Chapter 6
Machine Learning Algorithms for Network Intrusion Detection

Jie Li, Yanpeng Qu, Fei Chao, Hubert P. H. Shum, Edmond S. L. Ho and Longzhi Yang

Abstract Network intrusion is a growing threat with potentially severe impacts, which can be damaging in multiple ways to network infrastructures and digital/intellectual assets in the cyberspace. The approach most commonly employed to combat network intrusion is the development of attack detection systems via machine learning and data mining techniques. These systems can identify and disconnect malicious network traffic, thereby helping to protect networks. This chapter systematically reviews two groups of common intrusion detection systems using fuzzy logic and artificial neural networks, and evaluates them by utilizing the widely used KDD 99 benchmark dataset. Based on the findings, the key challenges and opportunities in addressing cyberattacks using artificial intelligence techniques are summarized and future work suggested.

6.1 Introduction

Cybersecurity can be assisted by a set of techniques that protect cyberspace and ensure the integrity, confidentiality, and availability of networks, applications, and data. Cybersecurity techniques also have the potential to defend against and recover from any type of attack. More devices, namely, Internet of Things (IoT) devices, are becoming connected to the cyberspace, and cybersecurity has become an elevated concern affecting governments, businesses, other organizations, and individuals. The scope of cybersecurity is broad, and can be grouped into five areas: critical infrastructure, network security, cloud security, application security, and IoT security. Network

J. Li · H. P. H. Shum · E. S. L. Ho · L. Yang (✉)
Northumbria University, Newcastle upon Tyne, UK
e-mail: longzhi.yang@northumbria.ac.uk

Y. Qu
Dalian Maritime University, Dalian, People's Republic of China
e-mail: yanpengqu@dlmu.edu.cn

F. Chao
Xiamen University, Xiamen, People's Republic of China
e-mail: fchao@xmu.edu.cn

L. F. Sikos (ed.), *AI in Cybersecurity*, Intelligent Systems Reference Library 151,
https://doi.org/10.1007/978-3-319-98842-9_6

462756_1_En_6_Chapter ☑ TYPESET ☐ DISK ☐ LE ☑ CP Disp.:23/7/2018 Pages: 30 Layout: T1-Standard

security is an important challenge in the field of cybersecurity, because networks provide the means for the crucial access to others devices, and for connectivity between all the assets in cyberspace. Severe network attacks can lead to system damage, network paralysis, and data loss or leakage. Network intrusion detection systems (NIDS) attempt to identify unauthorized, illicit, and anomalous behavior based solely on network traffic to support decision making in network preventative actions by network administrators.

Traditional network intrusion detection systems are mainly developed using available knowledge bases, which are comprised of the specific patterns or strings that correspond to already known network behaviors, i.e., normal traffic and abnormal traffic [1]. Those patterns are used to check monitored network traffic to recognize possible threats. Typically, the knowledge bases of such systems are defined based on expert knowledge, and the patterns must be updated to ensure the coverage of new threats [2]. Therefore, the detection performance of traditional network intrusion detection systems depends highly on the quality of the knowledge base. From a theoretical point of view, network intrusion detection systems mainly aim to classify the monitored traffic as either "legitimate" or "malicious." Therefore, machine learning approaches are appropriate to solve such problems; and they have recently been widely applied to help better manage network intrusion detection issues.

Machine learning (ML) is a field of artificial intelligence, which refers to a set of techniques that give computer systems the ability to "learn." Typically, machine learning algorithms, such as artificial neural networks, learn from data samples to categorize or find patterns in the data, and enable computer systems to make predictions on new or unseen data instances based on the discovered patterns [3]. Depending on the way of learning, machine learning can be further grouped into two main categories: supervised learning and unsupervised learning. Supervised learning discovers the patterns to map an input to an output based on the labeled input-output pairs of data samples [4]. The classification problem is a typical supervised learning problem, which has been commonly used for solving NIDS problems, such as those reported in [5–8]. The goal of unsupervised learning is to find a mapping that is able to describe a hidden structure from unlabeled data samples. It is a powerful tool for identifying structures when unlabeled data samples are given [4]. Thanks to the relaxation of the requirement for labels of training data in the unsupervised learning, various unsupervised learning approaches have also been widely applied for NIDS problems, such as the clustering-based NIDS [9] and self-organizing map based NIDS [10].

This chapter focuses primarily on network intrusion detection systems, and particularly how the machine learning and data mining techniques can help in developing network intrusion detection systems. The chapter firstly systematically reviews intrusion detection techniques from the perspective of both hardware deployment and software implementation. The two most commonly used NIDS development methods and the three most commonly used detection methodologies are reviewed first; these are followed by the investigation of applying machine learning and data mining techniques in the implementation of intrusion detection systems. Two representative machine learning approaches, including fuzzy inference systems and artificial

462756_1_En_6_Chapter ☑ TYPESET ☐ DISK ☐ LE ☑ CP Disp.:23/7/2018 Pages: 30 Layout: T1-Standard

neural networks, are of particular interest in this chapter, because they are the machine learning and data mining techniques most suitable for supporting intrusion detection systems. Traditionally, fuzzy inference systems are not classified as machine learning algorithms, however, the rule base generation mechanism follows the data mining principle; therefore, fuzzy inference systems with automatic rule base generation can also be considered as machine learning. Finally, the intrusion detection systems developed upon these machine learning approaches are evaluated using the widely used KDD 99 benchmark dataset.

The remainder of this chapter is organized as follows. Section 6.2 introduces the hardware deployment methods of network intrusion detection systems and detection methodologies. Section 6.3 reviews the existing machine learning-based network intrusion detection systems using fuzzy inference systems and artificial neural networks. The limitations and potential solutions of both techniques are also discussed in this section. Section 6.4 evaluates the studied systems using a well-known benchmark dataset KDD 99. Section 6.5 concludes the chapter and sets directions for future work.

6.2 Network Intrusion Detection Systems

Network intrusion detection systems are software-based or hardware-based devices that are used to monitor network traffic, i.e., to analyze them for signs of possible attacks or suspicious activities. There are usually one or more network traffic sensors used to monitor network activity on one or more network segments. The system constantly performs analysis and watches for certain patterns of passing traffic in a monitored network environment. If the detected traffic patterns match the defined signatures or policies in the knowledge base (e.g., based on a fuzzy rule base or a trained neural network), a security alert is generated.

6.2.1 Deployment Methods

There are multiple methods that can be adopted to deploy a NIDS in order to capture and monitor traffic in a network environment, with passive deployment and in-line deployment being the most commonly used, as shown in Fig. 6.1a and b.

In the passive deployment method, the NIDS device is connected to a network switch, which is deployed between the main firewall and the internal network. The switch is usually configured with a port mirroring technology, such as the Mirror Port supported by HP and the Switched Port Analyzer (SPAN) supported by Cisco. These port mirroring technologies are able to copy all network traffic, including incoming and outgoing traffic, to a particular interface of the NIDS for the purpose of traffic monitoring and analysis. This method usually requires a high-end network switch in order to enable the port mirroring technologies. There is a special case of

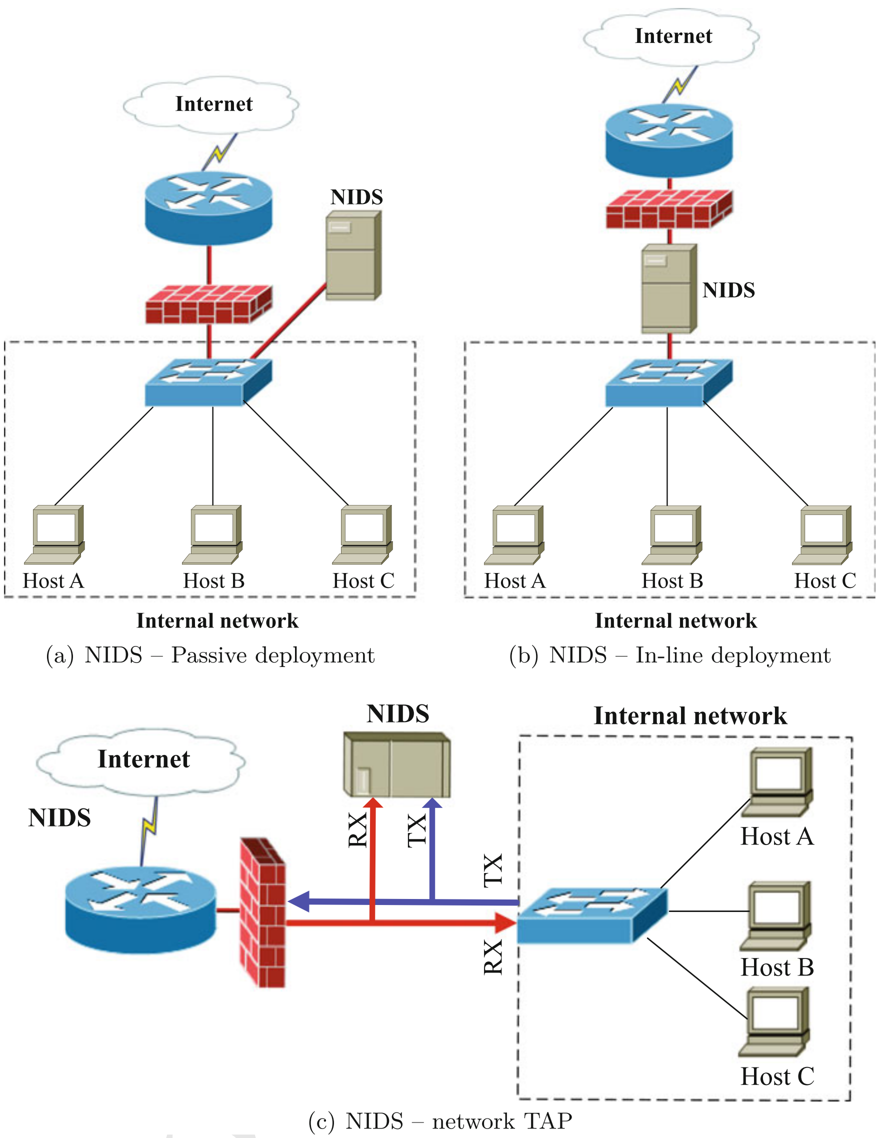


Fig. 6.1 Deployment methods for intrusion detection systems

passive deployment, which is the passive network TAP (Terminal Access Point) [11]. In particular, a network TAP uses pairs of cables included in the original Ethernet cable, as illustrated in Fig. 6.1c, to send a copy of the original network traffic to the NIDS.

