real Time Prediction, Passengers Travel Pattern	[48,47]
Support Vector Machine	All Use Cases	Classify Data, Real Time Prediction	[109,48]
K-Nearest Neighbors	Smart Citizen	Passengers' Travel Pattern, Efficiency of the Learned Metric	[47,96]
Naive Bayes	Smart Agriculture, Smart Citizen	Food Safety, Passengers Travel Pattern, Estimate the Numbers of Nodes	[49,47,148]
K-Means	Smart City, Smart Home, Smart Citizen, Controlling Air and Traffic	Outlier Detection, fraud detection, Analyze Small Data set, Forecasting Energy Consumption, Passengers Travel Pattern, Stream Data Analyze	[50,52,116,38, 47,117]
Density-Based Clustering	Smart Citizen	Labeling Data, Fraud Detection, Passengers Travel Pattern	[109,52,47]
Feed Forward Neural Network	Smart Health	Reducing Energy Consumption, Forecast the States of Elements, Overcome the Redundant Data and Information	[21,126,148]
Principal Component Analysis	Monitoring Public Places	Fault Detection	[51]
Canonical Correlation Analysis	Monitoring Public Places	Fault Detection	[51]
One-class Support Vector Machines	Smart Human Activity Control	Fraud Detection, Emerging Anomalies in the data	[52,53]

desired P-dimensional output vector. Moreover, we collect the M-dimensional input vectors in a matrix, written as $\mathbf{x} \equiv (x_1, \ldots, x_N)^T$, and we also collect their corresponding desired output vectors in a matrix written as $\mathbf{y} \equiv (y_1, \ldots, y_N)^T$. However, in Section 5.4 the training set does not contain the desired output vectors.

5.1. Classification

5.1.1. K-nearest neighbors

In K-Nearest Neighbors (KNN), the objective is to classify a new given unseen data point by looking at the K given data points in the training set that are closest to it in the input or feature space. Therefore, to find the K nearest neighbors of the new data point, we need to use a distance metric, such as Euclidean distance, L_∞ norm, angle, Mahalanobis distance or Hamming distance. To formulate the problem, let us denote the new input vector (data point) by x, its K nearest neighbors by $N_k(x)$, the predicted class label for x by y, and the class variable by a discrete random variable t. Additionally, $1(.)$ denotes the *indicator function*: $1(s) = 1$ if s is true and $1(s) = 0$ otherwise. The form of the classification task is

$$\begin{aligned} p(t = c|x, K) &= \frac{1}{K} \sum_{i \in N_k(x)} 1(t_i = c) \\ y &= \arg\max_{c} p(t = c|x, K) \end{aligned} \tag{1}$$

i.e., the input vector x will be labeled by the mode of its neighbors' labels [58].

One limitation of KNN is that it requires storing the entire training set, which makes KNN unscalable to large data sets. In [59], the authors have addressed this issue by constructing a tree-based search with a one-off computation. Moreover, there exists an online version of KNN classification. It is worth noting that KNN can also be used for regression tasks [55]. However, we will not explain this here, because it is not a frequently employed algorithm for smart data analysis. Ref. [96] proposes a new framework for learning a combination of multiple metrics for a robust KNN classifier. Furthermore, [47] compares KNN with a rough-set-based algorithm for classifying the travel pattern regularities.

5.1.2. Naive Bayes

Given a new, unseen data point (input vector) $z = (z_1, \ldots, z_M)$, naive Bayes classifiers, which are a family of probabilistic classifiers, classify z based on applying Bayes' theorem with the “naive” assumption of independence between the features (attributes) of z given the class variable t. By applying Bayes' theorem, we have

$$p(t = c|z_1, \ldots, z_M) = \frac{p(z_1, \ldots, z_M|t = c)p(t = c)}{p(z_1, \ldots, z_M)} \tag{2}$$

and by applying the naive independence assumption and some simplifications, we have

$$p(t = c|z_1, \ldots, z_M) \propto p(t = c) \prod_{j=1}^{M} p(z_j|t = c) \tag{3}$$

Therefore, the form of the classification task is

$$y = \arg\max_{c} p(t = c) \prod_{j=1}^{M} p(z_j|t = c) \tag{4}$$

where y denotes the predicted class label for z. Different naive Bayes classifiers use different approaches and distributions to estimate $p(t = c)$ and $p(z_j|t = c)$ [61].

Naive Bayes classifiers require a small number of data points to be trained, and can deal with high-dimensional data points, and also are fast and highly scalable [60]. Moreover, they are a popular model for applications, such as spam filtering [97], text categorization, and automatic medical diagnosis [98]. Ref. [49] used this algorithm to combine factors to evaluate the trust value and calculate the final quantitative trust of agricultural product.

5.1.3. Support vector machine

Classical Support Vector Machines (SVMs) are non-probabilistic, binary classifiers that aim to find the dividing hyperplane that separates both classes of the training set with the maximum margin. Then, the predicted label of a new, unseen data point is determined based on which side of the hyperplane it falls [62]. First, we discuss the linear SVM that finds a hyperplane that is a linear function of the input variable. To formulate the problem, we denote the normal vector to the hyperplane by w and the parameter for controlling the offset of the hyperplane from the origin along its normal vector by b. Moreover, to ensure that SVMs can deal with outliers in the data, we introduce a variable ξ_i, called a *slack variable*, for every training point x_i , which gives the distance by which this training point violates the margin in units of $|w|$. This binary linear classification task is described using a constrained optimization problem of the form

$$\begin{aligned}&\underset{w,b,\xi}{\text{minimize}} \quad f(w,b,\xi) = \frac{1}{2}w^T w + C\sum_{i=1}^{n}\xi_i \\ &\text{subject to} \quad y_i(w^T x_i + b) - 1 + \xi_i \geq 0 \quad i = 1,\dots,n, \\ &\xi_i \geq 0 \quad i = 1,\dots,n.\end{aligned} \tag{5}$$

where parameter $C > 0$ determines how heavily a violation is punished [63,64]. It should be noted that although we used the L_1 norm here for the penalty term $\sum_{i=1}^{n}\xi_i$, there exist other penalty terms, such as the L_2 norm, which should be chosen with respect to the needs of the application. Moreover, the parameter C is a hyperparameter which can be chosen via cross-validation or Bayesian optimization. There exist various techniques to solve the constrained optimization problem of equation (5), such as quadratic programming optimization [99], sequential minimal optimization [100], and P-packSVM [101]. One important property of SVMs is that the resulting classifier only uses a few training points, which are called *support vectors*, to classify a new data point. In addition to performing linear classification, SVMs can perform a non-linear classification, which finds a hyperplane that is a non-linear function of the input variable. To do so, we implicitly map an input variable into high-dimensional feature spaces in a process called a *kernel trick* [64]. In addition to performing binary classification, SVMs can perform multiclass classification. There are various methods to achieve this, such as One-vs-all (OVA) SVM, All-vs-all (AVA) SVM [54], Structured SVM [102], and the Weston and Watkins [103] version.

SVMs are among the best *off-the-shelf* supervised learning models that are capable of effectively dealing with high-dimensional data sets and are efficient in terms of memory usage, owing to the employment of support vectors for prediction. One significant drawback of this model is that it does not directly provide probability estimates. When given a solved SVM model, its parameters are difficult to interpret [104]. SVMs are useful in many real-world applications such as hand-written character recognition [105], image classification [106], and protein classification [107]. Finally, we should note that SVMs can be trained in an online fashion, which is addressed in [108]. Ref. [109] proposed a method on the Intel Lab Dataset. This data set consists of four environmental variables (temperature, voltage, humidity and light) collected through S4 Mica2Dot sensors over 36 days at a per-second rate.

5.2. Regression

5.2.1. Linear regression

In linear regression, the objective is to learn a function $f(x,w)$. This is a mapping $f : \phi(x) \to y$, and is a linear combination of a fixed set of linear or nonlinear functions of the input variable, denoted as $\phi_i(x)$ and called basic functions. The form of $f(x,w)$ is

$$f(x,w) = \phi(x)^T w \tag{6}$$

where w is the weight vector or matrix $w = (w_1,\dots,w_D)^T$, and $\phi = (\phi_1,\dots,\phi_D)^T$. There exists a broad range of basic functions, such as polynomial, gaussian, radial, and sigmoidal basic functions, which should be chosen with respect to the application [66,68].

