# A Survey of Machine Learning for Network Fault Management


**Mourad Nouioua, Philippe Fournier-Viger, Ganghuan He, Farid Nouioua, and Zhou Min**

**Abstract** Telecommunication networks play a major role in today's society as they support the transmission of information between businesses, governments, and individuals. Hence, ensuring excellent service quality and avoiding service disruptions are important. For this purpose, fault management is critical. It consists of detecting, isolating, and fixing network problems, a task that is complex for large networks, and typically requires considerable resources. As a result, an emerging research area is to develop machine learning and data mining-based techniques to improve various aspects of the fault management process. This chapter provides a survey of data mining and machine learning-based techniques for fault management, including a description of their characteristics, similarities, differences, and shortcomings.

## 1 Introduction

Computer networks are crucial to today's society as they support communication between individuals, governments, and businesses. They are used not only to connect

M. Nouioua · P. Fournier-Viger (✉) · G. He
Harbin Institute of Technology (Shenzhen), Shenzhen, China
e-mail: philfv8@yahoo.com

M. Nouioua
e-mail: mouradnouioua@gmail.com

G. He
e-mail: heganghuan@gmail.com

M. Nouioua · F. Nouioua
University of Bordj Bou Arreridj, El Anceur, Algeria
e-mail: faridnouioua@gmail.com

F. Nouioua
Aix-Marseille Université, LIS, UMR-CNRS 7020, Marseille, France

Z. Min
Huawei Noah's Ark Lab, Shenzhen, China
e-mail: zhoumin27@huawei.com



H. Chiroma et al. (eds.), *Machine Learning and Data Mining for Emerging Trend in Cyber Dynamics*, https://doi.org/10.1007/978-3-030-66288-2_1

desktop computers, but also all kinds of electronic devices such as smartphones, wearable devices, sensors, and industrial machines. They also play a key role in emerging domains such as sensor networks [66, 87], vehicular networks [86], cloud computing [8], big data analysis [79], and the Internet of Things [81].

To ensure effective and efficient communication between devices, a network must be carefully designed in terms of physical and logical topology, and software must be properly configured. This requires considering various aspects such as budget, facilities, performance, and security requirements. Then, during a network's lifetime, various maintenance tasks must be carried out such as to replace, install, and upgrade equipment and software. Moreover, a key activity is *fault management*, which is carried out to ensure a network's security, availability, reliability, and optimize its performance [21, 71].

Fault management aims at solving problems that are occurring in a network. It consists of four main tasks, which are (1) Detecting, (2) Diagnosing, (3) Isolating, and (4) Fixing network faults [84, 87]. Fault management is not an easy task because faults may be caused by complex interactions between network devices and sometimes only appear for a short time. A good fault management process may consist of (1) Preventive measures and logging solutions that may raise alarms indicating potential problems, (2) A method for prioritizing the most important alarms or faults to analyze, and (3) Appropriate methods to isolate and fix the issues. Fault management can be quite time-consuming and costly, especially for large and heterogeneous networks. Hence, it has become critical to develop improved fault management techniques [84, 87].

Since more then two decades, some attempts at developing computer systems for fault management were made. For instance, in the 1990s, some expert systems were designed that relied on a knowledge base of rules to diagnose network problems. But a drawback of such systems was that specifying rules by hand requires expert knowledge, these rules would not be noise tolerant, and that writing these rules is time consuming and prone to errors [42, 60].

To build computer systems for fault management that do not rely heavily on domain experts, a promising fault management approach has been to apply data mining and machine learning-based techniques [5, 17, 19, 48, 56, 69, 74, 77]. These techniques allow to semi-automatically extract knowledge and learn models from data. Though there has been several studies in this direction, no survey has been published on this topic.

This chapter fills this gap by reviewing key studies on data mining and machine learning for fault management. The chapter provides a brief description of each study, their key ideas, and the advantages and limitations of the proposed techniques. The reviewed studies are categorized into two main categories based on the type of algorithms that they used: (1) Pattern mining-based approaches (e.g., itemset mining, association rule mining, and clustering) and (2) Machine learning-based approaches (e.g., neural networks, decision trees, Bayesian networks, and dependency graphs).

The rest of this chapter is organized as follows: Sect. 2 reviews some important concepts related to telecommunication network fault management. Then, Sects. 3

and 4 survey data mining-based approaches and machine learning-based approaches, respectively. Finally, a conclusion is drawn.

## 2 Network Fault Management

This section introduces important concepts related to telecommunication networks and fault management.

A computer network connects a set of devices that can exchange information and share resources. Typical devices found on a network are end-user computers (e.g., workstations), servers, mobile devices, and networking devices (e.g., routers and switches). Because networks can be used in a wide range of contexts and to address different use cases, numerous hardware and software technologies have been proposed for networking. Choosing a set of technologies requires considering various criteria such as cost, security, and performance. For example, some key requirements for building a sensor network may be to preserve battery life and perform distributed calculations. This is quite different from the requirements for large-scale computer networks such as the Internet, or those of a mobile GSM or UMTS network.

The goal of fault management is to detect, identify, and correct malfunctions in telecommunication networks [21, 71, 84, 87]. A *fault* is a malfunction that occurs on a network and may cause errors. A fault can have various consequences such as making a network device unavailable or degrading its performance. For example, a router hardware malfunction may cause the device to reboot and to be temporarily unavailable. Though a fault may cause several undesirable events, a fault is said to not be caused by any other events. In other words, a fault is the root cause of some error(s). An *error* is defined as a discrepancy between some observed values and some expected values or the violation of some conditions [71]. An error is caused by a fault and can propagate inside a network causing other errors.

Two main types of faults may occur [84, 87]: *hard faults* (a device cannot communicate with others) and *soft faults* (a device continues to operate but with an abnormal behavior such as sending corrupted data or incorrectly routing data). Moreover, from the perspective of time, faults can be categorized as *permanent* (an action must be taken to fix the issue), *temporary/transient* (the fault may appear only for a short time) and *intermittent* (the fault may periodically re-appear if no action is taken) [87].

Fault are sometimes not observable for various reasons such that no evidences have been collected. To more easily detect faults, alarms may be raised by network devices. An *alarm* is a symptom that can be observed of a potential fault. Alarms can be generated by devices or network management systems to provide information about potential faults that may cause errors. Alarms are very important as they allow to infer the existence of faults so that remediation steps can be taken [71]. Many alarms may appear in a network because a single fault may cause multiple alarms and because some alarms may be triggered in situations where no fault has occurred.

To facilitate alarm management, alarms are often categorized into different *severity levels* such as cleared, low, medium, high, and critical. Based on severity levels,

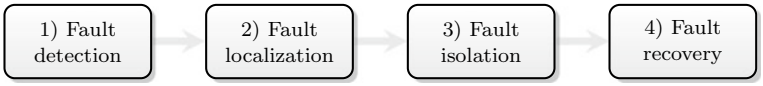


**Fig. 1** A generic network fault management process

alarms may be treated differently [74]. For instance, while an alarm of the cleared level may be ignored, an alarm considered as critical may trigger an emergency message sent by SMS to a network administrator so that he can take quick action. Moreover, alarms can be recorded in logs on each device or can be collected from all devices by some fault management software to facilitate alarm analysis. In *passive fault management*, each device is responsible of communicating its alarms to other devices or the user, while in *active fault management*, a device may be periodically probed by other devices to verify its state. Because device clocks may not be synchronized, it is sometimes not possible to know which alarms occurred first. A solution to this issue can be to use distributed algorithms to synchronize logical clocks [50]. A human can investigate an alarm and fix a fault. Besides, it is also possible to put recovery procedures into place (scripts) to automatically handle faults or errors detected by some specific alarm types [71].

Finding the reason(s) why some error occurs in a network can sometimes be quite difficult because of the complex and dynamic interaction between devices, and because some faults are not permanent, errors can propagate or are influenced by other faults or events. For example, some faults may only appear in some circumstances such as when the battery level of a sensor is low.

Managing faults in a telecommunication network is generally done by the following steps [71]: (1) Automatically collecting data about alarms generated by devices, (2) Preparing the data and enriching the data with additional information (if needed), (3) Identifying alarms that should be investigated with higher priority and inferring the root cause of alarms, (4) Applying recovery procedures or dispatching technicians to specific locations (physically or virtually) to isolate and fix the issues. This process is illustrated in Fig. 1. Step 1 is called *fault detection*, Step 2 is named *fault localization*, while Step 3 is sometimes called *fault localization*, *fault isolation* or *root cause analysis*. Finally, Step 4 is named *fault recovery* [71].

Another reason why fault management is challenging is that while thousands of alarms may be generated in network devices, the number of technicians or budget for maintaining a network is limited. Hence, it is easy for technicians to be overloaded with thousands of alarms and being unable to investigate all of them. Accordingly, it is critical to be able to prioritize some alarms and their relationships to investigate the most important alarms first. However, identifying the most important alarms is not easy. In some case, some fault in a device may cause many other alarms. Moreover, some alarms may only be triggered in some very specific and complex situations involving multiple devices.

The following Sects. 3 and 4 give a survey of the different approaches for fault management using pattern mining and machine learning techniques, respectively.