The in-line deployment method deploys NIDS devices the same way as firewalls, which allows all traffic to pass directly through the NIDS. Therefore, this deployment method does not require any particularly high-end network device, which is an ideal solution for those environments in which port mirroring technologies are unavailable, such as a small branch office with low-end networking equipment.

It is important to note that the deployment methods should be carefully selected while taking into account the network topology for optimal performance. For instance, in the example shown in Fig. 6.1a, the port mirroring method is not only able to monitor the outgoing traffic between the internal network and the Internet, but also the internal traffic between Hosts A, B, and C. However, the network TAP and in-line deployment method are only able to monitor the outgoing traffic that is generated between the internal network and the Internet. Therefore, the NIDS, which is deployed by either the network TAP or the in-line method, will not notice if there is suspicious traffic between two client machines. In addition, because the port mirroring method uses a signal network interface to monitor the entire switch traffic, traffic congestion may occur if the switch backbone traffic is beyond the capacity of the bandwidth of the monitored port. Therefore, it is a good strategy to deploy multiple NIDSes in complex network environments so that these blind spots can be eliminated.

6.2.2 Detection Methodologies

Generally speaking, intrusion detection methodologies can be grouped into three major categories: signature-based detection, anomaly-based detection, and specification-based detection [12].

The signature-based NIDS, also called knowledge-based detection or misuse detection, refers to the detection of attacks or threats by looking for specific patterns or strings that correspond to already known attacks or threats. These specific patterns or strings are saved in a knowledge base, such as the byte sequences of the network traffic, known malicious instruction sequences exploited by malware, the specific ports a host tries to access, etc. Signature-based detection is a process that compares known patterns against monitored network traffic to recognize possible intrusions. Therefore, signature-based detection is able to effectively detect known threats in a network environment, and its knowledge bases are usually generated by experts. A good example for this type of detection is a large amount of failed login attempts that have been detected in a Telnet session.

Anomaly-based detection primarily focuses on normal traffic behaviors rather than specific attack behaviors, which overcomes the limitation of signature-based detection that is only able to detect known attacks. This method is usually comprised of two processes: a training process and a detection process. In the training phase, machine learning algorithms are usually adopted to develop a model of trustworthy activity based on the behavior of the network traffic without attacks. In the detection phase, the developed trustworthy activity model is compared to the currently monitored traffic behavior, and any deviations indicate a potential threat. The anomaly-based detection method is usually adopted to detect unknown attacks [13–18]. However, the effectiveness of anomaly-based detection is greatly affected by the selected features the machine learning algorithms use. Unfortunately, the selection of the appropriate set of features has proven to be a big challenge. Also, the observed system behaviors constantly change, which causes anomaly-based detection to produce a weak profile accuracy.

Specification-based detection is similar to the anomaly-based detection method as in it also detects attacks as deviations from normal behavior. However, specification-based approaches are based on manually developed specifications that characterize legitimate behaviors rather than relying on machine learning algorithms. Although this method is not characterized by the high rate of false alarms typical to anomaly-based detection methods, the development of detailed specifications can be time-consuming. Because it detects attacks as deviations from legitimate behaviors, specification-based approaches are commonly used for unknown attacks detection [19, 20]. In addition, multiple detection methodologies could be adopted jointly to provide more extensive and accurate detection [21].

6.3 Machine Learning in Network Intrusion Detection

Machine learning and data mining techniques work by establishing an explicit or implicit model that enables the analyzed patterns to be categorized. In general, machine learning techniques are able to deal with three common problems: classification, regression, and clustering. Network intrusion detection can be considered as a typical classification problem. Therefore, a labeled training dataset is usually required for system modeling. A number of machine learning approaches have been used to solve network intrusion detection problems, and all of them consist of three general phases (as illustrated in Fig. 6.2):

- *Preprocessing*: the data instances that are collected from the network environment are structured, which can then be directly fed into the machine learning algorithm. The processes of feature extraction and feature selection are also applied in this phase.
- *Training*: a machine learning algorithm is adopted to characterize the patterns of various types of data, and build a corresponding system model.
- *Detection*: once the system model is built, the monitored traffic data will be used as system input to be compared to the generated system model. If the pattern of the observation is matched with an existing threat, an alarm will be triggered.

Both supervised and unsupervised machine learning approaches have already been utilized to solve network intrusion detection problems. For instance, supervised learning-based classifiers have been successfully employed to detect unauthorized access, such as k-nearest neighbor (k-NN) [6], support vector machine (SVM) [22], decision tree [23], naïve Bayes network [7], random forests [5], and artificial neu-

462756_1_En_6_Chapter ☑ TYPESET ☐ DISK ☐ LE ☑ CP Disp.:23/7/2018 Pages: 30 Layout: T1-Standard

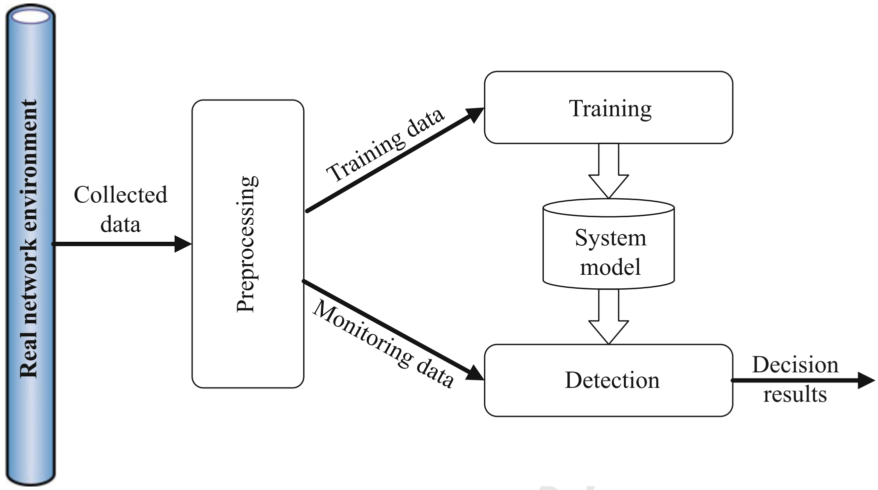


Fig. 6.2 ML-NIDS architecture

ral networks (ANN) [24]. In addition, unsupervised learning algorithms, including k-means clustering [25] and self-organized maps (SOM) [10], have also been applied to deal with network intrusion detection problems, with good results. For various reasons, such as the imbalance of training datasets and the high cost of computational requirement, it is currently very difficult to design a single machine learning approach that outperforms the existing ones. Therefore, hybrid machine learning approaches, such as clustering with classifier [16, 26] and hierarchical classifiers [27], have attracted a lot of attention in recent years. In addition, some data mining approaches have also been successfully utilized to solve intrusion detection problems. For instance, data mining approaches are employed to generate a fuzzy rule base, and a fuzzy inference approach is then applied for threat detection in [14]. This section examines the existing NIDSes utilizing two approaches, namely, fuzzy inference systems and artificial neural networks.

6.3.1 Fuzzy Inference Systems

Due to their great ability to deal with uncertainty, fuzzy inference systems (FIS) have been widely used for detecting potential network threats. Generally speaking, fuzzy inference systems are built upon fuzzy logic to map system inputs and outputs. A typical fuzzy inference system consists of two main parts: a rule base (or knowledge base) and an inference engine. A number of inference engines are well established, with the Mamdani inference [28] and the TSK inference [29] being the most widely used. Although fuzzy sets are used in both rule antecedents and rule consequences by the Mamdani fuzzy model, which is more intuitive and suitable for handling linguistic variables, a defuzzification progress is required to transfer the fuzzy outputs to crisp outputs. In contrast, the TSK inference approach produces crisp outputs directly, as crisp polynomials are used as rule consequences.

For a fuzzy inference-based NIDS (FIS-NIDS), the important features, which are extracted from the network packets, are used in the pre-detector component to analyze events with the set of rules to determine whether any incoming events have intrusive patterns or not. The set of rules is called a fuzzy rule base, which can be either predefined by expert knowledge (knowledge-driven), or extracted from labeled data instances (data-driven) [30, 31]. In contrast to knowledge-driven rule base generation approaches, which essentially limit the system's applicability as expert knowledge is not always available in some areas, data-driven rule base generation methods are most commonly used for intelligent NIDSes. Several data-driven approaches have been proposed to generate a rule base for FIS-NIDS use, which are usually derived from complete and dense datasets, such as [32, 33]. The generated rule bases are often optimized using a general optimization technique, such as genetic algorithms (GA), for optimal system performance. As the used datasets are dense and complete, the resulted rule bases are generally dense and complete, each of which covers the entire input domain, and accordingly the resulted fuzzy models often yield to great reasoning performance. However, these systems will suffer if only incomplete, imbalanced, and sparse datasets are available. In addition, these systems are usually signature-based NIDSes, which are only able to detect known network threats for which the intrusive patterns have been covered in the rule base.

In order to address the previous limitations, fuzzy interpolation has been used to develop NIDSes [18, 34]. Briefly, fuzzy interpolation enhances conventional fuzzy inference systems to work with sparse fuzzy rule bases, by which some inputs or observations are not covered [35]. Using fuzzy interpolation techniques, even if the traffic patterns of the incoming event do not match with any of the patterns stored in the rule base, an approximated result can still be obtained by considering the similar patters expressed as rules in the current rule base. A number of fuzzy interpolation approaches have been proposed in literature [36–47], many of which have already been applied to solve real-world problems [48–51].

A data-driven fuzzy interpolation-based NIDS can be developed in four steps: (1) training dataset generation and preprocessing, (2) rule base initialization, (3) rule base optimization, and (4) intrusion detection by fuzzy interpolation [14, 52], as illustrated in Fig. 6.3. These key steps are detailed in the following sections.