For training the model, there exists a range of approaches: Ordinary Least Square, Regularized Least Squares, Least-Mean-Squares (LMS) and Bayesian Linear Regression. Among these, LMS is of interest since it is fast, scalable to large data sets, and it learns the parameters online by applying the technique of stochastic gradient descent, also known as sequential gradient descent [55,67].

By using suitable basic functions it can be shown that arbitrary nonlinearities in the mapping from the input variable to the output variable can be modeled. However, the assumption of fixed basis functions leads to significant shortcomings with this approach. For example, an increase in the dimension of the input space is coupled with rapid growth in the number of basis functions [55,56,66]. Linear regression can process at a high rate [48], and this algorithm can be used to analyze and predict the energy usage of buildings.

5.2.2. Support vector regression

The SVM model described in Section 5.1.3 can be extended to solve regression problems through a process called Support Vector Regression (SVR). Analogous to support vectors in SVMs, the resulting SVR model depends only on a subset of the training points, owing to the rejection of training points that are close to the model prediction [69]. Various implementations of SVR exist such as epsilon-support vector regression and nu-support vector regression [70]. The authors of [46] proposed a hybrid method to obtain accurate temperature and humidity data predictions.

5.3. Combining models

5.3.1. Classification and regression trees

In Classification and Regression Trees (CART), the input space is partitioned into axis-aligned cuboid regions R_k, and then a separate classification or regression model is assigned to each region to predict the label for data points fall into that region [71]. Given a new, unseen input vector (data point) x, the process for predicting the corresponding target label can be explained by the traversal of a binary tree corresponding to a sequential decision-making process. An example of a model for classification is one that predicts a particular class for each region, and the example of regression is a model that predicts a constant for each region. To formulate the classification task, we denote a class variable by a discrete random variable t and the predicted class label for x by y. The classification task takes the form

$$\begin{aligned}&p(t = c|k) = \frac{1}{|R_k|}\sum_{i\in R_k}1(t_i = c), \\ &y = \arg\max_{c} p(t = c|x) = \arg\max_{c} p(t = c|k)\end{aligned} \tag{7}$$

where $1(.)$ is the indicator function described in Section 5.1.1. This equation states that x will be labeled by the most common (mode) label in its corresponding region [73].

To formulate the regression task, we denote the value of the output vector by t and the predicted output vector for x by y. The regression task is expressed as

$$y = \frac{1}{|R_k|}\sum_{i\in R_k}t_i \tag{8}$$

i.e., the output vector for x will be the mean of the output vectors of data points in its corresponding region [73].

To train CART, the structure of the tree should be determined based on the training set. This means determining the split criterion at each

```
ALGORITHM 1: Algorithm for Training CART
Input: labeled training data set $D = \{(x_i, y_i)\}_{i=1}^{N}$.
Output: Classification or regression tree.
fitTree(0, $D$, node)
  function fitTree(depth, $R$, node)
    if the task is classification then
      node.prediction := most common label in $R$
    else
      node.prediction := mean of the output vector of the data points in $R$
    end
    $(i^*, z^*, R_L, R_R)$ := split($R$)
    if worth splitting and stopping criteria is not met then
      node.test := $x_{i^*} < z^*$
      node.left := fitTree(depth + 1, $R_L$, node)
      node.right := fitTree(depth + 1, $R_R$, node)
    end
    return node
```

node, along with the threshold parameter value. Finding the optimal tree structure is an NP-complete problem, and therefore a greedy heuristic, which constructs the tree top-down and chooses the best split node-by-node, is used to train CART. To achieve a better generalization and reduce overfilling, some stopping criteria should be used for constructing the tree. Possible stopping criterion are the maximum depth reached, whether the distribution in the branch is pure, whether the benefit of splitting is below a certain threshold, and whether the number of samples in each branch is below the criteria threshold. Moreover, after constructing the tree a pruning procedure can be used to reduce overfitting [55,56,72]. Algorithm 1 describes how to train CART.

The major strength of CART is its human interpretability, owing to its tree structure. Additionally, it is fast and scalable to large data sets. However, it is very sensitive to the choice of the training set [110]. Another shortcoming of this model concerns unsmooth labeling of the input space since each region of the input space is associated with exactly one label [55,73]. Ref. [47] proposes an efficient and effective data-mining procedure that models the travel patterns of transit riders in Beijing, China.

5.3.2. Random forests

In random forests, instead of training a single tree, an army of trees are trained. Each tree is trained on a subset of the training set, chosen randomly along with a replacement, using a randomly chosen subset of M input variables (features) [74]. From here, there are two scenarios for predicting the label of a new, unseen data point: (1) in classification tasks, this is set as the mode of the labels predicted by each tree; (2) in regression tasks it is set as the mean of the labels predicted by each tree. There is a tradeoff between different values of M. A value of M that is too small leads to random trees with poor prediction power, whereas a value of M that is too large leads to very similar random trees.

Random forests have a very high accuracy, but this comes at the cost of losing human interpretability [111]. Additionally, they are fast and scalable to large data sets, and have many real-world applications, such as body pose recognition [112] and body part classification.

5.3.3. Bagging

Bootstrap aggregating, also called bagging, is an ensemble technique that aims to improve the accuracy and stability of machine learning algorithms and reduce overfitting. In this technique, K new M sized training sets are generated by randomly choosing data points from the original training set with replacements. Then, on each new generated training set, a machine learning model is trained. The predicted label of a new, unseen data point is set as the mode of the labels predicted by each model in the case of classification tasks, and is set as the mean in the case of regression tasks. There are various machine learning models, such as CART and neural networks, for which the bagging technique can improve the results. However, bagging degrades the performance of stable models such as KNN [75]. Examples of practical applications include customer attrition prediction [113] and preimage learning [114,115].

5.4. Clustering

5.4.1. K-means

In the K-means algorithm, the objective is to cluster the unlabeled data set into K clusters (groups), where data points belonging to the same cluster must have some similarities. In the classical K-means algorithm, the distance between data points is the measure of similarity. Therefore, K-means seeks to find a set of K cluster centers, denoted as $\{s_1, \dots, s_k\}$, such that the distances between data points and their nearest center are minimized [77]. To denote the assignment of data points to the cluster centers, we use a set of binary indicator variables $\pi_{nk} \in \{0, 1\}$, so that if the data point x_n is assigned to the cluster center s_k, then $\pi_{nk} = 1$. We formulate the problem as follows:

$$\begin{aligned}&\underset{s,\pi}{\text{minimize}}\sum_{n=1}^{N}\sum_{k=1}^{K}\pi_{nk}\|x_n - s_k\|^2\\&\text{subject to}\sum_{k=1}^{K}\pi_{nk} = 1,\ n = 1, \dots, N.\end{aligned} \tag{9}$$

Algorithm 2 describes how to learn the optimal cluster centers $\{s_k\}$ and the assignment of the data points $\{\pi_{nk}\}$.

ALGORITHM 2: K-means Algorithm

```
Input: $K$, and unlabeled data set $\{x_1, \ldots, x_N\}$.
Output: Cluster centers $\{s_k\}$ and the assignment of the data points $\{\pi_{nk}\}$.
Randomly initialize $\{s_k\}$.
repeat
    for $n := 1$ to $N$ do
        for $k := 1$ to $K$ do
            if $k = \arg\min_i \|s_i - x_i\|^2$ then
                $\pi_{nk} := 1$
            else
                $\pi_{nk} := 0$
            end
        end
    end
    for $k := 1$ to $K$ do
        $s_k := \frac{\sum_{n=1}^{N} x_n \pi_{nk}}{\sum_{n=1}^{N} \pi_{nk}}$
    end
until $\{\pi_{nk}\}$ or $\{s_k\}$ don't change;
```

In practice, K-means is a very fast and highly scalable algorithm. Moreover, there is an online stochastic version of K-means [78]. However, this approach has many limitations because of the use of Euclidean distance as the measure of similarity. For instance, it has limitations regarding the types of data variables that can be considered, and cluster centers are not robust against outliers. Additionally, the K-means algorithm assigns each data point only one of the clusters, which may lead to inappropriate clusters in some cases [76]. Ref. [116] used MapReduce to analyze the numerous small data sets and proposes a cluster strategy for a high volume of small data based on the K-means algorithm. Ref. [47] applied K-Means++ to cluster and classify travel pattern regularities. Ref. [117] introduced a real-time event processing and clustering algorithm for analyzing sensor data, by using the OpenIoT1 middleware as an interface for innovative analytical IoT services.