## 3 Pattern Mining-Based Approaches

To develop innovative fault management approaches, several studies have used pattern mining techniques. Pattern mining is a key task in the field of data mining, which consists of analyzing data to identify interesting patterns that may help to understand the data or support decision-making [3, 28, 57, 59, 64, 80]. Several pattern mining algorithms have been designed to search for patterns. To apply an algorithm, a user typically has to set constraints on patterns to be found. For example, some algorithms are designed to find frequent patterns (patterns that appear at least a minimum number of times in a database) [3, 28, 59, 64], while others for clustering can find groups of similar data records [80]. Most of pattern mining algorithms are unsupervised as the goal is to discover new knowledge.

Pattern mining techniques have been used to analyze telecommunication data and find various types of patterns for fault management. The next three subsections review studies that have used three main types of pattern mining techniques: (1) Episode and association rule mining, (2) Sequential pattern mining and (3) Clustering algorithms.

### *3.1 Episode and Association Rules Mining-Based Approaches*

Hatonen et al. [38] and Klemettinen et al. [48] first proposed using pattern mining techniques to analyze telecommunication network data. They designed a system called Telecommunication Alarm Sequence Analyser (TASA) to discover correlations between alarms.

Let $A$ be a set of alarm types and $T$ be a set of timestamps. The input of TASA is a sequence of alarms $S = \langle (a_1, t_1), (a_2, t_2), \ldots (a_n, t_n) \rangle$ that has been recorded from a network, where each alarm $a_i \in A (1 \leq i \leq n)$ is annotated with a timestamp $t_i \in T$. From such sequence of alarms, TASA applies a frequent episode mining algorithm [33, 34, 59] to extract *episode rules* [48].

An episode $E$ is a tuple that has the form $\langle a_1, a_2, \ldots a_k \rangle$ where $a_i \in A$, for all $i \in \{1, \ldots, k\}$. An episode rule has the form $E_1, E_2, \ldots E_m \Rightarrow_{(x,y)} a$, where $E_j \subseteq A (1 \leq j \leq m)$ and $a \in A$, and $x$ and $y$ are two amounts of time where $y > x$. This rule is interpreted as if all alarms of $E_1$ appears in any order, and then are followed by alarms of $E_2$ in any order, . . ., and then are followed by alarms of $E_m$ in any order, and all of this happens in no more than $x$ seconds, then the alarm $a$ will appear no more than $z = (y - x)$ seconds later.

To find interesting episode rules, two measures are used called the support and the confidence. Let there be an episode rule $\alpha = (X \Rightarrow_{(x,y)} Y)$ and a sequence $S$, where $X$ and $Y$ are the rule antecedent and consequent, respectively. The support of the episode rule $\alpha$ in TASA is the absolute number of occurrences of the episode $\langle X, Y \rangle$ in the sequence. Formally, given an episode rule $\alpha = X \Rightarrow_{(x,y)} Y$ and a sequence $S$, $\text{Support}(\alpha) = |Occ(\langle X, Y \rangle, S)|$, where $Occ(\langle X, Y \rangle, S)$ is the set of time intervals where the episode $\langle X, Y \rangle$ occurs in sequence $S$.

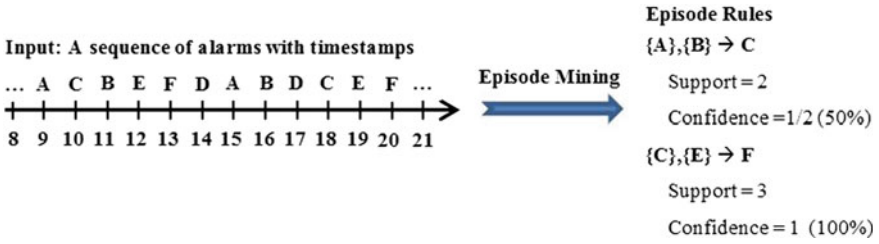


**Fig. 2** Example of discovering episode rules between alarms with TASA

The confidence measure can be interpreted as the conditional probability that the whole episode occurs within time $y$ given that the left side of the episode rule has already occurred within time $x$. Formally, we refer to the set of occurrences of an episode $\langle X, Y\rangle$ in a sequence $S$ within time $t$ as $Occ(\langle X, Y\rangle, S, t)$. Based on this, the confidence of an episode rule $\alpha = X \Rightarrow_{(x,y)} Y$ is Confidence$(\alpha) = \frac{|Occ(\langle X,Y\rangle,S,y)|}{|Occ(\langle X\rangle,S,x)|}$.

The TASA system outputs all rules having at least a minimum support and a minimum confidence [48] defined by the user. Moreover, the user can set constraints on the maximum amount of time for the duration $x$ and $y$ of a rule.

Figure 2 shows an example of episode rule extraction from a sequence of alarms using TASA. For this example, only timestamps from the 9th to the 20th s of that sequence are considered. Alarms are denoted as $A$, $B$, $C$, $D$, $E$, and $F$. We can see from Fig. 2 that $Occ(\langle A, B, C\rangle, S) = \{[9, 18], [15, 18]\}$. Thus, Support$(A, B \Rightarrow_{(x,y)} C = 2)$. If we consider the time durations $x$ and $y$ as $x = 3$ and $y = 5$ s, then $Occ(\langle A, B, C\rangle, S, 5) = \{[15, 18]\}$ and $Occ(\langle A, B\rangle, S, 3) = \{[9, 11], [15, 16]\}$. Accordingly, Confidence$(A, B \Rightarrow_{(3,5)} C) = 1/2$. Taking another episode rule $(C, E \Rightarrow_{(3,5)} F)$, Since $Occ(\langle C, E, F\rangle, S) = \{[10, 13], [10, 20], [18, 20]\}$, Support$(C, E \Rightarrow_{(x,y)} F) = 3$. Moreover, since $Occ(\langle C, E, F\rangle, S, 5) = \{[10, 13], [18, 20]\}$ and $Occ(\langle C, E\rangle, S, 3) = \{[10, 12], [18, 19]\}$, Confidence$(C, E \Rightarrow_{(3,5)} F) = 2/2 = 1$.

Besides finding episode rules, TASA can also discover interesting associations between alarm properties by applying an association rule mining algorithm [3, 63]. Let there be a set of property values $P$ to describe alarms. Furthermore, let $DE = \{D_1, D_2, \ldots D_l\}$ be the descriptions of $l$ alarm occurrences, where $D_h \in P (1 \leq h \leq l)$. An association rule has the form $X \Rightarrow Y$, where $X, Y \subseteq P$ and $X \cap Y = \emptyset$, and is interpreted as if an alarm occurrence has property values $X$, then it also has property values $Y$. For example, a rule $linkAlarm \Rightarrow BF$ may indicate that if an alarm of type $linkAlarm$ occurs, it is associated to a network device called $BF$. Two measures are used to select rules. The support of an association rule $X \Rightarrow Y$ is calculated as $sup(X \Rightarrow Y) = |\{D|X \cup Y \subseteq D \in DE\}|$, and the confidence of a rule $X \Rightarrow Y$ is calculated as $conf(X \Rightarrow Y) = sup(X \Rightarrow Y)/|\{D|X \subseteq D \in DE\}|$. An example of association rule extraction using TASA is shown in Fig. 3, where properties are denoted as $A$, $B$, $C$, and $D$. Contrarily to episode rules, association rules do not consider time.

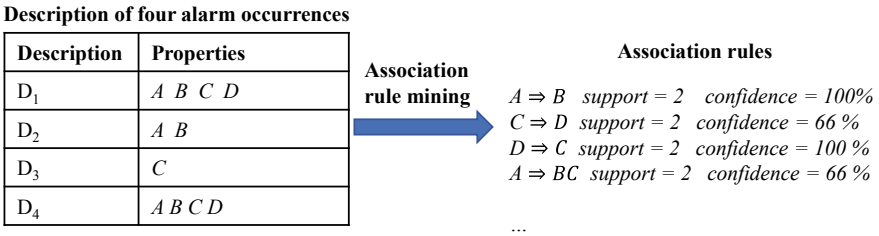


**Fig. 3** Example of discovering association rules between alarm properties with TASA

The TASA system also offers various tools to visualize rules, sort rules, and group rules having similar properties. A drawback of the TASA system is that it is not suitable for processing long alarm sequences.

## *3.2 Sequential Pattern Mining-Based Approaches*

Another pattern mining technique that has been applied for fault management is sequential pattern mining [28, 64].

Lozonavu et al. [56] proposed a method to discover alarm correlation patterns using sequential pattern mining [64]. This method not only discovers sequential patterns indicating interesting relationships between alarms instances, but also studies correlation and the relationships between different network elements such as network nodes and network problems [56]. The input is a sequence of alarms with their timestamps $\langle (a_1, t_1), (a_2, t_2), \ldots (a_n, t_n) \rangle$, as defined in the previous subsection. The approach of Lozonavu et al. first partitions the input sequence into several sequences based on the timestamps of alarms. Two consecutive alarms $(a_v, t_v)$ and $(a_{v+1}, t_{v+1})$ of the input sequence are put in the same sequence partition if $t_{v+1} - t_v < maxGap$, where $maxGap$ is a user-defined parameter. The result is a sequence database containing multiple input sequence partitions. In that sequence database, the timestamps of events are discarded. Then, the PrefixSpan [64] algorithm is applied on that database to find subsequences that appear in at least $minsup$ sequence partitions, where $minsup$ is a parameter set by the user. The result is a set of *sequential patterns* (frequent subsequences of alarms). A sequential pattern has the form $\langle b_1, b_2, b_3, \ldots, b_y \rangle$ indicating that some alarm $b_1$ appeared before another alarm $b_2$, was followed by $b_3$ and so on.