6.3.1.1 Dataset Generation and Preprocessing

The training dataset can either be collected from a real-world network environment, or it can be developed from an existing dataset. Whichever method is selected, the important features, which are selected for system modeling, have to be identified. In general, a number of features can be monitored by networking tools for network analysis during data packet transmission over the network, but some of these features are redundant or noisy. Therefore, a well-thought manual feature selection process

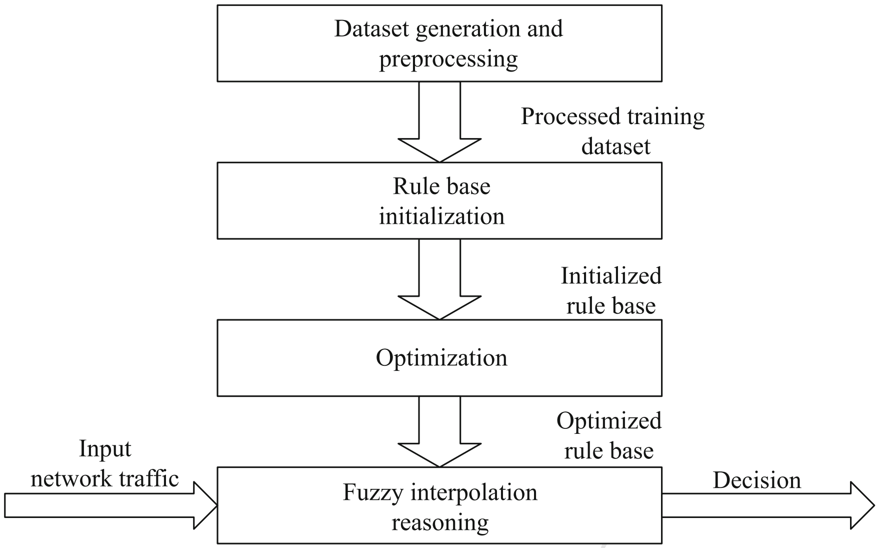


Fig. 6.3 The framework of TSK+ based NIDS

is often required for network attack detection [53]. This common practice is also applied here. In particular, four important features identified by experts are selected as NIDS signature for the proposed FIS-NIDS, which are listed in Table 6.1.

The establishment of the optimal number of features that should be retained in datasets by feature selection methods is always an argued point, because feature selection usually causes information loss from the original dataset. Several pieces of work in the area of feature selection have claimed that more attributes generally lead to better approximations [54–57]. This can be the case for perfect, entirely consistent, and noise-free data, with all features being independent. Generally speaking, feature relevancy and redundancy have to be considered by feature selection methods before the application of machine learning approaches [58, 59]. The selected features should be highly relevant to the problem and non-redundant if they are to be useful in an efficient manner [60]. In fact, a large volume of published results in relevant

Table 6.1 Features used in the NIDS

Feature	Description
Source bytes	The number of data bytes sent by the source IP host
Destination bytes	The number of data bytes sent by the destination IP host
Count	The number of connections to the same host as the current connection in the past 2 seconds
Dst_Host_Diff_Rate	% of connections whose ports are different, among the past 100 connections with the same destination IP

literature has demonstrated that smaller number of selected features can lead to much-improved modeling accuracy, including [61–66]. In addition, more attributes retained in datasets will also increase the computational complexity [60]. Therefore, it is necessary to consider as many features as possible under certain circumstances especially for noise-free and fully consistent datasets, but in others, a minimal subset of features satisfying some predefined criteria is more appealing.

Once the features are determined for machine learning, datasets for a given network of a particular environment need to be collected for model training. This is typically implemented in stages based first on an attack-free network, and then different types of attacks that need to be identified. In other words, data regarding normal network traffic is collected first from a threat-free network environment. Then, a number of attacks simulating the first type of attack are artificially launched so that this type of attack is sufficiently covered by the dataset. This process is repeated for every other type of attacks until all the classes that need to be considered are fully covered by the dataset. The final dataset covers all attack types and attack-free situations. In most cases, if an existing dataset is adopted for model training, the process of data collection may be skipped. However, ideally, the structure of the existing dataset should follow the structure explained above.

6.3.1.2 Rule Base Initialization

Suppose that the training dataset (T) contains l ($l \geq 1$, $l \in \mathbb{N}$) labeled classes, which covers $l - 1$ types of attacks and the normal situation. As illustrated in Fig. 6.4, the system first divides the training dataset T into l sub-datasets $T_1, T_2, \ldots, T_l$, each representing a type of attack or the normal traffic (i.e., $T = \cup_{s=1}^{l} T_s$).

Then, the K-means, one of the most widely used clustering algorithms, is employed to each sub-dataset to group its data points into k clusters based on their feature values. Note that the value of k in the K-means algorithm has to be predefined to enable the application of the algorithm. The Elbow method [67], which determines the number of clusters based on the criteria that adding another cluster is not much better for modeling the dataset, has been employed for determining the value of k. Based on this, each determined cluster is expressed as a fuzzy rule that contributes to the TSK rule base.

In this work, a 0-order TSK fuzzy model is adopted. All data instances in each class share the class label (an integer number), which is utilized as the consequent of the corresponding TSK rule. The triangular membership function is utilized in the rule antecedents. The support of the triangular fuzzy set is expressed as the span of the cluster along this input dimension, and the core of the corresponding fuzzy set is set as the cluster center. The final TSK fuzzy rule base is generated by combining all the extracted rules from all l sub-datasets, which is illustrated as follows:

$$
\begin{aligned}
R_{t_s}^{s}: \ & \textbf{IF}\, x_1 \text{ is } A_1^{st_s} \text{ and } x_2 \text{ is } A_2^{st_s} \text{ and } x_3 \text{ is } A_3^{st_s} \text{ and } \text{ and } x_4 \text{ is } A_4^{st_s}, \\
& \textbf{THEN}\, z = s,
\end{aligned}
\tag{6.1}
$$

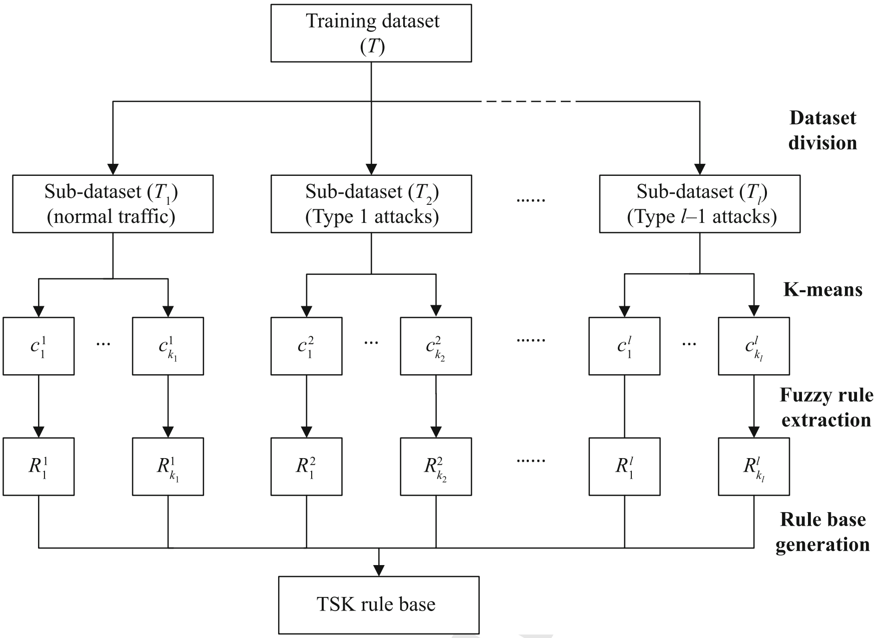


Fig. 6.4 Rule base generation

where $s = \{1, \ldots, l\}$ represents the s*th* sub dataset that indicates the s*th* type of network traffic, $t_s = \{1, \ldots, k_s\}$ denotes the t*th* cluster in the s*th* sub dataset. The number of rules in this rule base is equal to the sum of the numbers of clusters for all the sub-datasets (i.e., $k_1 + k_2 + \cdots + k_l$).

6.3.1.3 Rule Base Optimization

The generated initial rule base can be employed for intrusion detection, but with relatively poor performance. In order to increase the detection performance, a genetic algorithm (GA) is adopted here to fine-tune the membership functions involved in the initial rule base. Assume that a given initial TSK rule base is comprised of n fuzzy rules of the form shown in Eq. 6.1. Suppose a chromosome, denoted as I, is used to represent a potential solution in the GA, which is coded to represent the parameters of all rules in the rule base, as shown in Fig. 6.5. Based on this, the initial population $\mathbb{P} = \{I_1, I_2, \ldots, I_{|\mathbb{P}|}\}$ can be formed by taking the parameters of the initial rule base and its random variations. During the optimization process, the number of chromosomes is selected for offspring reproduction by applying the genetic operators of crossover and mutation. Specifically, the fitness proportionate selection method, also known as the roulette wheel selection, is implemented in this work for chromosome selection, and the signal point crossover and mutation operators are employed for

462756_1_En_6_Chapter ☑ TYPESET ☐ DISK ☐ LE ☑ CP Disp.:23/7/2018 Pages: 30 Layout: T1-Standard

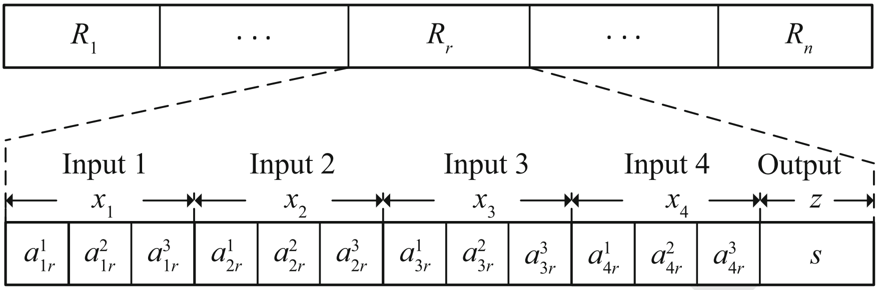


Fig. 6.5 Chromosome encoding

reproduction. In addition, in order to make sure that the resultant fuzzy sets are valid and convex, the constraint $a^1_{ir} < a^2_{ir} < a^3_{ir},\ i = \{1, 2, 3, 4\}$ is enforced to the genes during optimization. The selection and reproduction processes are iterated until the predefined maximum number of iterations is reached, or until the system performance reaches a predefined threshold. Optimized parameters and thus an optimized rule base can be obtained when the termination condition is satisfied.