5.4.2. Density-based spatial clustering of applications with noise

In a Density-Based approach to Spatial Clustering of Applications with Noise (DBSCAN), the objective is to cluster a given unlabeled data set based on the density of its data points. In this model, groups of dense data points (data points with many close neighbors) are considered as clusters and data points in regions with low-density are considered as outliers [80]. Ref. [79] present an algorithm to train a DBSCAN model.

In practice, DBSCAN is efficient on large datasets, and is fast and robust against outliers. Furthermore, it is capable of detecting clusters of arbitrary shape (i.e., spherical, elongated, and linear). Moreover, the model determines the number of clusters based on the density of the data points, unlike K-means, which requires the number of clusters to be specified [79]. However, there are some disadvantages associated with DBSCAN. For example, in the case of a data set with large differences in densities, the resulting clusters are destitute. Additionally, the performance of the model is highly sensitive to the distance metric that is used for determining if a region is dense [81]. It is worth, however, noting that DBSCAN is among the most widely used clustering algorithms with numerous real-world applications such as anomaly detection in temperature data [118] and X-ray crystallography [79]. The authors of [109] believe that knowledge discovery in data streams is an important task for research, business, and community. They applied DBSCAN to a data stream to reveal the number of existing classes and subsequently label the data. Furthermore, in [52] this algorithm is adopted to determine the arbitrary shape of a cluster. The DBSCAN algorithm produces sets of clusters of arbitrary shape and outlying objects.

5.5. Feature extraction

5.5.1. Principal component analysis

In Principle Component Analysis (PCA), the objective is to orthogonally project data points onto an L dimensional linear subspace, called the *principal subspace*, which has the maximal projected variance [83,85]. Equivalently, the objective can be defined as finding a complete orthonormal set of L linear M-dimensional basis vectors $\{w_j\}$ and the corresponding linear projections of data points $\{z_{nj}\}$ such that the average reconstruction error

$$\begin{aligned} J &= \frac{1}{N}\sum_n \|\tilde{x}_n - x_n\|^2 \\ \tilde{x}_n &= \sum_{j=1}^{L} z_{nj} w_j + \bar{x} \end{aligned} \tag{10}$$

is minimized, where $\bar{x}$ is the average of all data points [55,82].

```
ALGORITHM 3: PCA Algorithm
Input: $L$, and input vectors of an unlabeled or labeled data set $\{x_1, \ldots, x_N\}$.
Output: The projected data set $\{z_1, \ldots, z_N\}$, and basis vectors $\{w_j\}$ which form
        the principal subspace.
$\bar{x} := \frac{1}{N}\sum_n x_n$
$S := \frac{1}{N}\sum_n (x_n - \bar{x})(x_n - \bar{x})^T$
$\{w_j\}$ := the $L$ eigenvectors of $S$ corresponding to the $L$ largest eigenvalues.
for $n := 1$ to $N$ do
    for $j := 1$ to $L$ do
        $z_{nj} := (x_n - \bar{x})^T w_j$
    end
end
```

Algorithm 3 describes how the PCA technique achieves these objectives. Depending on how $\{w_1, \ldots, w_L\}$ is calculated, the PCA algorithm can have different run times, i.e., $O(M^3)$, $O(LM^2)$, $O(NM^2)$, and $O(N^3)$ [55,119,120]. To deal with high dimensional data sets, there is alternative version of the PCA algorithm based on the iterative *expectation maximization* technique. In this algorithm, the covariance matrix of the dataset is not explicitly calculated, and its most computationally demanding steps are $O(NML)$. In addition, this algorithm can be implemented in an online fashion, which can also be advantageous in cases where M and N are large [56,84].

PCA is one of the most important preprocessing techniques in machine learning. Its application involves data compression, whitening, and data visualization. Examples of its practical applications are face recognition, interest rate derivative portfolios, and neuroscience. Furthermore, there exists a kernelized version of PCA, called KPCA, which can find nonlinear principal components [84,86].

5.5.2. Canonical correlation analysis

Canonical Correlation Analysis (CCA) is a linear dimensionality reduction technique that is closely related to PCA. Unlike PCA, which deals with one variable, CCA deals with two or more variables. Its objective is to find a corresponding pair of highly cross-correlated linear subspaces so that within one of the subspaces there is a correlation between each component and a single component from the other subspace. The optimal solution can be obtained by solving a generalized eigenvector problem [55,87,88]. Ref. [51] compared PCA with CCA for detecting intermittent faults and masking failures of indoor environments.

5.6. Neural network

One of the shortcomings of linear regression is that it is necessary to decide the types of basic functions. It is often difficult to choose the optimal basic functions. Therefore, in neural networks we fix the number of basic functions, but allow the model to learn the parameters of the basic functions. There exist many different types of neural networks, with different architectures, use cases, and applications. In subsequent subsections, we discuss the successful models used in smart data analysis. Note that neural networks are fast to process new data, because they are compact models. However, in contrast they usually require a large amount of computation to be trained. Moreover, they are easily adaptable to regression and classification problems [90,91].

5.6.1. Feed forward neural network

Feed Forward Neural Networks (FFNN), also known as *Multilayer Perceptrons* (MLP), are the most common type of neural networks in practical applications. To explain this model, we begin with a simple two-layer FFNN model. Assume that we have D basic functions and our objective is to learn the parameters of these basic functions together with the function f discussed in Section 5.2.1. The form of the classification or regression task is

$$f\left(x, w^{(1)}, w^{(2)}\right) = \phi^{(2)}\left(\phi^{(1)}\left(x^T w^{(1)}\right)^T w^{(2)}\right) \tag{11}$$

where $w^{(1)} = \left(w_1^{(1)}, \ldots, w_M^{(1)}\right)^T$, $\phi^{(1)} = \left(\phi_1^{(1)}, \ldots, \phi_D^{(1)}\right)^T$, $w^{(2)} = \left(w_1^{(2)}, \ldots, w_D^{(2)}\right)^T$, and $\phi^{(2)} = \left(\phi_1^{(2)}, \ldots, \phi_P^{(2)}\right)^T$. Fig. 3 visualizes this FFNN model. The elements of the input vector x are units (neurons) in the input layer, $\phi_i^{(1)}$ are the units in the hidden layer, and $\phi_i^{(2)}$ are the units in the output layer, which outputs f. Note that the activities of the units in each layer are determined by a nonlinear function of the activities in the previous layer. In machine learning literature, $\phi(.)$ is also called an *activation function*. The activation function in the last layer is chosen with respect to the data processing task. For example, for a regression task we use a linear activation function, and for multiclass classification we use a *softmax* activation function [55,57,91].

With enough hidden units, an FFNN with at least two layers can approximate an arbitrary mapping from a finite input space to a finite output space [121–123]. However, finding the optimum set of weights w *for an FFNN* is an NP-complete problem [124]. To train the model, there exist a range of learning methods, such as stochastic gradient descent, adaptive delta, adaptive gradient, adaptive moment estimation, Nesterov's accelerated gradient, and RMSprob. There also exist a range of

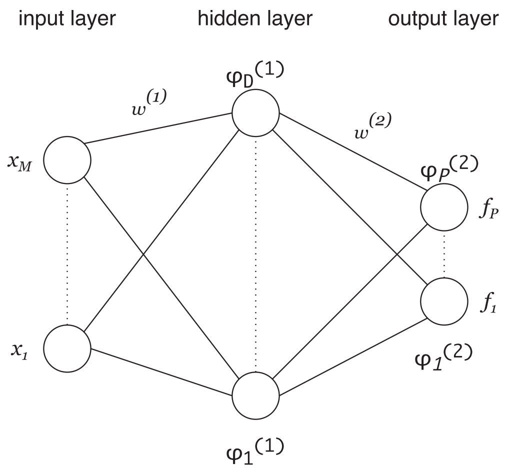

Fig. 3. A two-layer feed forward neural network. Note that each output neuron is connected to each input neuron, i.e., it is a fully connected neural network.

methods to enhance the generalization of the model and reduce overfitting, such as weight decay, weight-sharing, early stopping, Bayesian fitting of neural nets, dropout, and generative pre-training [89,92].