Then, the approach of Lozonavu et al. constructs a relationship graph, which shows the relationships between different alarms. This is done by transforming the discovered sequential patterns into relations. More precisely, the relationship graph is a directed weighted graph where each node represents a distinct alarm, a directed edge between two alarms indicates that an alarm occurred before the other in at least one discovered sequential pattern, and the weight of a relation reflects the strength

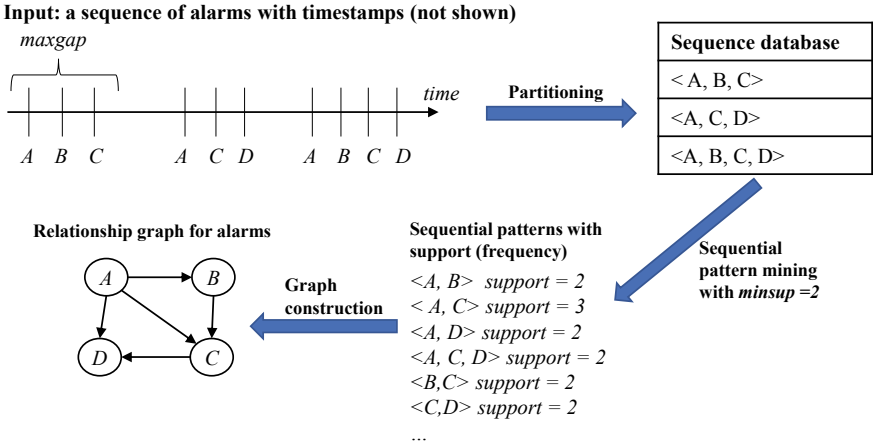


**Fig. 4** Example of applying Lozonavu et al.'s approach to study temporal relationships between alarms

of this relation which is calculated by the confidence measure. The confidence of a pattern $\langle b_1, b_2 \rangle$ is defined as how many sequence partitions contains the pattern $\langle b_1, b_2 \rangle$ divided by how many sequence partitions contain $b_1$. An example of using the approach of Lozonavu et al. is illustrated in Fig. 4, where alarms are denoted as $A$, $B$, $C$ and $D$.

Lozonavu et al. also presented a second type of relationship graph where the relationships between network devices are studied instead of the alarms. This graph is also derived from the sequential patterns because each alarm instance is associated with a device. It has been demonstrated that these graphs can then be used by network experts to better understand the network behavior and to discover hidden relations between network elements that were not known by network experts.

In another study, Wang et al. [74] have presented a system called Automatic Alarm Behavior Discovery (AABD) to study the behavior of alarms and select important alarms that should be brought to the attention of network operators from the thousands of alarms that may occur in a network. The AABD system takes as input a sequence of alarms, where each alarm is described using several fields such as the alarm type, the time that the alarm was produced and that it was cleared, the name of the device that has sent the alarm, and the network domain of the device.

AABD first preprocesses the input sequence to filter out invalid alarm instances. An alarm is said to be invalid if some of its fields contain invalid or missing values. For example, an alarm having no timestamp will be discarded as it will not be useful for the subsequent analysis performed by AABD. During the pre-processing step, alarm instances are also grouped by network domains as the behavior of an alarm may not be the same for devices of different domains.

Then, AABD applies an algorithm called Transient Flapping Determination (FTD) to detect alarms that are transient, i.e., alarms that usually only appear for a short time before they are cleared. A simple approach to determine if an alarm is transient is

to calculate the duration of each of its instances. Then, given two parameters $\Omega$ and $CT$, an alarm is called transient (flapping) if the proportion of its instances that have a duration no greater than $CT$ is greater than $\Omega$. Then, for each transient alarm, all alarm instances having a duration that is less than $CT$ are discarded to reduce the number of instances. It was shown that this approach based on transient alarm detection can reduce the number of alarms presented to operators by more than 84% [74]. However, a drawback of this definition is that setting the $CT$ parameter for different alarm types in different situations requires expert knowledge and is time-consuming. Thus, Wang et al. also proposed a method to automatically set these parameters for each alarm type based on the distribution of each alarm's duration [74]. The settings of the $CT$ parameter for an alarm type is called a *flapping rule*.

Then, AABD divides the sequence of alarms into several sequences to obtain a sequence database and applies the PrefixSpan [64] algorithm to extract frequent sequential patterns representing temporal correlations between alarm instances. This process is similar to the approach of Lozonavu et al. but the transformation from the input sequence to a sequence database is done differently. First, AABD calculates the support (occurrence frequency) of each alarm to find the $N$ most frequent alarms, called main alarms, where $N$ must be set by the user. Then, AABD creates a sequence for each main alarm. The created sequence contains that main alarm and all the alarms that occurred in the same time window width ($w$) with that main alarm. In other words, the created sequence will contain that main alarm and all the alarms that occurred no more than $w/2$ s before and $w/2$ s after. In the experiments, the value of the time window width $w$ was set to 5 min because it was empirically observed that 90% of related alarms occurred within 5 min. Sequential patterns are then extracted from the resulting sequence database. Note that each network domain is treated individually because the behavior of the same alarm type may not be the same for different domains.

Then, the AABD system creates rules called P-C Rules (Parent-Child rules) to reduce the number of alarms presented to users. A PC-rule indicates that an alarm (called *parent*) may cause several other alarms (called *child*). A PC rule generation is done using an algorithm named PCRG, which takes as input the sequential patterns and also a knowledge-base called the $P^2$ lookup table. This latter is created by network administrators for each network domain based on historical trouble tickets as well as based on their rich experience. The $P^2$ table specifies two possible relationships between alarm pairs, that is an alarm may cause another alarm or two alarms are mutually exclusive. In the $P^2$ lookup table, each potential parent alarm is represented by its name and a serial number which is an ordered list of integers that reveal the ranking property of this potential parent alarm. The PCRG algorithm utilizes the serial numbers to determine the relationship between alarms, to test the possibility of generating a PC rule from each discovered sequential pattern. It was found that PC rules can greatly reduce the number of alarms presented to users.

Costa et al. [19] proposed a complete alarm management system. The system's architecture has two main modules. The first one is a rule management system, which discovers rules in alarm data using a modified sequential pattern mining algorithm (GSP) [70], which reveals alarm correlation and can be used to perform root cause

analysis. Rules can be edited by hand. Moreover, a reinforcement learning algorithm is applied to evaluate how these rules are used for network malfunction resolution to refine the rule database. The second module performs alarm prioritization using a neural network to select alarms that should be treated with higher priority.

The system first pre-processes raw alarm data to keep only the relevant alarm attributes and sorts alarm instances by their start time. Moreover, if several alarm instances of the same type appear within a small amount of type, they are replaced by a single alarm with a counter so that patterns found only contain different alarms. The result is a sequence of alarms with timestamps. Then, sequential pattern extraction is performed by applying a modified GSP algorithm that extracts frequent sequential patterns using a sliding-window. From these patterns, rules are generated, which are evaluated using the lift and confidence measures to filter out spurious rules. To identify root cause problems, each extracted sequence is divided into two parts: The first part is the first alarm instance of the sequence and the second part is the following alarm instances with their occurrence counts. This method has some similarity to the approach of Lozonavu et al. [56] but it can be seen as more elaborated. The approach of Raúl is quite flexible as the sliding-window can be automatically enlarged in specific situations. The system was applied to data from a large Portugese telecommunication company and reduced the number of alarms presented to the user by up to 70%.

### *3.3 Clustering-Based Approaches*

Clustering algorithms are another type of unsupervised data mining techniques that have been used to perform alarm correlation and root cause analysis in telecommunication networks. Generally, the goal of clustering is to extract *clusters* (groups of instances) from a dataset, such that similar instances are grouped together while dissimilar instances are put in different clusters [40]. Instances are described using attributes and similarity between instances can be measured using various measures.

Clustering techniques, as other data mining techniques, have been studied for detecting and diagnosing faults in alarms data of telecommunication networks. In the context of alarm correlation analysis, clustering is used to automatically group alarms that occurred within a same short period of time and may have been triggered by the same cause. Clustering is interesting for fault management because it does not require training data and it can group alarms that are related into clusters based on various criteria such as their occurrence times. Then, experts can further analyze these clusters to discover the root cause of problems. For example, it has been suggested that the earliest alarm of a cluster may be considered as its root cause.

Sozuer et al. [69] proposed a method to discover clusters in alarm event data. Let $A$ be a set of alarm types. The proposed approach first transforms raw alarm event data into a sequence database, defined as a set of $n$ sequences $SDB = \{s_1, s_2, \ldots, s_n\}$. Each sequence $s_i$ $(1 \leq i \leq n)$ is an ordered list of alarm sets $\langle (A_1, t_1), (A_2, t_2), \ldots (A_m, t_m) \rangle$ where each alarm set $A_j$ $(1 \leq j \leq m)$ has a timestamp $t_j$ and $A_j \subseteq A$. Each alarm instance has a start time and clearance time, which defines a time interval called its life cycle. An alarm set is created by picking an

alarm instance and adding all other alarm instances that have a start time within its life cycle. Moreover, alarm instances within an alarm set are sorted by time. This approach allows capturing that an alarm may trigger other alarms until it is cleared. After the sequence database creation, two weight metrics are calculated for each item (alarm type) in the sequence database. These metrics are the term frequency and inverse item frequency, which are typically used for document classification. Then, item vectors are formed based on these weight vectors. Thereafter, the K means clustering algorithm is applied on the normalized item vectors to obtain a set of clusters where each cluster contains alarms that are highly related in view of the selected weight measure.