6.3.1.4 Intrusion Detection by TSK-Interpolation:

Once the rule base is generated, the TSK+ fuzzy inference approach can be deployed to perform inferences for attack detection. In order to generate network intrusion alerts in real time, the system is deployed by one of the deployment methods introduced in Sect. 6.2.1, which keeps capturing network traffic data for analysis. For each captured network packet, four important features, detailed in Table 6.1, are extracted and fed into the proposed system. From this input, the TSK+ fuzzy inference approach will classify the types of network traffic using the generated rule base. Assume that an optimized TSK fuzzy rule base is comprised of n rules as follows:

$$\begin{aligned} R_1 &: \textbf{IF}\ x_1 \text{ is } A^1_1 \text{ and } x_2 \text{ is } A^1_2 \text{ and } x_3 \text{ is } A^1_3 \text{ and } x_4 \text{ is } A^1_4\ \textbf{THEN}\ z = \mathbb{Z}_1, \\ &\ \ldots \\ R_n &: \textbf{IF}\ x_1 \text{ is } A^n_1 \text{ and } x_2 \text{ is } A^n_2 \text{ and } x_3 \text{ is } A^n_3 \text{ and } x_4 \text{ is } A^n_4\ \textbf{THEN}\ z = \mathbb{Z}_n, \end{aligned} \tag{6.2}$$

where $A^i_k (k \in \{1, 2, 3, 4\}$ and $i \in \{1, \ldots, n\})$ represents a normal and convex triangular fuzzy set in the rule antecedent denoted accordingly as $(a^i_{k_1}, a^i_{k_2}, a^i_{k_3})$, and $\mathbb{Z}_i$ is an integer number that indicates the type of network traffic, whether it is normal traffic or a particular type of attack. By taking a captured network packet as an example, the working procedure of the TSK+ fuzzy inference for intrusion detection can be summarized as the following steps:

1. Extract the four feature values from the network packet, and express them in the form $I = \{x^*_1, x^*_2, x^*_3, x^*_4\}$, which will be used as the system input. Note that the

462756_1_En_6_Chapter ☑ TYPESET ☐ DISK ☐ LE ☑ CP Disp.:23/7/2018 Pages: 30 Layout: T1-Standard

extracted feature values are normally crisp values. They have to be represented as fuzzy sets of the form $A_k^* = (x_k^*, x_k^*, x_k^*)$, where $k = \{1, 2, 3, 4\}$, for future use.

2. Determine the matching degree $S(A_k^*, A_k^i)$ between the inputs $I = \{A_1^*, A_2^*, A_3^*, A_4^*\}$ and rule antecedents $(A_1^i, A_2^i, A_3^i, A_4^i)$ for each rule R_i, $i = \{1, \ldots, n\}$ using:

$$S(A_k^*, A_k^i) = \left(1 - \frac{\sum_{j=1}^{3} |x_k^* - a_{kj}^i|}{3}\right) \cdot DF\ , \tag{6.3}$$

where DF, termed as distance factor, is a function of the distance between the two fuzzy sets of interest, which is defined as follows:

$$DF = 1 - \frac{1}{1 + e^{-sd+5}}\ , \tag{6.4}$$

where s ($s > 0$) is a sensitivity factor, and d represents the Euclidean distance between the two fuzzy sets. A smaller s value results in a similarity degree more sensitive to the distance of the two fuzzy sets.
3. Obtain the firing degree of each rule by integrating the matching degrees of its antecedents and the given input values as follows:

$$\alpha_i = S(A_1^*, A_1^i) \wedge S(A_2^*, A_2^i) \wedge S(A_3^*, A_3^i) \wedge S(A_4^*, A_4^i)\ , \tag{6.5}$$

where $\wedge$ is a t-norm usually implemented as a minimum operator.

4. Integrate the sub-consequences from all rules to get the final output using the following formula:

$$z = \frac{\sum_{i=1}^{n} \alpha_i \cdot \mathbb{Z}_i}{\sum_{i=1}^{n} \alpha_i}. \tag{6.6}$$

5. Apply the round function on the final output to obtain the integer number that indicates the network traffic type for the given network packet.

As discussed above, if an unknown network's threat behavior or traffic pattern has been captured, a result of "network security alert" can still be expected by considering all fuzzy rules in the rule base.

6.3.2 Artificial Neural Networks

An artificial neural network (ANN) is an information processing system inspired by biological nervous systems that constitute animal brains, which is one of the most widely used machine learning algorithms [68]. Typically, an ANN is composed of

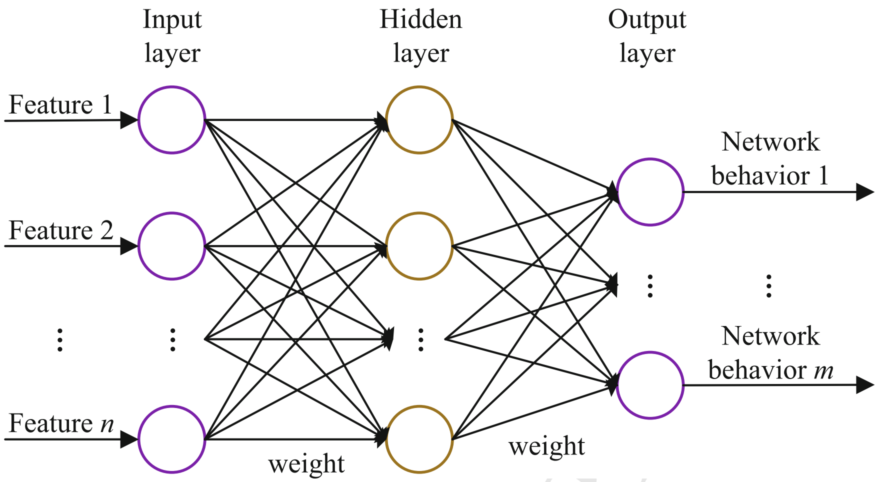


Fig. 6.6 Multilayer perception-based NIDS architecture

two main parts: a set of simple processing units, also known as nodes or artificial neurons, and the connections between these. These simple units or nodes are organized in layers, which usually consist of the input, output, and hidden layers. The hidden layers are those between the input and the output layers. Once the set of processing units and their connections are determined, or an ANN is built, the training process adjusts the connection weights between the connected units to determine to what extent one unit will affect the others. ANNs are successfully employed in NIDSes, which usually fall into two categories: supervised training-based NIDSes and unsupervised training-based NIDSes [69]. As demonstrated in Fig. 6.2, both types of NIDSes essentially follow the architecture and three general steps of ML-NIDS as specified in the beginning of this section.

If the supervised learning approach is applied, the desired output or pattern for a given input is learned from a set of labeled data. A well-known supervised neural network architecture is the multilayer perception (MLP), which is based on the feed-forward and backpropagation algorithms with one or more layers between the input and the output layer [1]. In this type of ANN-NIDS, the number of nodes in the input layer is set to the number of features selected from the original traffic flow, and the number of nodes in the output layer is configured to be the number of desired output classes [16, 70–73]. The number of hidden layers and the number of nodes for each hidden layer vary, and are usually configured according to the situation. A feed-forward-based MLP with a signal hidden layer ANN NIDS model is illustrated in Fig. 6.6.

Obviously, the entire data flow in the ANN is in one direction only: from the input layer, though the hidden layer, to the output layer (see Fig. 6.6). Therefore, given a network traffic package as the input, the corresponding network behavior can be predicted. The advantages of this model are its ability to represent both linear and non-linear relationships, and directly learn these relationships from the data by means of training. However, a number of research projects have reported that the training process of this type of ANN can be very time-consuming, which may pose a significant adverse impact for NIDS system updating [1, 24].

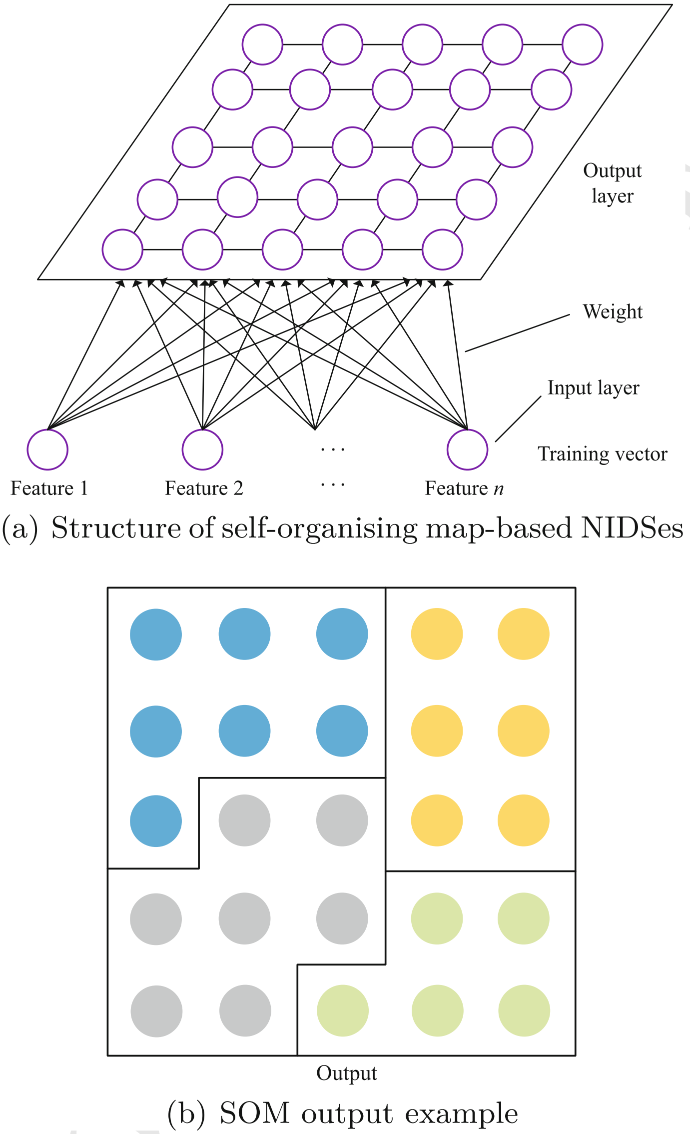
Another group of ANN NIDSes is based on unsupervised training, in which the network adapts to different clusters without having a desired output. One of the most popular algorithms in this group is the self-organizing map (SOM), which transforms the input of arbitrary dimension into a low-dimensional (usually 1- or 2-dimensional) discrete map by using Kohonen's unsupervised learning method [74]. The structure of a conventional self-organizing map is shown in Fig. 6.7a. A conventional SOM network model usually has two layers: an input layer and an output layer (also known as a competitive layer). Similar to the supervised training-based NIDS, the number of nodes in the input layer are usually set to the number of selected features of the training dataset. The output layer consists of neurons organized in a lattice, usually a finite two-dimensional space. Each neuron has a specific topological position and is associated with a weight vector of the same dimension as the input vectors [75].