A two-layer FFNN has the properties of restricted representation and generalization. Moreover, compactly represented functions with l layers may require an exponential size with $l-1$ layers. Therefore, an alternative approach would be an FFNN with more than one hidden layers, i.e., a *deep neural network*, in which different high-level features share low-level features [90,93]. Significant results using deep neural networks have led them to be the most commonly employed classifiers in machine learning [57,125]. Ref. [126] presented a method to forecast the states of IoT elements based on an artificial neural network. The presented architecture of the neural network is a combination of a multilayered perceptron and a probabilistic neural network. Furthermore, [21] applied an FFNN for processing health data.

5.7. Time series and sequential data

So far in this article, the discussed algorithms have all dealt with set of data points that are independent and identically distributed (i.i.d.). However, in many cases the set of data points are not i.i.d., and often result from time series measurements, such as the daily closing value of the Dow Jones Industrial Average or acoustic features at successive time frames. An example of a non-i.i.d. set of data points in a context other than a time series is a character sequence in a German sentence. In these cases, data points consist of sequences of (x, y) pairs rather than being drawn in an i.i.d. manner from a joint distribution $p(\mathbf{x}, \mathbf{y})$, and such the sequences exhibit significant sequential correlations [55,127].

In a sequential supervised learning problem, when data points are sequential, we are given a training set $\{(x_i, y_i)\}_{i=1}^{N}$ consisting of N samples, where each of these is a pair of sequences. In each sample, $x_i = \langle x_{i,1}, x_{i,2}, \ldots, x_{i,T_i}\rangle$ and $y_i = \langle y_{i,1}, y_{i,2}, \ldots, y_{i,T_i}\rangle$. Given a new, unseen input sequence x, the goal is to predict the desired output sequence y. Moreover, there exists a closely related problem called a *time-series prediction* problem, in which the goal is to predict the desired $t+1^{\text{st}}$ element of a sequence $\langle y_1, \ldots, y_t\rangle$. The key difference is that unlike in sequential supervised learning, where the entire sequence $\langle x_1, \ldots, x_T\rangle$ is available prior to any prediction, in a time-series prediction only the prefix of the sequence, up to the current time $t+1$, is available. In addition, in sequential supervised learning the entire output sequence y must be predicted, whereas in time-series prediction the true observed values of the output sequence up to time t are given. It is worth noting that there is another closely-related task, called *sequence classification*, in which the goal is to predict the desired single categorical output y given an input sequence x [127].

There are a variety of machine learning models and methods that can deal with these tasks. Examples of these models and methods are hidden Markov models [128,129], sliding-window methods [130], Kalman filter [131], conditional random fields [132], recurrent neural networks [57,133], graph transformer networks [89], and maximum entropy Markov models [134]. In addition, sequential time series and sequential data exists in many real-world applications, including speech recognition [135], handwriting recognition [136], musical score following [137], and information extraction [134].

5.8. Anomaly detection

The problem of identifying items or patterns in a data set that do not conform to other items or an expected pattern is referred to as *anomaly detection,* and these unexpected patterns are called anomalies, outliers, novelties, exceptions, noise, surprises, or deviations [138,139].

There are many challenges in the task of anomaly detection that distinguish it from a binary classification task. For example, an anomalous class is often severely underrepresented in the training set. In addition, anomalies exhibit considerably more diverse behavior than a normal system, and are sparse by nature [139,140].

There are three broad categories of anomaly detection techniques, based on the extent to which the labels are available. In *supervised anomaly detection* techniques, a binary (abnormal and normal) labeled data set is given, and then a binary classifier is trained. This should deal with the problem of *unbalanced data set* resulting from the existence of few data points with an abnormal label. *Semi-supervised anomaly detection* techniques require a training set that contains only normal data points. Anomalies are then detected by building the normal behavior model of the system, and then testing the likelihood of the generation of the test data point by the learned model. *Unsupervised anomaly detection* techniques deal with an unlabeled data set, by making the implicit assumption that most of the data points are normal [139].

Anomaly detection is of use in many real-world applications, such as system health monitoring, credit card fraud detection, intrusion detection [141], detecting eco-system disturbances, and military surveillance. Moreover, anomaly detection can be used as a preprocessing algorithm for removing outliers from a data set, which can significantly improve the performance of subsequent machine learning algorithms, especially in supervised learning tasks [142,143]. In the following subsection, we shall explain *one-class* support *vector machines,* which represent one of the most popular techniques for anomaly detection. Ref. [45] constructed a novel outlier detection algorithm that uses statistical techniques to identify outliers and anomalies in power datasets collected from smart environments.

5.8.1. One-class support vector machines

One-class support vector machines (OCSVMs) represent a semi-supervised anomaly detection technique, and are an extension of the SVMs discussed in Section 5.1.3 for unlabeled data sets. Given a training set drawn from an underlying probability distribution P, OCSVMs aim to estimate a subset S of the input space such that the probability that a drawn sample from P lies outside of S is bounded by a fixed value between 0 and 1. This problem is approached by learning a binary function f that captures the input regions where the probability density lives. Therefore, f is negative in the complement of S. The functional form of f can be computed by solving a quadratic programming problem [94,95].

One-class SVMs are useful in many anomaly detection applications, such as anomaly detection in sensor networks [144], system-called intrusion detection [145], network intrusion detection [146], and anomaly detection in wireless sensor networks [147]. Ref. [52] reviewed various techniques for stream data outlier detection and the related issues in detail. Ref. [53] employed one-class SVM to detect anomalies by modeling complex normal patterns in the data.

In the following section, we discuss how to overcome the challenges of applying machine learning algorithms to IoT smart data.

6. Discussion of taxonomy of machine learning algorithms

To reach suitable decisions for smart data analysis, it is necessary to determine which task should be accomplished out of structure discovery, finding unusual data points, predicting values, predicting categories, or feature extraction.

To discover the structure of unlabeled data, clustering algorithms can provide the most appropriate tools. K-means, described in 5.4.1, is the most well-known and frequently applied clustering algorithm, and can handle a large volume of data with a broad range of data types. Refs. [50,52] proposed a method for applying the K-means algorithm for managing smart city and smart home data. DB-scan, described in 5.4.2, is another clustering algorithm the structure of data from unlabeled data, and is applied in [47,52,109] to cluster smart citizen behavior.

To find unusual data points and anomalies in smart data, two important algorithms can be applied. Namely, these are the one-class SVM and PCA-based anomaly detection methods explained in 5.5.1, which can train anomalies and noisy data with a high performance. Refs. [52,53] applied a one-class SVM to monitor and detect human activity anomalies.

In order to predict values and classify sequenced data, the linear regression and SVR methods described in 5.2.1 and 5.2.2, respectively, are the two frequently applied algorithms. The objective of the models applied in these algorithms is to process and train data of high velocity. For example, [46,48] applied a linear regression algorithm for real-time prediction. Another fast training algorithm is the classification and regression tree described in 5.3.1, which has been applied to classify smart citizen behaviors [47,48].

To predict the categories of data, neural networks are suitable learning models for function approximation problems. Moreover, because smart data should be accurate and requires a long training time, a multi-class neural network can provide an appropriate solution. For instance, the FFNN explained in 5.6.1 has been applied to reduce future energy consumption, by predicting how data will be generated in the future and how the redundancy of this data will be removed [21,126,148]. The SVM explained in 5.1.3 is another popular classification algorithm, which is capable of handling massive amounts of data and classifying their different types. Because SVM can handle a high volume and a variety of types of data, it is commonly applied in most smart data processing algorithms. For example, [48,109] applied SVM to classify traffic data.