An experimental evaluation with two weeks of data from a radio network was performed to evaluate the approach. It was found that clusters could be used to perform prediction more accurately than using the RuleGrowth [32] sequential rule mining algorithm. A limitation of the approach of Sozuer et al. is that clustering is applied separately on each network node. Thus, generalizations for multiple nodes cannot be found.

In another study, Hashmi et al. [37] used various clustering and outlier detection techniques to analyze one year of network failure data collected from a national broadband network. Each failure was analyzed with respect to five attributes: fault occurrence date, time of the day, geographical region, fault cause (from 92 possible causes), and resolution time. Using clustering (k-means, fuzzy c-means or self-organizing maps), interesting spatio-temporal clusters of faults were found. For example, one cluster indicated that fault resolution typically takes a long time between 1 and 4 PM in a specific region for some fault types. Such insights can be useful to improve service in that area. Moreover, anomaly detection techniques were applied (either local outlier factor or local outlier probabilities) to identify abnormal data points. For example, some detected anomalies were faults occurring during the night [37]. In experiments, the sum of squared errors (SSE) and Davies-Bouldin index (DBI) values were used to evaluate the performance of different techniques. Note that the DBI is the average ratio of intra cluster variance and inter cluster distance of all clusters. Obtaining a low DBI value is desirable because it indicates that there is a high separation between clusters. The results indicated that the k-means clustering method outperformed fuzzy c-means clustering. In fact, the k-means clustering algorithm was able to create a larger separation between different clusters which improved the accuracy by obtaining clusters that are very different from each other. Moreover, experiments have shown that clustered self-organizing maps outperformed k-means and fuzzy c-means in terms of both SSE and DBI.

### *3.4 Summary and Perspective*

This section has reviewed some key approaches for fault management using pattern discovery algorithms. The reader can refer to Table 1 for a summary of these approaches. The main benefit of using a pattern mining approach is that patterns

**Table 1** Summary of reviewed pattern mining-based approaches

| Study | Category | Input | Output | Observation | Measure | Dataset |
|---|---|---|---|---|---|---|
| [38, 48] | Episode mining | Event sequence extracted from alarms data records | Episode rules, association rules | (1) Not suitable for long event sequences (2) The number of generated rules is large | Support, confidence | Several real-world datasets from telecommunication companies |
| [56] | Sequential pattern mining (PrefixSpan) | Alarm sequence database extracted using silent periods as delimiters | Relationship graph is extracted from frequent sequential patterns | The graph relationship is difficult to extract and to use by experts in case of large alarm datasets | Support, confidence | Dataset triggered in a 3G mobile network |
| [74] | Sequential pattern mining (PrefixSpan) | Alarm sequence database extracted using occurrence number of alarms with time window | Flapping rules, Parent–Child rules | a $P^2$ lookup table is required for each network domain which is not easy to create | Support | 6 alarms datasets with several network domains: 2G,3G,4G, CS and PS |
| [19] | Sequential pattern mining (GSP) | Alarm sequence database extracted using a sliding window | Association rules extracted from frequent sequential patterns | Reinforcement learning is performed to filter unimportant rules | Support, confidence, lift | Real-world dataset containing alarms from a Portuguese telecommunications company |
| [69] | Clustering (K-means) | Alarm sequence database extracted using the life cycle of alarms | Clusters of alarms | The clustering method is applied only on data of each node separately. | Term frequency (TF), inverse item frequency (ITF) | Alarms data extracted from a Nokia radio access network logs |
| [37] | Clustering (various) | Network failure log | Clusters of failures and anomalies | Can discover spatio-temporal clusters of failures and anomalies | Various | Data from a national broadband provider |

found can be easily understood by humans. There are many interesting possibilities for carrying further research on pattern mining for fault management. The next paragraphs discusses some of these possibilities.

First, it is observed that most of the above approaches are designed to handle rather simple data types (mostly discrete sequences) where alarms are viewed as events that have some attribute values. It would be interesting to consider more

complex data representation to extract richer patterns. For instance, none of the above studies consider the spatial dimension (the network topology) in the pattern mining process. An interesting possibility is to view the network as a dynamic graph where alarms are spreading along edges (communication links) between vertices (network devices) to find spatio-temporal patterns. To extract such patterns, graph-based pattern mining algorithms could be considered or extended. For example, there exists several algorithms for discovering patterns in dynamic attributed graphs. A dynamic attributed graph is a graph where vertices may be described using multiple attributes, and the graph evolves over time (edges and vertices may be added or deleted, and attribute values may change) [20, 25]. A network can be modeled as such a graph, where node attributes can be used to represent alarms. Then, various types of patterns could be discovered to reveal relationship between alarms such as (1) *Cohesive co-evolution pattern* [20] (a set of vertices that are similar and display the same trends for some attribute(s) during a time interval), (2) *Triggering patterns* [45] (a rule of the form $L \rightarrow R$ where $L$ is a sequence of attribute variations followed by a single topological change, $R$), and *significant trend sequences* [25] (correlated attribute variations for connected nodes).

Another possibility is to use other time representations for the input data as well as for patterns found. For example, while most of reviewed studies consider a strict sequential ordering between events in patterns, some algorithms have been designed to extract partial orders (patterns where events are partially ordered) [23, 65]. It is possible to consider richer relationships between events by explicitly representing each event as a time interval [18].

Another possibility is to consider extensions of traditional frequent pattern mining algorithms. For example, some extensions were proposed to handle items (events) having weights indicating their relative importance [53], weights and quantities [29, 72, 73], and cost values [27], as well as to use fuzzy functions [54] and taxonomies of items [12, 30, 82]. Using such algorithms would allow to consider richer information.

Another research direction is to explore the use or development of appropriate techniques for visualizing alarm patterns. For example, Jentner and Keim presented a detailed survey of many visualization techniques for patterns [41].

An important issue that could be also studied is how to reduce the number of patterns presented to the user by selecting the most important patterns or summarizing these patterns by reducing redundancy. There are several studies on this direction in the field of pattern mining such as to mine concise summary of patterns such as closed patterns [35, 51], maximal patterns [31, 58], and generator patterns [26, 83].

Another research direction is to go beyond the classical measures to select patterns such as the *support* and the *confidence*. A well-known problem with the support measure is that frequent alarms are sometimes unimportant and a limitation of the *confidence* is that it is very sensitive to the frequency of a rule's consequent in a database. Thus, other measures could be considered such as the *lift* [11], and application-specific measures could be designed.

Lastly, it is possible to build upon the research on stream pattern mining to adapt the current approaches for real-time processing [15, 61].

# 4 Machine Learning-Based Approaches

Apart from pattern mining-based techniques, several studies have applied machine learning techniques for network fault management. The next subsections review studies that have used different types of machine learning techniques: (1) artificial neural networks, (2) decision tree learning, (3) Bayesian networks, (4) Support-Vector Machines, (5) dependency graphs, and (6) other approaches.

## *4.1 Artificial Neural Networks-Based Approaches*

*Artificial Neural networks* (ANN) are one of the most popular types of machine learning models [52, 55]. Their structure and learning mechanisms are loosely inspired by the human brain. Over the years, multiple ANN architectures have been proposed for different needs. A traditional feed-forward ANN contains multiple layers of neurons that are interconnected, where some neurons receive some numeric values as input, while others output numeric values. An ANN is typically trained in a supervised manner using some examples of inputs and corresponding desired outputs. After training, a neural network can predict output values for new input values. Some key properties of artificial neural networks is that they can approximate various nonlinear functions (mapping inputs to outputs), and that they are resilient to noise [21, 71]. Because of these properties, artificial neural networks are suitable for fault management.

Wietgrefe et al. [77] presented an artificial neural network-based alarm correlation system for correlating alarms in a GSM network. The proposed system is called Cascade Correlation Alarm Correlator (CCAC). It relies on an ANN to find the causes of alarms. Each input neuron of the ANN represents an alarm type and takes a binary value (the alarm is active or inactive), while each output layer neuron correspond to a problem's cause. The neural network is trained using sets of alarms with their known causes. During the training, weights of an hidden layer of neurons are adjusted. Then, after training, the neural network can be fed with alarm data to obtain the likely alarm causes. It was demonstrated that CCAC works well even in the presence of noisy data, where noise is defined as some missing alarms or some additional irrelevant alarms. In a comparative study done by Wietgrefe [76], it was found that the trained ANN is more accurate at finding alarm causes than several other approaches such as case-based reasoning and rule-based reasoning approaches. A limitation of this study is that it does not consider the temporal relationships between alarms, and how devices are interconnected.

To take the time ordering of alarms into account, Marilly et al. [60] proposed an hybrid approach for fault management that combines a multi-layer feed-forward ANN with signal-processing techniques. This method receives as input an alarm log and first removes redundant alarms. Then, each alarm is feed to the input layer of the ANN. Each input neuron represents either an alarm type or a logical network entity. These inputs then pass through two hidden layers to produce an output (an alarm class). After repeating this process for each alarm, a sequence of alarm classes

is obtained ordered by time, which forms a signal. Then, the signal is treated using signal-processing methods such as a time-frequency visualization to extract pertinent information that could help identifying the cause of the malfunction (e.g., peaks of alarms that could indicate an equipment breakdown). This approach is robust to noise but still requires that an expert intervenes to identify malfunction in signals.