The training process adjusts the weight vectors of the neurons, thereby describing a mapping from a higher-dimensional input space to a lower-dimensional map space. As a result, the SOM eventually settles into a map of stable zones as a type of feature map of the input space. Based on these mappings, various traffic behaviors can be identified. Figure 6.7b illustrates an example of a SOM output, which clearly shows the four classes that have eventually been predicted.

When comparing the performance (speed and conversion rate) between SOM and supervised learning-based NIDS systems, it becomes clear that SOM is more suitable for real-time intrusion detection [76–80].

Although both types of ANN-network intrusion detection systems are successfully employed in detecting intrusions in real-world network environments with promising results, existing ANN-network intrusion detection systems have two main drawbacks: (1) lower detection precision for low-frequency attacks, and (2) weaker detection stability, which limits the applicability of such systems [16]. The reason behind these is the uneven distribution of different attack types. For example, the number of training data instances for low-frequency attacks are very limited compared to common attacks. As a consequence, it is not easy for the ANN to learn the characteristics of such low-frequency attacks [81].

To address these issues, a number of solutions have been proposed (e.g., [16, 82, 83]). Among these systems, a fuzzy clustering-based neural network NIDS approach (FC-ANN-NIDS) [16] can be a potential solution. Comparing to conventional ANN-NIDSes, in which data clustering techniques are typically not involved during the training process, FC-ANN-NIDSes adopt a fuzzy clustering technique to generate different training sub-datasets. This is followed by the application of multiple ANNs in the training stage based on the divided sub-datasets. Finally, a fuzzy aggregation module is applied to combine the results of the ANNs, in an effort to eliminate their errors. The framework of FC-ANN-NIDS is illustrated in Fig. 6.8, which basically contains three major stages: clustering, ANN modeling, and fuzzy aggregation. The details of this method (or FC-ANN-NIDS) are presented in the rest of this section.



(a) Structure of self-organising map-based NIDSes

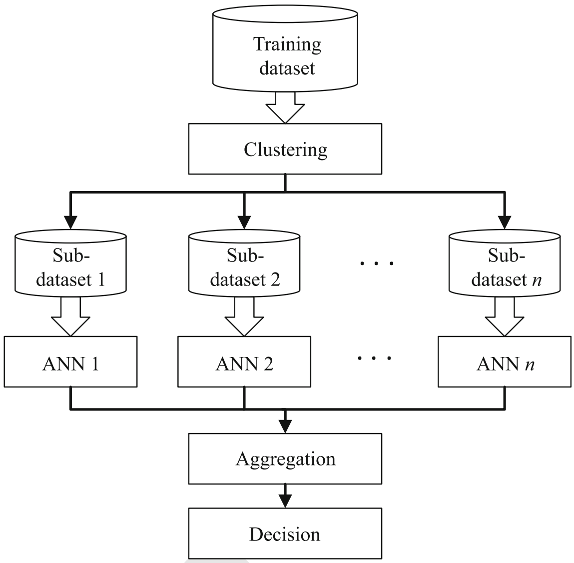
(b) SOM output example

Fig. 6.7 Self-organizing map-based NIDS architecture

6.3.2.1 Clustering

Given a training dataset that contains l network behaviors, the fuzzy C-means clustering technique [84] is employed to group the data instances in clusters, which essentially divides the entire training dataset into n sub-datasets. Note that only the

Fig. 6.8 FC-ANN-NIDS framework



size and complexity of the original training dataset is reduced after data clustering, and the data instances in each divided sub-dataset may still cover all the l network behaviors. Each divided training dataset will be forwarded to the next stage for ANN training. Unfortunately, the value of n (the number of clusters) in the proposed system is determined under a practice theory. Therefore, more intelligent methods, such as Elbow method [67] may be considered for determining the value of n.

6.3.2.2 ANN Training

A multi-layer perceptron model, illustrated in Fig. 6.6, is used in this study for modeling each sub-training dataset. As mentioned previously, the number of input nodes is set to match the number of selected features of the training dataset; and the number of nodes in the output layer is set to the number of network traffic behaviors covered by the training dataset. The number of hidden nodes is then obtained by adopting the empirical formula: $\sqrt{I+O}+\alpha,\ (\alpha=\{1,\ldots,10\})$, where I denotes the number of input nodes, O represents the number of nodes in the output layer, and α is a random number [81]. During the training process, the signals, which combine both the input values and the weight values between the corresponding input node and the hidden node, are received by each node in the hidden layer. These signals are processed by a sigmoid activation function, and broadcasted to all the neurons in the output layer with a special weight value. In this study, the most widely used first-order optimization algorithm, gradient descent, is employed for weight-updating during the backpropagation process. Once the entire training process is completed, multiple ANN models can be generated based on the different training sub-datasets. Note that

462756_1_En_6_Chapter ✓ TYPESET ☐ DISK ☐ LE ✓ CP Disp.:23/7/2018 Pages: 30 Layout: T1-Standard

each ANN model can be applied individually for network intrusion detection in real-world network environments. In order to reduce the detection errors, an aggregation module is applied to aggregate the results from different ANNs.

6.3.2.3 Aggregation

Although each ANN generated in the last stage can be deployed individually as an NIDS, some of them may have an unacceptably poor detection performance. In this study, another multi-layer perceptron model is applied for sub-results aggregation. In this stage, the number of nodes in both the input and the output layer is set to the number of network behaviors. Given the entire training dataset and the multiple trained ANN models with the corresponding training sub-datasets generated in the last stage, the modeling process in the aggregation stage can be summarized as follows:

Step 1: Feed each data instance j in the original training dataset to every trained ANN model $(ANN_1, ANN_2, \ldots, ANN_n)$. Denote the output of model ANN_i, $(i = \{1, \ldots, n\})$ from data instance j as o_i^j, then the outputs from all ANNs collectively as O^j and $O^j = [o_1^j, \ldots, o_n^j]$.

Step 2: Form the new input for the new ANN model based on the previous outputs. The new input I_{new}^j generated from data instance j is

$$I_{New}^j = [o_1^j \cdot \mu_1, \ldots, o_n^j \cdot \mu_n] \,, \tag{6.7}$$

where μ_i represents the degree of membership of data instance j belonging to cluster i. Note that the degree of membership for each data instance regarding each cluster has been determined in the clustering using the fuzzy C-means clustering algorithm.

Step 3: Generate a new ANN model and train it using the newly formed inputs generated in Step 2.

Once the entire model is built, the system can be deployed in real-world network environments for intrusion detection. Given an incoming network traffic package, the system first calculates the membership of the incoming data using the cluster centers obtained in the first stage. Next, the ANN models and the aggregation model will be applied to predict the final result, which indicates whether the incoming traffic poses a threat. Such hybrid ANN network intrusion detection solutions can increase detection performance, especially for low-frequency attacks. However, it may be costly in time because of the training processes for the large number of feed-forward neural networks.

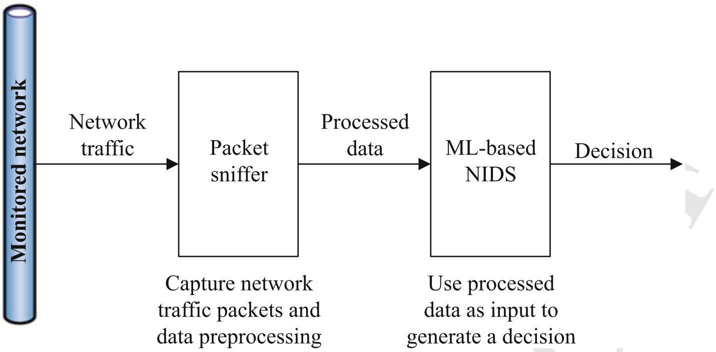


Fig. 6.9 The framework of ML-NIDS deployment

6.3.3 Deployment of ML-Based NIDSes

Although the developed ML-based network intrusion detection systems are able to take the network package (input) to predict whether it is a normal network behavior, these systems still cannot be directly implemented in real-world network environments for real time detection. The reason behind this is that the generated ML-based models do not have packet sniffers, which are used to capture the network traffic in real time. In order to achieve real-time detection, the developed ML-based network intrusion detection systems have to work with packet sniffers, such as Snort, Bro, or Spark. A packet sniffer (or network sniffer) is a network traffic monitoring and analyzing tool that can sniff out the network data traversing the monitored network in real time. A number of ML-based network intrusion detection systems have been successfully integrated with packet sniffers and achieved good real-time detection (e.g., [34, 85]). The general framework of these systems is illustrated in Fig. 6.9. A packet sniffer, which can be implemented by either a passive or an in-line deployment method as introduced in Sect. 6.2.1, continuously captures the network traffic, and extracts the required information from the captured network packets to feed into the system model developed by machine learning techniques, thereby generating the final decisions.

6.4 Experiment

A number of network intrusion detection systems developed by different machine learning approaches are evaluated in this section by applying them to the KDD 99 benchmark dataset.

6.4.1 Evaluation Environment

A well-known benchmark dataset, KDD 99, which has been utilized in a number pieces of recent research [14, 16, 18, 86], is used in this work to evaluate multiple machine learning-based network intrusion detection systems. The KDD 99 dataset is a popular benchmark for intrusion detection; it includes legitimate connections and a wide variety of intrusions simulated in a military network environment [87]. This dataset contains almost 5 million data instances with 42 attributes, including the "class" attribute, which indicates whether a given instance is a normal connection instance or one of the four types of attacks to be identified (i.e., normal, denial of service attacks, user-to-root attacks, remote-to-user attacks, and probes). An important feature of this dataset is that it is an imbalanced dataset, with most data instances belonging to the normal, denial of service attack, and probe categories. As with the type of low-frequency attacks, the classes of user-to-root attacks and remote-to-user attacks, are only covered by a small number of data samples. Knowing the inherent issues associated with the dataset, such as the high duplication rate of 78% [87], data instance selection methods, such as the random selection method, are used to reduce the size of the dataset for machine learning. It is worth mentioning that the KDD 99 dataset has been succeeded by the NSL-KDD-99 dataset [87], which reduces the size to 125,937 data samples, while keeping all the features of the original dataset. Table 6.2 details the information about the number of data instances in the training and testing datasets that were used by different network intrusion detection systems, as discussed in Sect. 6.3.