PCA and CCA, described in 5.5.1 and 5.5.2, respectively, are the two algorithms most commonly applied for extracting features of data. Moreover, CCA can show the correlation between two categories of data. A type of PCA or CCA can also be applied to discover anomalies. Ref. [51] applied PCA and CCA to monitor public places and detect events in social areas.

The chosen algorithm should be implemented and developed to reach the correct decisions.

A sample implemented code is available from the open source GitHub license at https://github.com/mhrezvan/SVM-on-Smart-Traffic-Data.

7. Research trends and open issues

As discussed previously, data analysis provides a significant contribution to IoT. In order to exploit the full potential of data analysis for extracting new insights from data, IoT must overcome some major challenges. These can be categorized into three different types.

7.1. IoT data characteristics

Because the data is the basis of extracting knowledge, it is vital to have high quality information. This condition can affect the accuracy of knowledge extraction in a direct manner. Because IoT produces a high volume, fast velocity, and different varieties of data, preserving the data quality is a challenging task. Although many solutions have been introduced to solve these problems, none of them can handle all aspects of data characteristics in an accurate manner, because of the distributed nature of big data management solutions and real-time processing platforms. The abstraction of IoT data is low; that is, the data that comes from different resources in IoT consists mostly of raw data, and not sufficient for analysis. A wide variety of solutions have been proposed, while most of these require further improvement. For instance, semantic technologies tend to enhance the abstraction of IoT data through annotation algorithms, while they require further effort to overcome its velocity and volume.

7.2. IoT applications

IoT applications fall into different categories according to their unique attributions and features. Certain issues should be consider for running data analysis in IoT applications in an accurate manner. First, the privacy of collected data is highly critical, because data collection processes can include personal or critical business data, which is inevitable to solve the privacy issues. Second, according to the vast number of resources and simply-designed hardware in IoT, it is vital to consider security parameters, such as network security and data encryption. Otherwise, by ignoring security in the design and implementation, an infected network of IoT devices can lead to a crisis.

7.3. IoT data analytic algorithms

According to the characteristic of smart data, analytic algorithms should be able to handle big data. That is, IoT requires algorithms that can analyze data that comes from a variety of sources in real-time. Many attempts have been made to address this issue. For example, deep learning algorithms, which are a form of neural networks incorporating evolution can reach a high accuracy rate if they have enough data and time. Deep learning algorithms can easily be influenced by noisy smart data. Furthermore, neural network-based algorithms lack interpretation, that is, data scientists cannot understand the reasons for the model results. In the same manner, semi-supervised algorithms, which model a small amount of labeled data with a large amount of unlabeled data, can assist IoT data analysis.

8. Conclusions

IoT consists of a vast number of different devices that are connected with each other and transmit huge amounts of data. The smart city is one of the most important applications of IoT, and provides various services in domains such as energy, mobility, and urban planning. These services can be enhanced and optimized by analyzing the smart data collected from these areas. To extract knowledge from collected data, many data analytic algorithms can be applied. Choosing a suitable algorithm for specific IoT or smart city application is an important issue. In this article, many IoT data analytic studies are reviewed to address this issue. Here, three facts should be considered in applying data analytic algorithms to smart data. The first is that different applications in IoT and smart cities have their own characteristics, such as the number of devices and types of the data that they generate. The second is that the generated data will have specific features that should be considered. The third is that the taxonomy of algorithms is another important aspect in applying data analysis to smart data. The findings in this article make it easy to choose a suitable algorithm for a particular problem. The analytic algorithms fall into eight categories, described in detail. This is followed by reviewing application specifics of smart city use cases. The characteristics and qualities of smart data are described in detail. In the discussion section, the manner in which data characteristics and application specifics can lead to choosing a proper data analytic algorithms is reviewed. In the regarding future trends, section recent issues and the future path for research in the field of smart data analytics are discussed.

Acknowledgments

The authors would like to thank Dr. Mohammadsaeid Ehsani and Mr. Ajit Joakar for their comments on the draft of the paper. We also thank Dr. Alireza Ahrabian and Utkarshani Jiamini for reviewing our paper.

References

[1] L. Atzori, A. Iera, G. Morabito, The internet of things: a survey, Comput. Netw. 54 (15) (2010) 2787–2805.

[2] C. Cecchinel, M. Jimenez, S. Mosser, M. Riveill, An architecture to support the collection of big data in the internet of things, in: 2014 IEEE World Congress on Services, IEEE, 2014, pp. 442–449.

[3] M. Weiser, The computer for the 21st century, Mob. Comput. Commun. Rev. 3 (3) (1999) 3–11.

[4] A. Sheth, Computing for human experience: semantics-empowered sensors, services, and social computing on the ubiquitous web, IEEE Internet Comput. 14 (1) (2010) 88–91.

[5] J. Manyika, M. Chui, B. Brown, J. Bughin, R. Dobbs, C. Roxburgh, A.H. Byers, Big Data: The Next Frontier for Innovation, Competition, and Productivity, McKinsey Global Institute, 2011, 156 p.

[6] A. Sheth, Transforming big data into smart data: deriving value via harnessing volume, variety, and velocity using semantic techniques and technologies, in: Data Engineering (ICDE), 2014 IEEE 30th International Conference on, IEEE, 2014, 2–2.

[7] Amit Sheth, Transforming big data into smart data: deriving value via harnessing volume, variety and velocity using semantics and semantic web, in: keynote at the 21st Italian Symposium on Advanced Database Systems, June 30-July 03, 2013. Roccella Jonica, Italy.

[8] A. Sheth, Internet of things to smart iot through semantic, cognitive, and perceptual computing, IEEE Intell. Syst. 31 (2) (2016) 108–112.

[9] S. Bin, L. Yuan, W. Xiaoyi, Research on data mining models for the internet of things, in: 2010 International Conference on Image Analysis and Signal Processing, IEEE, 2010, pp. 127–132.

[10] H. Gonzalez, J. Han, X. Li, D. Klabjan, Warehousing and analyzing massive rfid data sets, in: 22nd International Conference on Data Engineering (ICDE'06), IEEE, 2006, 83–83.

[11] F. Chen, P. Deng, J. Wan, D. Zhang, A.V. Vasilakos, X. Rong, Data mining for the internet of things: literature review and challenges, Int. J. Distrib. Sens. Netw. 2015 (2015) 12.

[12] C.-W. Tsai, C.-F. Lai, M.-C. Chiang, L.T. Yang, Data mining for internet of things: a survey, IEEE Commun. Surv. Tutor. 16 (1) (2014) 77–97.

[13] A. Zanella, N. Bui, A. Castellani, L. Vangelista, M. Zorzi, Internet of things for smart cities, IEEE Internet Things J. 1 (1) (2014) 22–32.

[14] Y. Qin, Q.Z. Sheng, N.J. Falkner, S. Dustdar, H. Wang, A.V. Vasilakos, When things matter: a survey on data-centric internet of things, J. Netw. Comput. Appl. 64 (2016) 137–153.

[15] M. Ma, P. Wang, C.-H. Chu, Ltcep: efficient long-term event processing for internet of things data streams, in: 2015 IEEE International Conference on Data Science and Data Intensive Systems, IEEE, 2015, pp. 548–555.

[16] Payam Barnaghi, Amit Sheth, The Internet of things: The story so far, IEEE Internet Things, 915 (2014).

[17] Z. Sheng, S. Yang, Y. Yu, A.V. Vasilakos, J.A. McCann, K.K. Leung, A survey on the ietf protocol suite for the internet of things: standards, challenges, and opportunities, IEEE Wirel. Commun. 20 (6) (2013) 91–98.

[18] F. Bonomi, R. Milito, J. Zhu, S. Addepalli, Fog computing and its role in the internet of things, in: Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing, ACM, 2012, pp. 13–16.

[19] M. Aazam, E.-N. Huh, Fog computing micro datacenter based dynamic resource estimation and pricing model for iot, in: 2015 IEEE 29th International Conference on Advanced Information Networking and Applications, IEEE, 2015, pp. 687–694.