In another study, Arhouma and Amaitik [5] have compared different types of ANN models for alarm correlation in the ALMADAR GSM network by changing parameters of a neural network such as the network type, number of hidden neurons, and the learning algorithm. As in the work of Wietgrefe et al. [77], input neurons represent alarms while output neurons represent an initial cause of failure. It was found that the cascade-forward network type with Levenberg–Marquardt back-propagation training function gave the best diagnoses.

Differently, from the above studies, Barreto et al. [7] designed a fault detection approach to monitor the condition of a cellular network and detect abnormal behaviors. In this approach, the current state of a network is described as a vector of KPI (key performance indicator) values. Then, an ANN is trained with normal and abnormal states to then be able to classify a novel network state as abnormal or not. In that study, various neural network types were compared such as Self-Organizing Maps (SOM) and Neural Gas using a simulator.

In summary, although neural network-based approaches have the ability to extract patterns from complex and noisy data, these methods sometimes have a long training time and have difficulty to predict correctly for data that is largely different from the training data [21, 71]. Another issue with neural network-based approaches is that it is sometimes difficult to understand their inner-working.

## *4.2 Decision Tree-Based Approaches*

Another machine learning technique that has been used for fault management is *decision tree learning*. It is a supervised learning method, which requires to provide a set of training instances described using some attributes and where an attribute is selected as the target attribute to be predicted. Each possible value for the target attribute is called a *class*. From the training data, a decision tree is built using a learning algorithm. Then, the model (tree) can be used to classify (predict the class) new instances (data records).

A *decision tree* is a tree-like structure that is designed to support the classification of data instances based on their attribute values. In a decision tree, each leaf represents a decision (a class), while internal nodes represent a test of an attribute and each outgoing branch from an internal node represents a possible value of this attribute. An instance can be classified using a decision tree by traversing it from the root node and following the branches based on the instance's attribute values until a leaf node is reached [14].

Chen et al. [17] proposed a decision tree based method for failure diagnosis in large Internet systems. A decision tree was trained to classify the failed and successful

requests occurring during faulty periods. In the tree, the leaf nodes indicate whether a request is successful or has failed while internal nodes represent different system features such as a Machine's name and the name of a software program running on that machine. To identify causes of failures, post-processing is performed on paths of the generated decision tree by ranking them based on their correlation with failures. At this point, important features that correlate with the largest number of failures are selected.

Kiciman and Fox [47] have proposed a system for automatic fault detection in Internet services in which runtime paths of different requests served by the system are studied to identify the system's faulty components. A runtime path is a sequence of components, resources, and control-flow used to service a request. Besides, after detecting and recording the anomalous or successful requests with their corresponding paths, a decision tree is used to identify the components that may cause the failures. More precisely, a decision tree classifies the different paths into anomalous or successful paths based on their properties. Then, from the decision tree, a set of rules are extracted and used to extract the software and hardware components that lead to the failures.

Apart from fault localization in computer networks, decision trees have also been used for fault localization in other domains such as speech recognition [14] and Enterprise software systems (ESS) [67]. A drawback of decision tree based approaches for fault localization is that they may suffer from a degraded accuracy when dealing with noisy data [21, 71].

### *4.3 Bayesian Networks-Based Approaches*

Bayesian networks are a type of Probabilistic Graphical Models that can be used to build models from data or from expert opinions. Bayesian networks are directed acyclic graphs (DAG) in which nodes correspond to random variables over a multi-valued domain while edges represent the causal relationship between nodes which is measured by the conditional probability [21]. Bayesian networks can be used to deal with a wide range of tasks such as prediction, decision under uncertainty, and diagnosis [13].

Barco et al. [6] proposed a model based on discrete Bayesian networks, namely smooth Bayesian network, for automatic fault identification in cellular networks. The main purpose of this model is to improve the fault identification process by decreasing the sensitivity of diagnosis accuracy which happens principally due to the imprecision of the model's parameters. In other words, the model was designed to improve the diagnosis accuracy by overcoming the inaccuracy of a model's parameters. The proposed approach considers alarms and key performance indicators registered daily by the network management system for fault identification. Experimental results on data from GSM/GPRS networks have shown that the proposed smooth Bayesian network outperforms traditional Bayesian networks in case of inaccuracy of the model's parameters.

In another study, Ruiz et al. [68] used a Bayesian Network to identify the causes of network failures at the optical layer, and give them probabilities. The input of the Bayesian network is monitoring data represented as two time series about bit error rates and received power. The data was discretized to train the Bayesian network. Then, it can predict two types of failures, namely inter-channel interference and tight filtering. The approach was found to provide high accuracy in a simulated environment.

Khanafer et al. [46] developed a Bayesian network-based fault isolation approach for Universal Mobile Telecommunications System (UMTS) networks. Given some symptoms (KPIs and alarms), the system can predict the cause. Because data about symptoms is continuous, Khanafer et al. first discretized the data (using two methods, namely percentile-based dicretization and entropy minimization discretization). This allowed to automatically find thresholds for symptoms that may indicate faults. Then, a Naive Bayes network was built in which the conditional probabilities linking causes to symptoms were learnt from training data based on the thresholds. Experiments on data from a real UMTS network have shown that the proposed approach identified the correct cause of problems 88% of the time.

To handle time, Ding et al. [42] proposed an approach for fault management in IP networks, which relies on a dynamic Bayesian network. In that network, dependencies between network objects (e.g., devices, software processes) are modeled, and how they evolve over time with conditional probabilities. The dynamic Bayesian network can be used to predict the state of a network object and the evolution of dependencies between two objects. Moreover, it can also be used to infer the likely causes of some observed symptoms using backward inference. This study focused on faults caused by *soft changes* (changes that gradually occur over time) in a network, and their causes. The approach was only tested with simulated data.

Besides fault isolation, Bayesian networks were also used to predict faults in communication networks [2, 39, 49] to perform pro-active maintenance.

The above studies have shown promising results but Bayesian networks have some limitations. One of them is that several Bayesian models rely on some assumptions of independence between some events. Another is that a considerable amount of training data is required to correctly estimate conditional probabilities. A third one is that complex temporal relationships between alarms and interactions between devices are difficult to model using Bayesian Networks.

## *4.4 Support-Vector Machine-Based Approaches*

Another popular machine learning technique that is used for fault management is support-vector machine (SVM). It is a supervised technique that is generally used for classification or regression analysis. SVM is a linear classifier which is based on the margin maximization principle [1].

Given labeled data as input, the main idea of SVM is to find an optimal separating hyperplane that divides the input data into two classes. Note that to deal with non-

linear problems, the data can be mapped to higher dimensions to find a separating hyperplane. This mapping is performed using kernel methods and this process is called nonlinear SVM [4].

Wang et al. [75] have combined SVM with double-exponential smoothing (DES) to predict optical network equipment failures. To perform this task, some selected indicators are used as features to predict equipment failures. If an indicator is related to equipment failure, a change of this indicator's value will directly affect the equipment's state. Besides, since the relation between different indicators and equipment failures is not linear, a kernel function with punishment vector needs to be selected. This selection is performed by trying multiple kernel functions with different punishment factor values on SVM, using tenfold cross-validation to calculate their accuracy. Then, the combination that yield the highest accuracy is selected. Before applying SVM, DES is used to predict the value of each indicator at time $(t + T)$ taking the historical data from time $(t - n)$ to $(t - 1)$. At this point, the SVM method is applied to predict equipment failures at time $(t + T)$. Note that $(t - n)$ to $(t - 1)$ is a period of time that is selected in terms of days in the experiment and $(t + T)$ is the next period of time from the end of the previous time period to the end of the whole observation period. It was found that this improved SVM method can achieve an average of 95% accuracy (it can predict 95% of equipment failures).

In another study, Yuan et al. [85] proposed a system to automatically identify the root causes of problems in computer systems based on low-level traces of their behavior. This approach is different from that of utilizing a text-based search to find solutions to problems, which has been used in other systems. The proposed system has two main components: the tracer and the classifier.

The tracer collects the list of events that occur in the system when a problem's symptom is reproduced. Besides, the tracer records most system calls, which have several attributes such as Sequence number, Process ID, and Thread ID. After collecting all system call sequences related to symptoms, the system extracts *n-grams* from them. An n-gram can be viewed as sequential pattern where events must be consecutive. Then, the system encodes each log sequence as a bit vector where each dimension indicates the presence or absence of an n-gram.

At this point, the SVM classifier is applied on the set of bit vectors generated in the previous step for predicting the root cause of new traces from the previous registered traces with their known root causes. Besides, to prevent over-fitting due to the limited data, k-fold cross-validation is applied. It divides the training data into k partitions and then repeat selecting one partition to test it with the classifier trained with the remaining data until all data has been used for testing. The proposed approach was evaluated using four case examples containing diverse root causes. A prediction accuracy of nearly 90% was obtained.