Table 6.2 Details of datasets for machine learning-based NIDSes

Machine learning approach	Training		Testing		Dataset
	Normal	Abnormal	Normal	Abnormal	
TSK+ [14]	67,343	58,630	9,711	9,083	Entire NSL-KDD-99
Conventional fuzzy inference [33]	67,343	58,630	9,711	9,083	Entire NSL-KDD-99
FC-ANN [16]	3,000	15,285	60,593	250,496	Random part
MLP [24]	5,922	6,237	3,608	3,388	Random part
SOM [10]	97,277	396,744	60,593	250,436	Random part
Hierarchical SOM [10]	97,277	396,744	60,593	250,436	Random part

6.4.2 Model Construction

This section details the model construction of the aforementioned six ML-based network intrusion detection approaches.

6.4.2.1 TSK+ Fuzzy Inference

As discussed in Sect. 6.3.1, this system brings four important features to the system model. During rule base initialization, the training dataset was divided into five sub-datasets based on the five symbolic labels, which are represented by five integer numbers. The fuzzy model takes four inputs, and predicts a crisp number. According to the Elbow method, 46 TSK fuzzy rules have been generated, which constructed the initial rule base. The final rule base has then been optimized using the GA. The objective function in this work is defined as the root mean square error (RMSE), while the GA parameters are listed in Table 6.3.

6.4.2.2 Conventional Mamdani Fuzzy Inference

The conventional Mamdani fuzzy inference model is investigated in this work. The system uses 34 features for system modeling, which results in 34 inputs and one output Mamdani fuzzy model. Each input domain has been equally partitioned into four regions, described by four linguistic terms, namely, "very low," "low," "medium," and "high;" and two fuzzy sets, "low" and "high," are used to indicate normal and abnormal network traffic, respectively. The fuzzy rules are obtained by a mapping mechanism based on the given training dataset. Given the input, which is a network traffic package, the system first fuzzifies the crisp value of the required features based on the mapping mechanism, then generates a fuzzy output based on the generated rule base. Finally, the center of gravity method is employed to defuzzify the fuzzy output to a crisp one, which indicates whether the traffic is normal or attack traffic.

Table 6.3 GA parameters

Parameters	Values
Population size	100.00
Crossover rate	0.85
Mutation rate	0.05
Maximum iteration	10,000.00
Termination threshold	0.01

6.4.2.3 Fuzzy Clustering-Based ANN

Fuzzy clustering-based ANN uses all the 41 features to predict the five network behaviors. Note that the symbolic values contained in the dataset have been converted to continuous values. In the beginning, six training sub-datasets are obtained by using fuzzy C-means clustering. From there, six signal-hidden-layered neural network models are trained, each of which is referred to as [41;18;5] structure. This means that each network takes 41 inputs, goes through 18 hidden nodes, and finally produces 5 outputs. In the aggregation progress, a new signal hidden-layered ANN model with the structure [5;13;5] is designed to aggregate all results from upper-level ANN models. The mean square error (MSE) is used as the fitness function during system modeling, and the threshold of MSE is set to 0.001. Also, the learning rate and the momentum factor at both ANN model levels are set to 0.01 and 0.2, respectively.

6.4.2.4 Multilayer Perceptron

Expert knowledge has been used in this work to help select the most important features. In particular, 35 features, including five symbolic features and 30 numerical features, have been selected. Similar to the FC-ANN approach introduced above, the symbolic values were converted to numerical values. Because of the lack of data samples in U2R and R2U attacks, only three categories, namely, "normal," "DoS," and "probes," were considered. As a result, 35 input nodes and three output nodes were used. In this experiment, a two hidden-layered MLP network model was implemented, constituting a four-layer MLP, whose structure is referred to as [35;35;35;3].

6.4.2.5 Hierarchical Self-organizing Maps

A hierarchical self-organizing map architecture, which consists of two levels of SOM networks, each comprised of three layers, was used in this experiment. The first layer was an input layer, with 20 input nodes (corresponding to 20 selected features). At the first level of SOM, six SOM networks were deployed, each of which represented one of the basic TCP features, including "duration," "protocol type," "service," "flag," "destination bytes," and "source bytes." During the training process, each training data sample was fed into each SOM network, thereby creating a number of mappings between inputs and six 6×6 grids on the second layer, which resulted in $36 \times 6 = 216$ neurons. After this, potential function clustering [84] was employed on each output layer of the first SOM level to reduce the total neurons from 36 to 6. As a consequence, the total number of neurons in the second layer was reduced to 36. These 36 neurons were used as inputs for the second SOM level to train a new SOM network that consists of a 20×20 grid of neurons, which indicates the mapping from the input space to the different network behaviors. The learning rate was set to 0.05, and the neighborhood function was configured as a Gaussian function.

6.4.2.6 Conventional Self-organizing Maps

In this experiment, all the 41 features have been used for the intrusion detection system. During the training process, the learning rate was set to 0.05, and the Gaussian function was used as the neighborhood function. The developed system took 41 inputs to create a mapping between five categories of network behaviors into a 6×6 grid of neurons.

6.4.3 Result Comparisons

In order to enable a direct comparison between the different ML-NIDS approaches, a common measurement, the detection rate, is employed in this work. In particular, the detection rate can be defined as follows:

$$\text{Detection rate} = \frac{\text{Number of instances correctly detected}}{\text{Total number of instances}} \cdot 100 \qquad (6.8)$$

The detection rates of the classification results for each network traffic category are summarized in Table 6.4.

The results show that all the approaches achieved a high detection performance in the normal, DoS, and probes category, which contain sufficient data samples for training. Note that conventional ANN-based network intrusion detection systems, such as the MLP-based approach and the SOM-based approach, led to an extremely poor detection performance in the case of U2R and R2U. As discussed in Sect. 6.3.2, this issue is caused by the lack of training data samples for both U2R and R2U. In this case, a future investigation may be required to identify how the detection threshold affects the detection performance. Obviously, similar to the modified version of the ANN approaches, the FC-ANN-based approach and the hierarchical SOM-based

Table 6.4 Performance comparison

Approach	Normal	DoS	U2R	R2U	Probes
TSK+ [14]	93.10	97.84	65.38	84.65	85.69
Conventional fuzzy inference [33]	82.93	90.42	19.05	15.58	37.08
FC-ANN [16]	91.32	96.70	76.92	58.57	80.00
MLP [24]	89.20	90.90	N/A	N/A	90.30
SOM [10]	98.50	96.80	0.00	0.15	63.40
Hierarchical SOM [10]	92.40	96.50	22.90	11.30	72.80

approach increased the detection rate. It is worth mentioning that the TSK+ based intrusion detection system not only achieved the best detection performance in the normal, DoS, and probes classes, but also had an outstanding performance in the other two classes.

6.5 Conclusion

This chapter investigated how machine learning algorithms can be used to develop NIDSes. In particular, the chapter first reviewed the existing intrusion detection techniques, including hardware deployment and software implementations. They are followed by the discussion of a number of machine learning algorithms and their applications in network intrusion detection. Finally, a well-known network security benchmark dataset, KDD 99, was employed for the evaluation of the reviewed machine learning-based network intrusion detection systems, with a critical analysis of the results. Although the benchmark dataset, KDD 99, is still popular in recent research, it is relatively outdated and many of today's network threats are not covered by the KDD 99 dataset. Therefore, future research may consider using alternate datasets (e.g., [88, 89]). In addition, as IoT continues to expand, the data being generated will continue to grow in volume and velocity. How conventional machine learning and artificial intelligence techniques can be expanded to deal with the continuously growing data is an interesting research direction.

References

1. Stampar M, Fertalj K (2015) Artificial intelligence in network intrusion detection. In: Biljanovic P, Butkovic Z, Skala K, Mikac B, Cicin-Sain M, Sruk V, Ribaric S, Gros S, Vrdoljak B, Mauher M, Sokolic A (eds) Proceedings of the 38th international convention on information and communication technology, electronics and microelectronics, pp 1318–1323. https://doi.org/10.1109/MIPRO.2015.7160479
2. Sommer R, Paxson V (2010) Outside the closed world: on using machine learning for network intrusion detection. In: Proceedings of the 2010 IEEE symposium on security and privacy. IEEE Computer Society, Los Alamitos, CA, USA, pp 305–316. https://doi.org/10.1109/SP.2010.25
3. Buczak AL, Guven E (2016) A survey of data mining and machine learning methods for cyber security intrusion detection. IEEE Commun Surv Tutor 18(2):1153–1176. https://doi.org/10.1109/COMST.2015.2494502
4. Russell SJ, Norvig P (2009) Artificial intelligence: a modern approach, 3rd edn. Pearson, Essex
5. Farnaaz N, Jabbar M (2016) Random forest modeling for network intrusion detection system. Procedia Comput Sci 89:213–217. https://doi.org/10.1016/j.procs.2016.06.047
6. Ma Z, Kaban A (2013) K-nearest-neighbours with a novel similarity measure for intrusion detection. In: Jin Y, Thomas SA (eds) Proceedings of the 13th UK workshop on computational intelligence. IEEE, New York, pp 266–271. https://doi.org/10.1109/UKCI.2013.6651315
7. Mukherjee S, Sharma N (2012) Intrusion detection using Naïve Bayes classifier with feature reduction. Proc Tech 4:119–128. https://doi.org/10.1016/j.protcy.2012.05.017