[20] Y. Shi, G. Ding, H. Wang, H.E. Roman, S. Lu, The fog computing service for healthcare, in: Future Information and Communication Technologies for Ubiquitous HealthCare (Ubi-HealthTech), 2015 2nd International Symposium on, IEEE, 2015, pp. 1–5.

[21] F. Ramalho, A. Neto, K. Santos, N. Agoulmine, et al., Enhancing ehealth smart applications: a fog-enabled approach, in: 2015 17th International Conference on E-health Networking, Application & Services (HealthCom), IEEE, 2015, pp. 323–328.

[22] A. Joakar, A Methodology for Solving Problems with DataScience for Internet of Things, Open Gardens (blog), July 21, 2016, http://www.opengardensblog.futuretext.com/archives/2016/07/a-methodology-for-solvingproblems-with-datascience-for-internet-of-things.html.

[23] A. Papageorgiou, M. Zahn, E. Kovacs, Efficient auto-configuration of energy-related parameters in cloud-based iot platforms, in: Cloud Networking (CloudNet), 2014 IEEE 3rd International Conference on, IEEE, 2014, pp. 236–241.

[24] L. Wang, R. Ranjan, Processing distributed internet of things data in clouds, IEEE Cloud Comput. 2 (1) (2015) 76–80.

[25] H. Zhao, C. Huang, A data processing algorithm in epc internet of things, in: Cyber-enabled Distributed Computing and Knowledge Discovery (CyberC), 2014 International Conference on, IEEE, 2014, pp. 128–131.

[26] R. Petrolo, V. Loscrì, N. Mitton, Towards a smart city based on cloud of things, a survey on the smart city vision and paradigms, Trans. Emerg. Telecommun. Technol. 28 (2017) e2931.

[27] E. Von Hippel, Democratizing innovation: the evolving phenomenon of user innovation, J. für Betriebswirtschaft 55 (1) (2005) 63–78.

[28] D. Puiu, P. Barnaghi, R. Tönjes, D. Kümper, M.I. Ali, A. Mileo, J.X. Parreira, M. Fischer, S. Kolozali, N. Farajidavar, et al., Citypulse: large scale data analytics framework for smart cities, IEEE Access 4 (2016) 1086–1108.

[29] B. Bowerman, J. Braverman, J. Taylor, H. Todosow, U. Von Wimmersperg, The vision of a smart city, in: 2nd International Life Extension Technology Workshop, Paris, vol. 28, 2000.

[30] J. Pan, R. Jain, S. Paul, T. Vu, A. Saifullah, M. Sha, An internet of things framework for smart energy in buildings: designs, prototype, and experiments, IEEE Internet Things J. 2 (6) (2015) 527–537.

[31] J. Torriti, Demand side management for the european supergrid: occupancy variances of european single-person households, Energy Policy 44 (2012) 199–206.

[32] Y. Wang, J. Yuan, X. Chen, J. Bao, Smart grid time series big data processing system, in: 2015 IEEE Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), IEEE, 2015, pp. 393–400.

[33] S. Karnouskos, T.N. De Holanda, Simulation of a smart grid city with software agents, in: Computer Modeling and Simulation, 2009. EMS'09. Third UKSim European Symposium on, IEEE, 2009, pp. 424–429.

[34] D.R. Nagesh, J.V. Krishna, S. Tulasiram, A real-time architecture for smart energy management, in: Innovative Smart Grid Technologies (ISGT), 2010, IEEE, 2010, pp. 1–4.

[35] T. Robles, R. Alcarria, D. Martín, A. Morales, M. Navarro, R. Calero, S. Iglesias, M. López, An internet of things-based model for smart water management, in: Advanced Information Networking and Applications Workshops (WAINA), 2014 28th International Conference on, IEEE, 2014, pp. 821–826.

[36] Z. Zhao, W. Ding, J. Wang, Y. Han, A hybrid processing system for large-scale traffic sensor data, IEEE Access 3 (2015) 2341–2351.

[37] M.M. Rathore, A. Ahmad, A. Paul, G. Jeon, Efficient graph-oriented smart transportation using internet of things generated big data, in: 2015 11th International Conference on Signal-image Technology & Internet-based Systems (SITIS), IEEE, 2015, pp. 512–519.

[38] C. Costa, M.Y. Santos, Improving cities sustainability through the use of data mining in a context of big city data, in: The 2015 International Conference of Data Mining and Knowledge Engineering, vol. 1, IAENG, 2015, pp. 320–325.

[39] A.J. Jara, D. Genoud, Y. Bocchi, Big data in smart cities: from poisson to human dynamics, in: Advanced Information Networking and Applications Workshops (WAINA), 2014 28th International Conference on, IEEE, 2014, pp. 785–790.

[40] H. Wang, O.L. Osen, G. Li, W. Li, H.-N. Dai, W. Zeng, Big data and industrial internet of things for the maritime industry in northwestern Norway, in: TENCON 2015-2015 IEEE Region 10 Conference, IEEE, 2015, pp. 1–5.

[41] P. Barnaghi, M. Bermudez-Edo, R. Tönjes, Challenges for quality of data in smart cities, J. Data Inf. Qual. (JDIQ) 6 (2–3) (2015) 6.

[42] A. Sheth, C. Henson, S.S. Sahoo, Semantic sensor web, IEEE Internet Comput. 12 (4) (2008) 78–83.

[43] M.A. Kafi, Y. Challal, D. Djenouri, M. Doudou, A. Bouabdallah, N. Badache, A study of wireless sensor networks for urban traffic monitoring: applications and architectures, Proc. Comput. Sci. 19 (2013) 617–626.

[44] D. Toshniwal, et al., Clustering techniques for streaming data-a survey, in: Advance Computing Conference (IACC), 2013 IEEE 3rd International, IEEE, 2013, pp. 951–956.

[45] V. Jakkula, D. Cook, Outlier detection in smart environment structured power datasets, in: Sixth International Conference on Intelligent Environments (IE), 2010, IEEE, 2010, pp. 29–33.

[46] P. Ni, C. Zhang, Y. Ji, A hybrid method for short-term sensor data forecasting in internet of things, in: 2014 11th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), 2014.

[47] X. Ma, Y.-J. Wu, Y. Wang, F. Chen, J. Liu, Mining smart card data for transit riders' travel patterns, Transp. Res. Part C Emerg. Technol. 36 (2013) 1–12.

[48] W. Derguech, E. Bruke, E. Curry, An autonomic approach to real-time predictive analytics using open data and internet of things, in: Ubiquitous Intelligence and Computing, 2014 IEEE 11th Intl Conf on and IEEE 11th Intl Conf on and Autonomic and Trusted Computing, and IEEE 14th Intl Conf on Scalable Computing and Communications and its Associated Workshops (UTC-ATC-ScalCom), IEEE, 2014, pp. 204–211.

[49] W. Han, Y. Gu, Y. Zhang, L. Zheng, Data driven quantitative trust model for the internet of agricultural things, in: Internet of Things (IOT), 2014 International Conference on the, IEEE, 2014, pp. 31–36.

[50] A.M. Souza, J.R. Amazonas, An outlier detect algorithm using big data processing and internet of things architecture, Proc. Comput. Sci. 52 (2015) 1010–1015.

[51] D.N. Monekosso, P. Remagnino, Data reconciliation in a smart home sensor network, Expert Syst. Appl. 40 (8) (2013) 3248–3255.

[52] M. Shukla, Y. Kosta, P. Chauhan, Analysis and evaluation of outlier detection algorithms in data streams, in: International Conference on Computer, Communication and Control (IC4), 2015, IEEE, 2015, pp. 1–8.

[53] A. Shilton, S. Rajasegarar, C. Leckie, M. Palaniswami, Dp1svm: a dynamic planar one-class support vector machine for internet of things environment, in: International Conference on Recent Advances in Internet of Things (RIoT), 2015, IEEE, 2015, pp. 1–6.

[54] D. Barber, Bayesian Reasoning and Machine Learning, Cambridge University Press, 2012.