Zidi et al. studied fault management for Wireless Sensor Networks (WSNs). A WSN is a set of autonomous devices that collaborate together through a wireless channel. WSNs are used to collect, process, and send data in various situations. However, WSNs may suffer from numerous failures [88]. Hence, Zidi et al. [88] proposed a new SVM-based technique for failure detection in WSNs. The proposed approach has two phases: The first phase is performed on anticipated time; whereas,

the second one is performed on real time. The first phase is the learning phase where the main objective is to obtain a decision function from learning data using SVM. Besides, data is composed of a set of normal data as well as a set of faulty data. Then, the decision function is further used in the second phase to detect in real time whether a new observation is normal or belongs to faulty cases. To evaluate the performance of the proposed method, 21 datasets were formed from a previously published database. The collected datasets are composed of a set of sensor measurements where different types of faults are injected with certain degrees. The experimental results show that the proposed method achieves high detection rate (99% in most cases).

## *4.5 Dependency Graph-Based Approaches*

These approaches are based on the construction of a dependency graph to represent different network elements. In [44], a graph-based method is developed for the fault localization problem. Besides, a dependency graph is constructed for the different network objects. Each vertex in the graph dependency is assigned with a weight which represents the probability that this object fails (triggers alarms) independently of other dependent network elements. On the other hand, each directed edge between two vertices is assigned with a weight that represents the strength of dependency between these vertices. In other words, the assigned weight is the conditional probability that the failure of one object is due to the failure of the other object. These weights can be estimated from the system specification information or from the history of previous failures [44].

After the graph construction, the system finds the domain of each alarm in the system which is defined as the set of objects that can cause this alarm. Note that, the problem of finding alarm domains is formulated as a variant of single source problem. At this point, a set of localization algorithms are used to discover alarm correlation patterns and identify fault locations.

Bouillard et al. [9, 10] have proposed another graph dependency based approach to correlate alarms in a network. This method is based on the assumption that frequently occurring alarms just refer to general information about the system, while rare alarms are viewed as more important since they may reveal a critical problem in the system. Accordingly, this method focuses on observing rare alarms.

First, this approach calculates the most frequent alarms. Then, based on these alarms, the alarm sequence is cut into set of small patterns (set-patterns) where frequent alarms are used as separators. Then, set-patterns are reduced using some transformation rules to facilitate their analysis.

At this point, the dependency graph from the reduced sets pattern is constructed and is divided into a set of subgraphs where each subgraph focuses on one rare alarm or a set of rare alarms. Finally, these subgraphs are further analyzed by network experts to discover the root cause of these alarms.

### 4.6 Other Approaches

Several of the reviewed approaches are passive. Johnsson and Meirosu [43] proposed an active fault management approach to perform fault localization in a packet-switched network. In that study, a network is viewed as an undirected graph connecting devices, and a fault is a performance degradation such as a large packet loss rate or transmission delay. Periodically, packets are sent between pairs of device to evaluate the network performance. They collected information about delays, jitters, and errors for the different edges of this graph and then used to calculate the probability that an edge is the source of a fault. The proposed approach utilizes probabilistic inference with a discrete state-space particle filter (also called histogram filter) to calculate the most probable location of a fault. This approach is lightweight and is applied in real time. The approach was evaluated using a simulator.

### 4.7 Summary and Perspective

This section has reviewed several studies, which have applied machine learning for fault management. Table 2 provides a summary of the main reviewed approaches discussed in this section. Most of the approaches surveyed in this section can be viewed as supervised approaches (requiring training data).

There are many possibilities for future work about using machine learning techniques. First, only a handful of machine learning techniques have been used for fault management, and there has been many advances in this field in recent years. Thus, newer techniques may be considered such as deep learning models [52], which may provide better results.

Second, it would be interesting to explore using models that consider richer information. For example, some models designed to handle temporal information could be used such as LSTM (Long Short Term Memory) [36]. Finding a way of also considering the network topology could lead to interesting results.

Third, the use of larger datasets with more features could be helpful to train better models. Semi-supervised machine learning techniques could also be used to reduce the need for human intervention. Synthetic but realistic data could also be generated to increase the size of the training data.

Fourth, an interesting alternative to passive monitoring (where a network management system waits passively for alarms sent by network nodes) is active monitoring (where the management system probes each nodes to verify its state). The advantage of active monitoring is that it may help recovering from faults more quickly. However, designing active monitoring techniques brings more challenges for network management [21].

Fifth, we have noticed that there are some recent studies on the use of machine learning in general and neural networks in particular for LTE and 5G networks [16, 22]. However, to the best of our knowledge, few studies have applied neural

**Table 2** Summary of machine learning-based approaches

| Study | Category | Input | Output | Observation | Algorithm | Dataset |
|---|---|---|---|---|---|---|
| [77] | Neural network | The input layer is a set of binary alarm vectors where each alarm is represented by a neuron | Each neuron of the output layer represents an initial cause of failure | The algorithm was found to tolerate well noisy data but only a few alarm vectors were used | Cascade Correlation Algorithm | Subset of 94 alarms chosen from real network |
| [60] | Neural network | Each input neuron represents a network's logical entity or alarm type from alarm log | An output neuron represents an alarm class | The neural network's output classes must be predefined by a network operator (not an easy task) | Multi-layer feed-forward ANN | Small set of alarm logs of an SDH network |
| [5] | Neural network | Each input neuron represents an alarm | Each output neuron represents an initial cause of failure | The neural network is applied separately to each network subarea. Small training set (55 patterns) | Cascade-forward ANN | Subparts of real GSM mobile network |
| [7] | Neural network | A vector of KPIs representing a network state | The state is abnormal or not | | SOM, Net-Gas and others | Data from simulator |
| [17] | Decision tree | A set of vectors corresponding to request paths with their associated features | A set of paths extracted from a decision tree | Leaf nodes that contain less failure requests will be ignored in the diagnosis process | C4.5 | Months of data from eBay's production website |
| [47] | Decision tree | A set of runtime paths with their associated features | A set of rules extracted from a decision tree's paths | The decision tree is a part of a pinpoint approach that is applied without requiring any apriori information from experts | ID3 | A testbed is deployed with three Internet services that include a wide variety of failures |
| [6] | Bayesian network | Causes and symptoms (KPIs, alarms) | The conditional probabilities of fault causes for the observed symptoms | In experiments, the diagnostic model does not take alarms into consideration | Naive Bayes model | 3 months of data from a GSM/GPRS network |
| [68] | Bayesian network | Time series about received power and error rates | One of two cause of failure | The model has good accuracy but consider only two causes | Bayesian network | Simulated data |
| [46] | Bayesian network | KPIs and alarms | A cause of failure | Dicretization methods were employed to find thresholds for continuous variables | Naive Bayes model | UMTS network data |

(continued)

**Table 2** (continued)

| Study | Category | Input | Output | Observation | Algorithm | Dataset |
|---|---|---|---|---|---|---|
| [75] | SVM | A set of vectors, each vector is composed of set of indicator values | Predict if the equipment will fail | Tenfold cross-validation is used twice for selecting relevant indicators to failures as well as for selecting the best failure diagnostic model | double-exponential smoothing with Support-Vector Machine | 44 d of real data collected in a WDM network from a telecommunications operator |
| [85] | SVM | System call sequences | Predict the class failure of new traces | To perform a prediction, it is necessary to reproduce the problem's symptoms, which is not always convenient | Linear SVM classifier | Collect event traces of four computer problems occurring in the Windows XP SP2 system |
| [88] | SVM | A set of observation vectors where each vector has 12 dimensions. Each dimension represent a sensor measurement and these sensor measurements are taken in 3 consecutive instances | Detection of faulty data | The faulty data are injected by authors with different rates of faults and different types of faults. This method allows to detect faults in real time. However, it cannot predict faults before its occurrence to prevent it | Non-linear SVM classifier | A set of 21 labeled WSNs datasets, each dataset consists of a set of sensor measurements where different type of faults are injected |
| [9] | Dependency graph | An event sequence divided into small pattern-sets using frequent alarms as delimiters | The dependency graph is divided into small subgraphs. Conditional probabilities are used to find root causes | This method is based on the assumption that rare alarms can give information about failure sources | Dependency graph | 1 year alarm log from two element networks of the Alcatel-Lucent operator |
| [9] | Dependency graph | An event sequence divided into small pattern-sets using frequent alarms as delimiters | The dependency graph is divided into small subgraphs. Conditional probabilities are used to find root causes | This method is based on the assumption that rare alarms can give information about failure sources | Dependency graph | 1 year alarm log from two element networks of the Alcatel-Lucent operator |
| [44] | Dependency graph | Network elements are modeled as a dependency graph | Discover the number of fault hypotheses and assign a confidence measure to each one | This method was not tested on a real network | Dependency graph | Simulated network with 50 nodes, where each node has a failure probability |
| [43] | Other | KPIs about packet transmission on different links | The network link that is likely faulty | A fault is a performance degradation. Considers the network topology. Can be applied in real-time | Probabilistic inference | Simulated data |

network in LTE and 5G networks for fault management [24, 62]. Thus, exploiting recent advances in machine learning in general and neural network in particular for fault management in 5G and LTE networks is a promising direction for future work.

Another research direction is to develop hybrid machine-learning based systems for fault management, which is to combine different machine learning techniques in a complementary way to efficiently perform fault management, while overcoming the disadvantages of using only one machine learning technique [13].

Moreover, with large and complex networks, both time and complexity become very large which makes the computation intractable [21]. As a result, developing techniques to reduce runtime and memory requirements of models while preserving their accuracy is crucial.

Finally, it is worth noticing that fault management in computer networks is a very active research topic. The reason is that, even with the development of many techniques based on pattern mining and machine learning, novel networking technologies and applications raise new challenges. Hence, existing systems become insufficient and must be improved [78].