8. Thaseen IS, Kumar CA (2017) Intrusion detection model using fusion of chi-square feature selection and multi class SVM. J King Saud Univ Comput Inf Sci 29(4):462–472. https://doi.org/10.1016/j.jksuci.2015.12.004
9. Zhang C, Zhang G, Sun S (2009) A mixed unsupervised clustering-based intrusion detection model. In: Huang T, Li L, Zhao M (eds) Proceedings of the third international conference on genetic and evolutionary computing. IEEE Computer Society, Los Alamitos, CA, USA, pp 426–428. https://doi.org/10.1109/WGEC.2009.72
10. Kayacik HG, Zincir-Heywood AN, Heywood MI (2007) A hierarchical SOM-based intrusion detection system. Eng Appl Artif Intell 20(4):439–451. https://doi.org/10.1016/j.engappai.2006.09.005
11. Garfinkel S (2002) Network forensics: tapping the Internet. https://paulohm.com/classes/cc06/files/Week6%20Network%20Forensics.pdf
12. Liao HJ, Lin CHR, Lin YC, Tung KY (2013) Intrusion detection system: a comprehensive review. J Netw Comput Appl 36(1):16–24. https://doi.org/10.1016/j.jnca.2012.09.004
13. Bostani H, Sheikhan M (2017) Modification of supervised OPF-based intrusion detection systems using unsupervised learning and social network concept. Pattern Recogn 62:56–72. https://doi.org/10.1016/j.patcog.2016.08.027
14. Li J, Yang L, Qu Y, Sexton G (2018) An extended Takagi-Sugeno-Kang inference system (TSK+) with fuzzy interpolation and its rule base generation. Soft Comput 22(10):3155–3170. https://doi.org/10.1007/s00500-017-2925-8
15. Ramadas M, Ostermann S, Tjaden B (2003) Detecting anomalous network traffic with self-organizing maps. In: Vigna G, Krügel C, Jonsson E (eds) Recent advances in intrusion detection. Springer, Heidelberg, pp 36–54. https://doi.org/10.1007/978-3-540-45248-5_3
16. Wang G, Hao J, Ma J, Huang L (2010) A new approach to intrusion detection using artificial neural networks and fuzzy clustering. Expert Syst Appl 37(9):6225–6232. https://doi.org/10.1016/j.eswa.2010.02.102
17. Wang W, Battiti R (2006) Identifying intrusions in computer networks with principal component analysis. In: Revell N, Wagner R, Pernul G, Takizawa M, Quirchmayr G, Tjoa AM (eds) Proceedings of the first international conference on availability, reliability and security. IEEE Computer Society, Los Alamitos, CA, USA. https://doi.org/10.1109/ARES.2006.73
18. Yang L, Li J, Fehringer G, Barraclough P, Sexton G, Cao Y (2017) Intrusion detection system by fuzzy interpolation. In: Proceedings of the 2017 IEEE international conference on fuzzy systems. https://doi.org/10.1109/FUZZ-IEEE.2017.8015710
19. Sekar R, Gupta A, Frullo J, Shanbhag T, Tiwari A, Yang H, Zhou S (2002) Specification-based anomaly detection: a new approach for detecting network intrusions. In: Proceedings of the 9th ACM conference on computer and communications security. ACM, New York, pp 265–274. https://doi.org/10.1145/586110.586146
20. Tseng CY, Balasubramanyam P, Ko C, Limprasittiporn R, Rowe J, Levitt K (2003) A specification-based intrusion detection system for AODV. In: Swarup V, Setia S (eds) Proceedings of the 1st ACM workshop on security of ad hoc and sensor networks. ACM, New York, pp 125–134. https://doi.org/10.1145/986858.986876
21. Bostani H, Sheikhan M (2017) Hybrid of anomaly-based and specification-based IDS for Internet of Things using unsupervised OPF based on MapReduce approach. Comput Commun 98:52–71. https://doi.org/10.1016/j.comcom.2016.12.001
22. Mukkamala S, Sung A (2003) Feature selection for intrusion detection with neural networks and support vector machines. Trans Res Rec 1822:33–39. https://doi.org/10.3141/1822-05
23. Kumar M, Hanumanthappa M, Kumar TVS (2012) Intrusion detection system using decision tree algorithm. In: Proceedings of the 14th IEEE international conference on communication technology. IEEE, New York, pp 629–634. https://doi.org/10.1109/ICCT.2012.6511281
24. Moradi M, Zulkernine M (2004) A neural network based system for intrusion detection and classification of attacks. http://research.cs.queensu.ca/~moradi/148-04-MM-MZ.pdf
25. Ravale U, Marathe N, Padiya P (2015) Feature selection based hybrid anomaly intrusion detection system using K means and RBF kernel function. Proc Comput Sci 45:428–435. https://doi.org/10.1016/j.procs.2015.03.174

26. Liu G, Yi Z (2006) Intrusion detection using PCASOM neural networks. In: Wang J, Yi Z, Zurada JM, Lu BL, Yin H (eds) Advances in neural networks–ISNN 2006. Springer, Heidelberg, pp 240–245. https://doi.org/10.1007/11760191_35
27. Chen Y, Abraham A, Yang B (2007) Hybrid flexible neural-tree-based intrusion detection systems. Int J Intell Syst 22(4):337–352. https://doi.org/10.1002/int.20203
28. Mamdani EH (1977) Application of fuzzy logic to approximate reasoning using linguistic synthesis. IEEE Trans Comput C-26(12):1182–1191. https://doi.org/10.1109/TC.1977.1674779
29. Takagi T, Sugeno M (1985) Fuzzy identification of systems and its applications to modeling and control. IEEE Trans Syst Man Cybern SMC-15(1):116–132. https://doi.org/10.1109/TSMC.1985.6313399
30. Li J, Shum HP, Fu X, Sexton G, Yang L (2016) Experience-based rule base generation and adaptation for fuzzy interpolation. In: Cordón O (ed) Proceedings of the 2016 IEEE international conference on fuzzy systems. IEEE, New York, pp 102–109. https://doi.org/10.1109/FUZZ-IEEE.2016.7737674
31. Tan Y, Li J, Wonders M, Chao F, Shum HP, Yang L (2016) Towards sparse rule base generation for fuzzy rule interpolation. In: Cordon O (ed) Proceedings of the 2016 IEEE international conference on fuzzy systems. IEEE, New York, pp 110–117. https://doi.org/10.1109/FUZZ-IEEE.2016.7737675
32. Chaudhary A, Tiwari V, Kumar A (2014) Design an anomaly based fuzzy intrusion detection system for packet dropping attack in mobile ad hoc networks. In: Batra U (ed) Proceedings of the 2014 IEEE international advance computing conference. IEEE, New York, pp 256–261. https://doi.org/10.1109/IAdCC.2014.6779330
33. Shanmugavadivu R, Nagarajan N (2011) Network intrusion detection system using fuzzy logic. Indian J Comput Sci Eng 2(1):101–111
34. Naik N, Diao R, Shen Q (2017) Dynamic fuzzy rule interpolation and its application to intrusion detection. IEEE Trans Fuzzy Syst https://doi.org/10.1109/TFUZZ.2017.2755000
35. Kóczy TL, Hirota K (1993) Approximate reasoning by linear rule interpolation and general approximation. Int J Approx Reason 9(3):197–225. https://doi.org/10.1016/0888-613X(93)90010-B
36. Huang Z, Shen Q (2006) Fuzzy interpolative reasoning via scale and move transformations. IEEE Trans Fuzzy Syst 14(2):340–359. https://doi.org/10.1109/TFUZZ.2005.859324
37. Huang Z, Shen Q (2008) Fuzzy interpolation and extrapolation: a practical approach. IEEE Trans Fuzzy Syst 16(1):13–28. https://doi.org/10.1109/TFUZZ.2007.902038
38. Li J, Yang L, Fu X, Chao F, Qu Y (2018) Interval Type-2 TSK+ fuzzy inference system. In: 2018 IEEE international conference on fuzzy systems, Rio de Janeiro, Brazil, 8–13 July 2018
39. Yang L, Shen Q (2010) Adaptive fuzzy interpolation and extrapolation with multiple-antecedent rules. In: Proceedings of the 2010 IEEE international conference on fuzzy systems. Curran Associates, Red Hook, NY, USA. https://doi.org/10.1109/FUZZY.2010.5584701
40. Naik N, Diao R, Quek C, Shen Q (2013) Towards dynamic fuzzy rule interpolation. In: Proceedings of the 2013 IEEE international conference on fuzzy systems. IEEE, New York. https://doi.org/10.1109/FUZZ-IEEE.2013.6622404
41. Naik N, Diao R, Shen Q (2014) Genetic algorithm-aided dynamic fuzzy rule interpolation. In: Proceedings of the 2014 IEEE international conference on fuzzy systems. IEEE, New York, pp 2198–2205. https://doi.org/10.1109/FUZZ-IEEE.2014.6891816
42. Shen Q, Yang L (2011) Generalisation of scale and move transformation-based fuzzy interpolation. J Adv Comput Intell Int Inf 15(3):288–298. https://doi.org/10.20965/jaciii.2011.p0288
43. Yang L, Chao F, Shen Q (2017) Generalised adaptive fuzzy rule interpolation. IEEE Trans Fuzzy Syst 25(4):839–853. https://doi.org/10.1109/TFUZZ.2016.2582526
44. Yang L, Chen C, Jin N, Fu X, Shen Q (2014) Closed form fuzzy interpolation with interval type-2 fuzzy sets. In: Proceedings of the 2014 IEEE international conference on fuzzy systems. IEEE, pp 2184–2191. https://doi.org/10.1109/FUZZ-IEEE.2014.6891643
45. Yang L, Shen Q (2011) Adaptive fuzzy interpolation. IEEE Trans Fuzzy Syst 19(6):1107–1126. https://doi.org/10.1109/TFUZZ.2011.2161584