[55] C.M. Bishop, Pattern Recognition and Machine Learning, Springer, 2006.

[56] K.P. Murphy, Machine Learning: a Probabilistic Perspective, MIT press, 2012.

[57] I.G.Y. Bengio, A. Courville, Deep Learning, Book in Preparation for MIT Press, 2016.

[58] T. Cover, P. Hart, Nearest neighbor pattern classification, IEEE Trans. Inf. Theory 13 (1) (1967) 21–27.

[59] H.V. Jagadish, B.C. Ooi, K.-L. Tan, C. Yu, R. Zhang, Idistance: an adaptive b+-tree based indexing method for nearest neighbor search, ACM Trans. Database Syst. (TODS) 30 (2) (2005) 364–397.

[60] A. McCallum, K. Nigam, et al., A comparison of event models for naive bayes text classification, in: AAAI-98 Workshop on Learning for Text Categorization, vol. 752, Citeseer, 1998, pp. 41–48.

[61] H. Zhang, The optimality of naive bayes, AA 1 (2) (2004) 3.

[62] C. Cortes, V. Vapnik, Support-vector networks, Mach. Learn. 20 (3) (1995) 273–297.

[63] I. Guyon, B. Boser, V. Vapnik, Automatic capacity tuning of very large vc-dimension classifiers, in: Advances in Neural Information Processing Systems, 1993, 147–147.

[64] N. Cristianini, J. Shawe-Taylor, An Introduction to Support Vector Machines and Other Kernel-based Learning Methods, Cambridge university press, 2000.

[65] B. Scholkopf, A.J. Smola, Learning with Kernels: Support Vector Machines, Regularization, Optimization, and beyond, MIT press, 2001.

[66] J. Neter, M.H. Kutner, C.J. Nachtsheim, W. Wasserman, Applied Linear Statistical Models, vol. 4, Irwin Chicago, 1996.

[67] G.A. Seber, A.J. Lee, Linear Regression Analysis, vol. 936, John Wiley & Sons, 2012.