## 5 Conclusion

This chapter has presented a survey of the main studies on using data mining and machine learnin- based techniques for network fault management, including a description of their characteristics, similarities, differences, and shortcomings. This is an active research area with many research opportunities.

## References

1. Adankon, M.M., Cheriet, M., et al.: Support vector machine. In: Encyclopedia of Biometrics. Springer, Boston, MA (2009)
2. Agbinya, J.I., Omlin, C.W., Kogeda, O.P.: A probabilistic approach to faults prediction in cellular networks. In: International Conference on Mobile Communications and Learning Technologies, Conference on Networking, Conference on Systems, p. 130 (2006)
3. Agrawal, R., Srikant, R., et al.: Fast algorithms for mining association rules. In: Proceedings of the 20th International Conference on Very Large Data Bases, VLDB, vol. 1215, pp. 487–499 (1994)
4. Amirabadi, M.: A survey on machine learning for optical communication (machine learning view). arXiv preprint arXiv:1909.05148 (2019)
5. Arhouma, A.K., Amaitik, S.M.: Decision support system for alarm correlation in GSM networks based on artificial neural networks. In: Conference Papers in Science, vol. 2013, Hindawi (2013)
6. Barco, R., Díez, L., Wille, V., Lázaro, P.: Automatic diagnosis of mobile communication networks under imprecise parameters. Exp. Syst. Appl. **36**(1), 489–500 (2009)
7. Barreto, G.A., Mota, J.C.M., Souza, L.G.M., Frota, R.A., Aguayo, L.: Condition monitoring of 3G cellular networks through competitive neural models. IEEE Trans. Neural Netw. **16**(5), 1064–1075 (2005)

8. Botta, A., de Donato, W., Persico, V., Pescapè, A.: Integration of cloud computing and internet of things: a survey. Fut. Gen. Comp. Syst. **56**, 684–700 (2016)
9. Bouillard, A., Junier, A., Ronot, B.: Alarms correlation in telecommunication networks. Research Report RR-8321, INRIA (2013). https://hal.inria.fr/hal-00838969
10. Bouillard, A., Junier, A., Ronot, B.: Impact of rare alarms on event correlation. In: Proceedings of the 9th International Conference on Network and Service Management (CNSM 2013), pp. 126–129. IEEE (2013)
11. Brin, S., Motwani, R., Ullman, J.D., Tsur, S.: Dynamic itemset counting and implication rules for market basket data. ACM Sigmod Record **26**(2), 255–264 (1997)
12. Cagliero, L., Chiusano, S., Garza, P., Ricupero, G.: Discovering high-utility itemsets at multiple abstraction levels. In: Proceedings of 21st European Conference on Advances in Databases and Information Systems, pp. 224–234 (2017)
13. Cai, B., Huang, L., Xie, M.: Bayesian networks in fault diagnosis. IEEE Trans. Indus. Inform. **13**(5), 2227–2240 (2017)
14. Cerňak, M.: A comparison of decision tree classifiers for automatic diagnosis of speech recognition errors. Comput. Inform. **29**(3), 489–501 (2012)
15. Chen, M., Zheng, A.X., Lloyd, J., Jordan, M.I., Brewer, E.: Failure diagnosis using decision trees. In: International Conference on Autonomic Computing, Proceedings, pp. 36–43. IEEE (2004)
16. Chen, Y.C., Peng, W.C., Lee, S.Y.: Mining temporal patterns in time interval-based data. IEEE Trans. Knowl. Data Eng. **27**(12), 3318–3331 (2015)
17. Chen, C.C., Shuai, H.H., Chen, M.S.: Distributed and scalable sequential pattern mining through stream processing. Knowl. Inform. Syst. **53**(2), 365–390 (2017)
18. Chen, M., Challita, U., Saad, W., Yin, C., Debbah, M.: Artificial neural networks-based machine learning for wireless networks: a tutorial. IEEE Commun. Surv. Tutor. **21**(4), 3039–3071 (2019)
19. Costa, R., Cachulo, N., Cortez, P.: An intelligent alarm management system for large-scale telecommunication companies. In: Portuguese Conference on Artificial Intelligence, pp. 386–399. Springer (2009)
20. Desmier, E., Plantevit, M., Robardet, C., Boulicaut, J.F.: Cohesive co-evolution patterns in dynamic attributed graphs. In: International Conference on Discovery Science, pp. 110–124. Springer (2012)
21. Ding, J., Kramer, B., Xu, S., Chen, H., Bai, Y.: Predictive fault management in the dynamic environment of ip networks. In: 2004 IEEE International Workshop on IP Operations and Management, pp. 233–239 (2004)
22. Dusia, A., Sethi, A.S.: Recent advances in fault localization in computer networks. IEEE Commun. Surv. Tutor. **18**(4), 3030–3051 (2016)
23. Eugenio, M., Cayamcela, M., Lim, W.: Artificial intelligence in 5G technology: a survey. In: 2018 International Conference on Information and Communication Technology Convergence (ICTC) (2018)
24. Fabrègue, M., Braud, A., Bringay, S., Le Ber, F., Teisseire, M.: Mining closed partially ordered patterns, a new optimized algorithm. Knowl.-Based Syst. **79**, 68–79 (2015)
25. Feng, W., Teng, Y., Man, Y., Song, M.: Cell outage detection based on improved BP neural network in LTE system (2015)
26. Fournier-Viger, P., Cheng, C., Cheng, Z., Lin, J.C.W., Selmaoui-Folcher, N.: Mining significant trend sequences in dynamic attributed graphs. Knowl.-Based Syst. **182** (2019)
27. Fournier-Viger, P., Gomariz, A., Šebek, M., Hlosta, M.: VGEN: fast vertical mining of sequential generator patterns. In: International Conference on Data Warehousing and Knowledge Discovery, pp. 476–488. Springer (2014)
28. Fournier-Viger, P., Li, J., Lin, J.C.W., Chi, T.T., Kiran, R.U.: Mining cost-effective patterns in event logs. Knowl.-Based Syst. **191**, 105241 (2020)
29. Fournier-Viger, P., Lin, J.C.W., Truong-Chi, T., Nkambou, R.: A survey of high utility itemset mining. In: High-Utility Pattern Mining, pp. 1–45. Springer (2019)
30. Fournier-Viger, P., Wang, Y., Chun-Wei, J., Luna, J.M., Ventura, S.: Mining cross-level high utility itemsets. In: Proceedings of 33rd International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems. Springer (2020)

31. Fournier-Viger, P., Wu, C.W., Gomariz, A., Tseng, V.S.: VMSP: Efficient vertical mining of maximal sequential patterns. In: Canadian Conference on Artificial Intelligence, pp. 83–94. Springer (2014)
32. Fournier-Viger, P., Yang, P., Lin, J.C.W., Yun, U.: Hue-span: fast high utility episode mining. In: Proceedings of 14th International Conference on Advanced Data Mining and Applications, pp. 169–184. Springer (2019)
33. Fournier-Viger, P., Yang, Y., Yang, P., Lin, J.C.W., Yun, U.: TKE: Mining top-k frequent episodes. In: Proceedings of 33rd International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems. Springer (2020)
34. Fournier-Viger, P., Wu, C.W., Tseng, V.S., Cao, L., Nkambou, R.: Mining partially-ordered sequential rules common to multiple sequences. IEEE Trans. Knowl. Data Eng. **27**(8), 2203–2216 (2015)
35. Fournier-Viger, P., Lin, J.C.W., Kiran, R.U., Koh, Y.S., Thomas, R.: A survey of sequential pattern mining. Data Sci. Pattern Recogn. **1**(1), 54–77 (2017)
36. Fumarola, F., Lanotte, P.F., Ceci, M., Malerba, D.: Clofast: closed sequential pattern mining using sparse and vertical id-lists. Knowl. Inform. Syst. **48**(2), 429–463 (2016)
37. Gers, F.A., Schmidhuber, J., Cummins, F.: Learning to forget: continual prediction with LSTM. Neural Computation **12**(10) (2000)
38. Hashmi, U.S., Darbandi, A., Imran, A.: Enabling proactive self-healing by data mining network failure logs. In: 2017 International Conference on Computing, Networking and Communications (ICNC), pp. 511–517. IEEE (2017)
39. Hatonen, K., Klemettinen, M., Mannila, H., Ronkainen, P., Toivonen, H.: TASA: Telecommunication alarm sequence analyzer or how to enjoy faults in your network. In: Proceedings of NOMS'96-IEEE Network Operations and Management Symposium, vol. 2, pp. 520–529. IEEE (1996)
40. Hood, C.S., Ji, C.: Proactive network-fault detection (telecommunications). IEEE Trans. Reliab. **46**(3), 333–341 (1997)
41. Jain, A.K.: Data clustering: 50 years beyond k-means. Pattern Recogn. Lett. **31**(8), 651–666 (2010)
42. Jentner, W., Keim, D.A.: Visualization and visual analytic techniques for patterns. In: High-Utility Pattern Mining, pp. 303–337. Springer (2019)
43. Johnsson, A., Meirosu, C.: Towards automatic network fault localization in real time using probabilistic inference. In: 2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013), pp. 1393–1398. IEEE (2013)
44. Katzela, I., Schwartz, M.: Schemes for fault identification in communication networks. IEEE/ACM Trans. Network. **3**(6), 753–764 (1995)
45. Kaytoue, M., Pitarch, Y., Plantevit, M., Robardet, C.: Triggering patterns of topology changes in dynamic graphs. In: Proceedings of the 2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, pp. 158–165. IEEE (2014)
46. Khanafer, R.M., Solana, B., Triola, J., Barco, R., Moltsen, L., Altman, Z., Lazaro, P.: Automated diagnosis for UMTS networks using Bayesian network approach. IEEE Trans. vehic. Technol. **57**(4), 2451–2461 (2008)
47. Kiciman, E., Fox, A.: Detecting application-level failures in component-based internet services. IEEE Trans. Neural Netw. **16**(5), 1027–1041 (2005)
48. Klemettinen, M., Mannila, H., Toivonen, H.: Rule discovery in telecommunication alarm data. J. Netw. Syst. Manag. **7**(4), 395–423 (1999)
49. Kogeda, P., Agbinya, J.I.: Prediction of faults in cellular networks using Bayesian network model. In: International conference on Wireless Broadband and Ultra Wideband Communication. UTS ePress (2006)
50. Lamport, L.: Time, clocks, and the ordering of events in a distributed system. In: Concurrency: The Works of Leslie Lamport, pp. 179–196. ACM (2019)
51. Le, B., Duong, H., Truong, T., Fournier-Viger, P.: Fclosm, Fgensm: two efficient algorithms for mining frequent closed and generator sequences using the local pruning strategy. Knowl. Inform. Syst. **53**(1), 71–107 (2017)

52. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436–444 (2015)
53. Lee, G., Yun, U., Kim, D.: A weight-based approach: frequent graph pattern mining with length-decreasing support constraints using weighted smallest valid extension. Adv. Sci. Lett. **22**(9), 2480–2484 (2016)
54. łgorzata Steinder, M., Sethi, A.S.: A survey of fault localization techniques in computer networks. Sci. Comput. Program. **53**(2), 165–194 (2004)
55. Li, H., Wang, Y., Zhang, N., Zhang, Y.: Fuzzy maximal frequent itemset mining over quantitative databases. In: Asian Conference on Intelligent Information and Database Systems, pp. 476–486. Springer (2017)
56. Liu, W., Wang, Z., Liu, X., Zeng, N., Liu, Y., Alsaadi, F.E.: A survey of deep neural network architectures and their applications. Neurocomputing **234**, 11–26 (2017)
57. Lozonavu, M., Vlachou-Konchylaki, M., Huang, V.: Relation discovery of mobile network alarms with sequential pattern mining. In: 2017 International Conference on Computing, Networking and Communications (ICNC), pp. 363–367. IEEE (2017)
58. Luna, J.M., Fournier-Viger, P., Ventura, S.: Frequent itemset mining: a 25 years review. Wiley Interdisc. Rev.: Data Mining Knowl. Disc. **9**(6), e1329 (2019)
59. Luo, C., Chung, S.M.: Efficient mining of maximal sequential patterns using multiple samples. In: Proceedings of the 2005 SIAM International Conference on Data Mining, pp. 415–426. SIAM (2005)
60. Mannila, H., Toivonen, H., Verkamo, A.I.: Discovery of frequent episodes in event sequences. Data Mining Knowl. Disc. **1**(3), 259–289 (1997)
61. Marilly, E., Aghasaryan, A., Betge-Brezetz, S., Martinot, O., Delegue, G.: Alarm correlation for complex telecommunication networks using neural networks and signal processing. In: IEEE Workshop on IP Operations and Management, pp. 3–7. IEEE (2002)
62. Mendes, L.F., Ding, B., Han, J.: Stream sequential pattern mining with precise error bounds. In: 2008 Eighth IEEE International Conference on Data Mining, pp. 941–946. IEEE (2008)
63. Mismar, F.B., Evans, B.L.: Deep q-learning for self-organizing networks fault management and radio performance improvement. In: 2018 52nd Asilomar Conference on Signals, Systems, and Computers, pp. 1457–1461. IEEE (2018)
64. Nguyen, L.T., Vo, B., Nguyen, L.T., Fournier-Viger, P., Selamat, A.: Etarm: an efficient top-k association rule mining algorithm. App. Intell. **48**(5), 1148–1160 (2018)
65. Pei, J.: Mining sequential patterns efficiently by prefix-projected pattern growth. In: International Conference of Data Engineering (ICDE2001) (2001)
66. Pei, J., Wang, H., Liu, J., Wang, K., Wang, J., Yu, P.S.: Discovering frequent closed partial orders from strings. IEEE Trans. Knowl. Data Eng. **18**(11), 1467–1481 (2006)
67. Rashid, B., Rehmani, M.H.: Applications of wireless sensor networks for urban areas: a survey. J. Netw. Comput. Appl. **60**, 192–219 (2016)
68. Reidemeister, T., Munawar, M.A., Jiang, M., Ward, P.A.: Diagnosis of recurrent faults using log files. In: Proceedings of the 2009 Conference of the Center for Advanced Studies on Collaborative Research, pp. 12–23. IBM Corporation (2009)
69. Ruiz, M., Fresi, F., Vela, A.P., Meloni, G., Sambo, N., Cugini, F., Poti, L., Velasco, L., Castoldi, P.: Service-triggered failure identification/localization through monitoring of multiple parameters. In: Proceedings of 42nd European Conference on Optical Communication, pp. 1–3. VDE (2016)
70. Sozuer, S., Etemoglu, C., Zeydan, E.: A new approach for clustering alarm sequences in mobile operators. In: NOMS 2016-2016 IEEE/IFIP Network Operations and Management Symposium, pp. 1055–1060. IEEE (2016)
71. Srikant, R., Agrawal, R.: Mining sequential patterns: Generalizations and performance improvements. In: International Conference on Extending Database Technology, pp. 1–17. Springer (1996)
72. Truong, T., Duong, H., Le, B., Fournier-Viger, P.: Efficient vertical mining of high average-utility itemsets based on novel upper-bounds. IEEE Trans. Knowl. Data Eng. **31**(2), 301–314 (2018)

73. Truong-Chi, T., Fournier-Viger, P.: A survey of high utility sequential pattern mining. In: High-Utility Pattern Mining, pp. 97–129. Springer (2019)
74. Wang, J., He, C., Liu, Y., Tian, G., Peng, I., Xing, J., Ruan, X., Xie, H., Wang, F.L.: Efficient alarm behavior analytics for telecom networks. Inform. Sci. **402**, 1–14 (2017)
75. Wang, Z., Zhang, M., Wang, D., Song, C., Liu, M., Li, J., Lou, L., Liu, Z.: Failure prediction using machine learning and time series in optical network. Opt. Exp. **25**(16), 18553–18565 (2017)
76. Wietgrefe, H., Tuchs, K.D., Jobmann, K., Carls, G., Fröhlich, P., Nejdl, W., Steinfeld, S.: Using neural networks for alarm correlation in cellular phone networks. In: International Workshop on Applications of Neural Networks to Telecommunications (IWANNT), pp. 248–255. Citeseer (1997)
77. Wietgrefe, H.: Investigation and practical assessment of alarm correlation methods for the use in gsm access networks. In: NOMS 2002. IEEE/IFIP Network Operations and Management Symposium. Management Solutions for the New Communications World (Cat. No. 02CH37327), pp. 391–403. IEEE (2002)
78. Wong, W.E., Debroy, V.: A survey of software fault localization. Department of Computer Science, University of Texas at Dallas, Technical Report, UTDCS-45 **9** (2009)
79. Wu, X., Zhu, X., Wu, G., Ding, W.: Data mining with big data. IEEE Trans. Knowl. Data Eng. **26**, 97–107 (2014)
80. Xu, Y., Zeng, M., Liu, Q., Wang, X.: A genetic algorithm based multilevel association rules mining for big datasets. Math. Prob. Eng. (2014)
81. Xu, D., Tian, Y.: A comprehensive survey of clustering algorithms. Ann. Data Sci. **2**(2), 165–193 (2015)
82. Xu, L., He, W., Li, S.: Internet of things in industries: a survey. IEEE Trans. Indus. Inform. **10**, 2233–2243 (2014)
83. Yi, S., Zhao, T., Zhang, Y., Ma, S., Yin, J., Che, Z.: Seqgen: mining sequential generator patterns from sequence databases. Adv. Sci. Lett. **11**(1), 340–345 (2012)
84. Yu, C.B., Hu, J.J., Li, R., Deng, S.H., Yang, R.M.: Node fault diagnosis in WSN based on RS and SVM. In: Proceedings of 2014 International Conference on Wireless Communication and Sensor Network, pp. 153–156 (2014)
85. Yuan, C., Lao, N., Wen, J.R., Li, J., Zhang, Z., Wang, Y.M., Ma, W.Y.: Automated known problem diagnosis with event traces. ACM SIGOPS Oper. Syst. Rev. **40**(4), 375–388 (2006)
86. Zeadally, S., Hunt, R., Chen, Y.S., Irwin, A., Hassan, A.: Vehicular ad hoc networks (VANETS): status, results, and challenges. Telecommun. Syst. **50**, 217–241 (2012)
87. Zhang, Z., Mehmood, A., Shu, L., Huo, Z., Zhang, Y.L., Mukherjee, M.: A survey on fault diagnosis in wireless sensor networks. IEEE Access **6**, 11349–11364 (2018)
88. Zidi, S., Moulahi, T., Alaya, B.: Fault detection in wireless sensor networks through SVM classifier. IEEE Sens. J. **18**(1), 340–347 (2017)