46. Yang L, Shen Q (2011) Adaptive fuzzy interpolation with uncertain observations and rule base. In: Lin C-T, Kuo Y-H (eds) Proceedings of the 2011 IEEE international conference on fuzzy systems. IEEE, New York, pp 471–478. https://doi.org/10.1109/FUZZY.2011.6007582
47. Yang L, Shen Q (2013) Closed form fuzzy interpolation. Fuzzy Sets Syst 225:1–22. https://doi.org/10.1016/j.fss.2013.04.001
48. Li J, Yang L, Fu X, Chao F, Qu Y (2017) Dynamic QoS solution for enterprise networks using TSK fuzzy interpolation. In: Proceedings of the 2017 IEEE international conference on fuzzy systems. IEEE, New York. https://doi.org/10.1109/FUZZ-IEEE.2017.8015711
49. Li J, Yang L, Shum HP, Sexton G, Tan Y (2015) Intelligent home heating controller using fuzzy rule interpolation. In: UK workshop on computational intelligence, 7–9 September 2015, Exeter, UK
50. Naik N (2015) Fuzzy inference based intrusion detection system: FI-Snort. In: Wu Y, Min G, Georgalas N, Hu J, Atzori L, Jin X, Jarvis S, Liu L, Calvo RA (eds) Proceedings of the 2015 IEEE international conference on computer and information technology; Ubiquitous computing and communications; Dependable, autonomic and secure computing; Pervasive intelligence and computing. IEEE Computer Society, Los Alamitos, CA, USA, pp 2062–2067. https://doi.org/10.1109/CIT/IUCC/DASC/PICOM.2015.306
51. Yang L, Li J, Hackney P, Chao F, Flanagan M (2017) Manual task completion time estimation for job shop scheduling using a fuzzy inference system. In: Wu Y, Min G, Georgalas N, Al-Dubi A, Jin X, Yang L, Ma J, Yang P (eds) Proceedings of the 2017 IEEE international conference on internet of things (iThings) and IEEE green computing and communications (GreenCom) and IEEE cyber, physical and social computing (CPSCom) and IEEE smart data (SmartData). IEEE Computer Society, Los Alamitos, CA, USA, pp 139–146. https://doi.org/10.1109/iThings-GreenCom-CPSCom-SmartData.2017.26
52. Li J, Qu Y, Shum HPH, Yang L (2017) TSK inference with sparse rule bases. In: Angelov P, Gegov A, Jayne C, Shen Q (eds) Advances in computational intelligence systems. Springer, Cham, pp 107–123. https://doi.org/10.1007/978-3-319-46562-3_8
53. Guha S, Yau SS, Buduru AB (2016) Attack detection in cloud infrastructures using artificial neural network with genetic feature selection. In: Proceedings of the 2016 IEEE 14th international conference on dependable, autonomic and secure computing, 14th international conference on pervasive intelligence and computing, 2nd international conference on big data intelligence and computing and cyber science and technology congress. IEEE Computer Society, Los Alamitos, CA, USA, pp 414–419. https://doi.org/10.1109/DASC-PICom-DataCom-CyberSciTec.2016.32
54. Jensen R, Shen Q (2008) Computational intelligence and feature selection: rough and fuzzy approaches. Wiley-IEEE Press, New York
55. Jensen R, Shen Q (2009) New approaches to fuzzy-rough feature selection. IEEE Trans Fuzzy Syst 17(4):824–838. https://doi.org/10.1109/TFUZZ.2008.924209
56. Tsang EC, Chen D, Yeung DS, Wang XZ, Lee JW (2008) Attributes reduction using fuzzy rough sets. IEEE Trans Fuzzy Syst 16(5):1130–1141. https://doi.org/10.1109/TFUZZ.2006.889960
57. Zuo Z, Li J, Anderson P, Yang L, Naik N (2018) Grooming detection using fuzzy-rough feature selection and text classification. In: 2018 IEEE international conference on fuzzy systems, Rio de Janeiro, Brazil, 8–13 July 2018
58. Dash M, Liu H (1997) Feature selection for classification. Intell. Data Anal 1(3):131–156. https://doi.org/10.1016/S1088-467X(97)00008-5
59. Langley P (1994) Selection of relevant features in machine learning. In: Proceedings of the AAAI fall symposium on relevance. AAAI Press, Palo Alto, CA, USA, pp 245–271
60. Jensen R, Shen Q (2009) Are more features better? A response to attributes reduction using fuzzy rough sets. IEEE Trans Fuzzy Syst 17(6):1456–1458. https://doi.org/10.1109/TFUZZ.2009.2026639
61. Guyon I, Elisseeff A (2003) An introduction to variable and feature selection. J Mach Learn Res 3:1157–1182. http://www.jmlr.org/papers/volume3/guyon03a/guyon03a.pdf

62. Jensen R, Shen Q (2004) Semantics-preserving dimensionality reduction: rough and fuzzy-rough-based approaches. IEEE Trans Knowl Data Eng 16(12):1457–1471. https://doi.org/10.1109/TKDE.2004.96
63. Parthaláin NM, Shen Q (2009) Exploring the boundary region of tolerance rough sets for feature selection. Pattern Recogn 42(5):655–667. https://doi.org/10.1016/j.patcog.2008.08.029
64. Parthaláin NM, Shen Q, Jensen R (2010) A distance measure approach to exploring the rough set boundary region for attribute reduction. IEEE Trans Knowl Data Eng 22(3):305–317. https://doi.org/10.1109/TKDE.2009.119
65. Saeys Y, Inza I, Larrañaga P (2007) A review of feature selection techniques in bioinformatics. Bioinformatics 23(19):2507–2517. https://doi.org/10.1093/bioinformatics/btm344
66. Yu L, Liu H (2004) Efficient feature selection via analysis of relevance and redundancy. J Mach Learn Res 5:1205–1224
67. Thorndike RL (1953) Who belongs in the family? Psychometrika 18(4):267–276. https://doi.org/10.1007/BF02289263
68. Anderson JA (1995) An introduction to neural networks. MIT Press, Cambridge, MA, USA
69. Planquart J-P (2001) Application of neural networks to intrusion detection. Sans Institute. https://www.sans.org/reading-room/whitepapers/detection/application-neural-networks-intrusion-detection-336
70. Cameron R, Zuo Z, Sexton G, Yang L (2017) A fall detection/recognition system and an empirical study of gradient-based feature extraction approaches. In: Chao F, Schockaert S, Zhang Q (eds) Advances in computational intelligence systems. Springer, Cham, pp 276–289. https://doi.org/10.1007/978-3-319-66939-7_24
71. Linda O, Vollmer T, Manic M (2009) Neural network based intrusion detection system for critical infrastructures. In: Proceedings of the 2009 international joint conference on neural networks. IEEE, Piscataway, NJ, USA, pp 1827–1834. https://doi.org/10.1109/IJCNN.2009.5178592
72. Subba B, Biswas S, Karmakar S (2016) A neural network based system for intrusion detection and attack classification. In: Proceedings of the twenty second national conference on communication. IEEE, New York. https://doi.org/10.1109/NCC.2016.7561088
73. Zuo Z, Yang L, Peng Y, Chao F, Qu Y (2018) Gaze-informed egocentric action recognition for memory aid systems. IEEE Access 6:12894–12904. https://doi.org/10.1109/ACCESS.2018.2808486
74. Beghdad R (2008) Critical study of neural networks in detecting intrusions. Comput Secur 27(5):168–175. https://doi.org/10.1016/j.cose.2008.06.001
75. Ouadfel S, Batouche M (2007) Antclust: an ant algorithm for swarm-based image clustering. Inf Technol J 6(2):196–201. https://doi.org/10.3923/itj.2007.196.201
76. De la Hoz E, de la Hoz E, Ortiz A, Ortega J, Martínez-Álvarez A: Feature selection by multi-objective optimisation: application to network anomaly detection by hierarchical self-organising maps. Knowl Based Syst 71:322–338. https://doi.org/10.1016/j.knosys.2014.08.013
77. Labib K, Vemuri R (2002) NSOM: a real-time network-based intrusion detection system using self-organizing maps. http://web.cs.ucdavis.edu/~vemuri/papers/som-ids.pdf
78. Vasighi M, Amini H (2017) A directed batch growing approach to enhance the topology preservation of self-organizing map. Appl Soft Comput 55:424–435. https://doi.org/10.1016/j.asoc.2017.02.015
79. Vokorokos L, Balaz A, Chovanec M (2006) Intrusion detection system using self organizing map. Acta Electrotechnica et Informatica 6(1). http://www.aei.tuke.sk/papers/2006/1/Vokorokos.pdf
80. Prabhakar SY, Parganiha P, Viswanatham VM, Nirmala M (2017) Comparison between genetic algorithm and self organizing map to detect botnet network traffic. In: IOP conference series: materials science and engineering, vol 263. IOP Publishing, Bristol. https://doi.org/10.1088/1757-899X/263/4/042103
81. Haykin S (2009) Neural networks and learning machines, 3rd edn. Prentice Hall, Upper Saddle River, NJ, USA

82. Joo D, Hong T, Han I (2003) The neural network models for IDS based on the asymmetric costs of false negative errors and false positive errors. Expert Syst Appl 25(1):69–75. https://doi.org/10.1016/S0957-4174(03)00007-1
83. Patcha A, Park JM (2007) An overview of anomaly detection techniques: existing solutions and latest technological trends. Comput Netw 51(12):3448–3470. https://doi.org/10.1016/j.comnet.2007.02.001
84. Chiu SL (1994) Fuzzy model identification based on cluster estimation. J Intell Fuzzy Syst 2(3):267–278. https://doi.org/10.3233/IFS-1994-2306
85. Mahoney MV (2003) A machine learning approach to detecting attacks by identifying anomalies in network traffic. Ph.D. thesis, Florida Institute of Technology, Melbourne, FL, USA
86. Elisa N, Yang L, Naik N (2018) Dendritic cell algorithm with optimised parameters using genetic algorithm. In: 2018 IEEE congress on evolutionary computation, Rio de Janeiro, Brazil, 8–13 July 2018
87. Tavallaee M, Bagheri E, Lu W, Ghorbani A (2009) A detailed analysis of the KDD Cup 99 data set. In: Wesolkowski S, Abbass H, Abielmona R (eds) Proceedings of the 2009 IEEE symposium on computational intelligence for security and defense applications. https://doi.org/10.1109/CISDA.2009.5356528
88. Gharib A, Sharafaldin I, Lashkari AH, Ghorbani AA (2016) An evaluation framework for intrusion detection dataset. In: Joukov N, Kim H (eds) Proceedings of the 2016 international conference on information science and security. Curran Associates, Red Hook, NY, USA. https://doi.org/10.1109/ICISSEC.2016.7885840
89. Sharafaldin I, Lashkari AH, Ghorbani AA (2018) Toward generating a new intrusion detection dataset and intrusion traffic characterization. In: Mori P, Furnell S, Camp O (eds) Proceedings of the 4th international conference on information systems security and privacy, vol 1, pp 108–116. https://doi.org/10.5220/0006639801080116