[68] D.C. Montgomery, E.A. Peck, G.G. Vining, Introduction to Linear Regression Analysis, John Wiley & Sons, 2015.
[69] A. Smola, V. Vapnik, Support vector regression machines, Adv. Neural Inf. Process. Syst. 9 (1997) 155–161.
[70] A.J. Smola, B. Schölkopf, A tutorial on support vector regression, Statistics Comput. 14 (3) (2004) 199–222.
[71] L. Breiman, J. Friedman, C.J. Stone, R.A. Olshen, Classification and Regression Trees, CRC press, 1984.
[72] A.M. Prasad, L.R. Iverson, A. Liaw, Newer classification and regression tree techniques: bagging and random forests for ecological prediction, Ecosystems 9 (2) (2006) 181–199.
[73] W.-Y. Loh, Classification and regression trees, Wiley Interdiscip. Rev. Data Min. Knowl. Discov. 1 (1) (2011) 14–23.
[74] L. Breiman, Random forests, Mach. Learn. 45 (1) (2001) 5–32.
[75] L. Breiman, Bagging predictors, Mach. Learn. 24 (2) (1996) 123–140.
[76] A. Likas, N. Vlassis, J.J. Verbeek, The global k-means clustering algorithm, Pattern Recognit. 36 (2) (2003) 451–461.
[77] A. Coates, A.Y. Ng, Learning feature representations with K-Means, in: G. Montavon, G.B. Orr, K.R. Müller (Eds.), Neural Networks: Tricks of the Trade. Lecture Notes in Computer Science, vol. 7700, Springer, Berlin, Heidelberg, 2012.
[78] V. Jumutc, R. Langone, J.A. Suykens, Regularized and sparse stochastic k-means for distributed large-scale clustering, in: Big Data (Big Data), 2015 IEEE International Conference on, IEEE, 2015, pp. 2535–2540.
[79] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al., A density-based algorithm for discovering clusters in large spatial databases with noise, in: Kdd, vol. 96, 1996, pp. 226–231.
[80] H.-P. Kriegel, P. Kröger, J. Sander, A. Zimek, Density-based clustering, Wiley Interdiscip. Rev. Data Min. Knowl. Discov. 1 (3) (2011) 231–240.
[81] R.J. Campello, D. Moulavi, J. Sander, Density-based clustering based on hierarchical density estimates, in: Pacific-asia Conference on Knowledge Discovery and Data Mining, Springer, 2013, pp. 160–172.
[82] K. Pearson, Liii. on lines and planes of closest fit to systems of points in space, Lond. Edinb. Dublin Philos. Mag. J. Sci. 2 (11) (1901) 559–572.
[83] H. Hotelling, Analysis of a complex of statistical variables into principal components, J. Educ. Psychol. 24 (6) (1933) 417.
[84] I. Jolliffe, Principal Component Analysis, Wiley Online Library, 2002.
[85] H. Abdi, L.J. Williams, Principal component analysis, Wiley Interdiscip. Rev. Comput. Stat. 2 (4) (2010) 433–459.
[86] R. Bro, A.K. Smilde, Principal component analysis, Anal. Methods 6 (9) (2014) 2812–2831.
[87] H. Hotelling, Relations between two sets of variates, Biometrika 28 (3/4) (1936) 321–377.
[88] F.R. Bach, M.I. Jordan, Kernel independent component analysis, J. Mach. Learn. Res. 3 (Jul) (2002) 1–48.
[89] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324.
[90] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: Aistats, vol. 9, 2010, pp. 249–256.
[91] R.C. Eberhart, Neural Network PC Tools: a Practical Guide, Academic Press, 2014.
[92] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, Deep residual learning for image recognition, in: In Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.
[93] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436–444.
[94] B. Schölkopf, J.C. Platt, J. Shawe-Taylor, A.J. Smola, R.C. Williamson, Estimating the support of a high-dimensional distribution, Neural Comput. 13 (7) (2001) 1443–1471.
[95] G. Ratsch, S. Mika, B. Scholkopf, K.-R. Muller, Constructing boosting algorithms from svms: an application to one-class classification, IEEE Trans. Pattern Anal. Mach. Intell. 24 (9) (2002) 1184–1199.
[96] C.-T. Do, A. Douzal-Chouakria, S. Marié, M. Rombaut, Multiple metric learning for large margin knn classification of time series, in: Signal Processing Conference (EUSIPCO), 2015 23rd European, IEEE, 2015, pp. 2346–2350.
[97] V. Metsis, I. Androutsopoulos, G. Paliouras, Spam Filtering with Naive Bayes-which Naive Bayes? CEAS, 2006, pp. 27–28.
[98] G.I. Webb, J.R. Boughton, Z. Wang, Not so naive bayes: aggregating one-dependence estimators, Mach. Learn. 58 (1) (2005) 5–24.
[99] N.I. Gould, P.L. Toint, Numerical Analysis Group Internal Report, A Quadratic Programming Bibliography, vol. 1, 2000, p. 32.
[100] John Platt, Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines, 1998.
[101] Z.A. Zhu, W. Chen, G. Wang, C. Zhu, Z. Chen, P-packsvm: parallel primal gradient descent kernel svm, in: 2009 Ninth IEEE International Conference on Data Mining, IEEE, 2009, pp. 677–686.
[102] C.-N.J. Yu, T. Joachims, Learning structural svms with latent variables, in: Proceedings of the 26th Annual International Conference on Machine Learning, ACM, 2009, pp. 1169–1176.
[103] J. Weston, C. Watkins, et al., Support Vector Machines for Multi-class Pattern Recognition, vol. 99, ESANN, 1999, pp. 219–224.
[104] C.-W. Hsu, C.-J. Lin, A comparison of methods for multiclass support vector machines, IEEE Trans. Neural Netw. 13 (2) (2002) 415–425.
[105] H.-C. Kim, S. Pang, H.-M. Je, D. Kim, S.Y. Bang, Constructing support vector machine ensemble, Pattern Recognit. 36 (12) (2003) 2757–2767.
[106] G.M. Foody, A. Mathur, A relative evaluation of multiclass image classification by support vector machines, IEEE Trans. Geosci. Remote Sens. 42 (6) (2004) 1335–1343.
[107] C.S. Leslie, E. Eskin, W.S. Noble, The spectrum kernel: a string kernel for svm protein classification, in: Pacific symposium on Biocomputing, vol. 7, 2002, pp. 566–575.
[108] T. Poggio, G. Cauwenberghs, Incremental and decremental support vector machine learning, Adv. neural Inf. Process. Syst. 13 (2001) 409.
[109] M.A. Khan, A. Khan, M.N. Khan, S. Anwar, A novel learning method to classify data streams in the internet of things, in: Software Engineering Conference (NSEC), 2014 National, IEEE, 2014, pp. 61–66.
[110] H. Trevor, T. Robert, F. Jerome, The Elements of Statistical Learning: Data Mining, Inference and Prediction 1(8), Springer-Verlag, New York, 2001, pp. 371–406.
[111] R. Caruana, A. Niculescu-Mizil, An empirical comparison of supervised learning algorithms, in: Proceedings of the 23rd International Conference on Machine Learning, ACM, 2006, pp. 161–168.
[112] J. Shotton, T. Sharp, A. Kipman, A. Fitzgibbon, M. Finocchio, A. Blake, M. Cook, R. Moore, Real-time human pose recognition in parts from single depth images, Commun. ACM 56 (1) (2013) 116–124.
[113] T. Au, M.-L.I. Chin, G. Ma, Mining Rare Events Data by Sampling and Boosting: a Case Study, in: International Conference on Information Systems, Technology and Management, Springer, 2010, pp. 373–379.
[114] A. Sahu, G. Runger, D. Apley, Image denoising with a multi-phase kernel principal component approach and an ensemble version, in: 2011 IEEE Applied Imagery Pattern Recognition Workshop (AIPR), IEEE, 2011, pp. 1–7.
[115] A. Shinde, A. Sahu, D. Apley, G. Runger, Preimages for variation patterns from kernel pca and bagging, IIE Trans. 46 (5) (2014) 429–456.
[116] X. Tao, C. Ji, Clustering massive small data for iot, in: 2nd International Conference on Systems and Informatics (ICSAI), 2014, IEEE, 2014, pp. 974–978.
[117] H. Hromic, D. Le Phuoc, M. Serrano, A. Antonić, I.P. Žarko, C. Hayes, S. Decker, Real time analysis of sensor data for the internet of things by means of clustering and event processing, in: 2015 IEEE International Conference on Communications (ICC), IEEE, 2015, pp. 685–691.
[118] M. Çelik, F. Dadaşer-Çelik, A.Ş. Dokuz, Anomaly detection in temperature data using dbscan algorithm, in: Innovations in Intelligent Systems and Applications (INISTA), 2011 International Symposium on, IEEE, 2011, pp. 91–95.
[119] G.H. Golub, C.F. Van Loan, Matrix Computations, vol. 3, JHU Press, 2012.
[120] L. Sirovich, Turbulence and the dynamics of coherent structures part i: coherent structures, Q. Appl. Math. 45 (3) (1987) 561–571.
[121] G. Cybenko, Approximation by superpositions of a sigmoidal function, Mathematics of control, Signals Syst. 2 (4) (1989) 303–314.
[122] K. Hornik, Approximation capabilities of multilayer feedforward networks, Neural Netw. 4 (2) (1991) 251–257.
[123] K. Fukushima, Neocognitron: a self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position, Biol. Cybern. 36 (4) (1980) 193–202.
[124] W. Blum, D. Burghes, N. Green, G. Kaiser-Messmer, Teaching and learning of mathematics and its applications: first results from a comparative empirical study in england and Germany, Teach. Math. Appl. 11 (3) (1992) 112–123.
[125] J. Schmidhuber, Deep learning in neural networks: an overview, Neural Netw. 61 (2015) 85–117.
[126] I. Kotenko, I. Saenko, F. Skorik, S. Bushuev, Neural network approach to forecast the state of the internet of things elements, in: XVIII International Conference on Soft Computing and Measurements (SCM), 2015, IEEE, 2015, pp. 133–135.
[127] T.G. Dietterich, Machine learning for sequential data: a review, in: Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR), Springer, 2002, pp. 15–30.
[128] L.E. Baum, J.A. Eagon, et al., An inequality with applications to statistical estimation for probabilistic functions of markov processes and to a model for ecology, Bull. Am. Math. Soc. 73 (3) (1967) 360–363.
[129] L.R. Rabiner, A tutorial on hidden markov models and selected applications in speech recognition, Proc. IEEE 77 (2) (1989) 257–286.
[130] T.J. Sejnowski, C.R. Rosenberg, Parallel networks that learn to pronounce english text, Complex Syst. 1 (1) (1987) 145–168.
[131] R.E. Kalman, R.S. Bucy, New results in linear filtering and prediction theory, J. Basic Eng. 83 (1) (1961) 95–108.
[132] J. Lafferty, A. McCallum, F. Pereira, Conditional random fields: probabilistic models for segmenting and labeling sequence data, in: Proceedings of the Eighteenth International Conference on Machine Learning, vol. 1, ICML, 2001, pp. 282–289.
[133] R.J. Williams, D. Zipser, A learning algorithm for continually running fully recurrent neural networks, Neural Comput. 1 (2) (1989) 270–280.
[134] A. McCallum, D. Freitag, F.C. Pereira, Maximum Entropy Markov Models for Information Extraction and Segmentation, vol. 17, Icml, 2000, pp. 591–598.
[135] H. Sak, A.W. Senior, F. Beaufays, Long Short-term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modeling, 2014, pp. 338–342. Interspeech.
[136] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, J. Schmidhuber, A novel connectionist system for unconstrained handwriting recognition, IEEE Trans. Pattern Anal. Mach. Intell. 31 (5) (2009) 855–868.
[137] B. Pardo, W. Birmingham, Modeling form for on-line following of musical performances, in: Proceedings of the National Conference on Artificial Intelligence, vol. 20, AAAI Press; MIT Press, Menlo Park, CA; Cambridge, MA; London, 1999, p. 1018, 2005.
[138] V.J. Hodge, J. Austin, A survey of outlier detection methodologies, Artif. Intell. Rev. 22 (2) (2004) 85–126.
[139] V. Chandola, A. Banerjee, V. Kumar, Anomaly detection: a survey, ACM Comput. Surv. (CSUR) 41 (3) (2009) 15.

[140] Z.Á. Milacski, M. Ludersdorfer, A. Lőrincz, P. van der Smagt, Robust detection of anomalies via sparse methods, in: International Conference on Neural Information Processing, Springer, 2015, pp. 419–426.

[141] D.E. Denning, An intrusion-detection model, IEEE Trans. Softw. Eng. 2 (1987) 222–232.

[142] I. Tomek, An experiment with the edited nearest-neighbor rule, IEEE Trans. Syst. Man Cybern. 6 (1976) 448–452.

[143] M.R. Smith, T. Martinez, Improving classification accuracy by identifying and removing instances that should be misclassified, in: Neural Networks (IJCNN), The 2011 International Joint Conference on, IEEE, 2011, pp. 2690–2697.

[144] S. Rajasegarar, C. Leckie, J.C. Bezdek, M. Palaniswami, Centered hyperspherical and hyperellipsoidal one-class support vector machines for anomaly detection in sensor networks, IEEE Trans. Inf. Forensics Secur. 5 (3) (2010) 518–533.

[145] K.A. Heller, K.M. Svore, A.D. Keromytis, S.J. Stolfo, One class support vector machines for detecting anomalous windows registry accesses, in: Proc. Of the Workshop on Data Mining for Computer Security, vol. 9, 2003.

[146] M. Zhang, B. Xu, J. Gong, An anomaly detection model based on one-class svm to detect network intrusions, in: 2015 11th International Conference on Mobile Ad-hoc and Sensor Networks (MSN), IEEE, 2015, pp. 102–107.

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

[148] S. Hu, Research on data fusion of the internet of things, in: Logistics, Informatics and Service Sciences (LISS), 2015 International Conference on, IEEE, 2015, pp. 1–5.