Babylonian Journal of Networking
Vol.2025, **pp**. 14–24
DOI: https://doi.org/10.58496/BJN/2025/002; ISSN: 3006-5372
https://mesopotamian.press/journals/index.php/BJN

Research Article

Utilizing Artificial Intelligence in Cybersecurity: A Study of Neural Networks and Support Vector Machines

Akram Kadhim Abed[1,*]

[1] *Department of Computer and Communication Engineering - Informatics, Faculty of Engineering, Islamic University of Lebanon. Lebanon.*

ARTICLE INFO

Article History
Received 24 Dec 2024
Revised: 02 Jan 2025
Accepted 04 Feb 2025
Published 25 Feb 2025

Artificial Intelligence (AI)
Network Security
Multi-agent Systems
Production Rules
Bayes' Theorem
Neural Networks

ABSTRACT

The rapid evolution of network security threats necessitates the integration of advanced technologies, particularly artificial intelligence (AI), in the development of effective protection systems. This article reviews contemporary methodologies employing AI for enhancing computer network security, with a focus on neural networks (NN) and support vector machines (SVM). It begins by elucidating the architecture of neural networks, including the training and recognition phases essential for detecting malicious activities within a network. The effectiveness of NN in identifying patterns indicative of unauthorized access is highlighted, alongside the challenges associated with training datasets. Further, the article explores the application of SVM in classifying network traffic and detecting unwanted software through geometric interpretations of classification tasks. It also emphasizes the growing trend of AI technology in modern antivirus utilities and network security analysis programs, advocating for the integration of multi-layered protective measures that leverage AI's learning capabilities. Finally, the potential of AI methodologies to unveil new pathways for research and application in network security is discussed, underscoring the need for continued exploration of these promising technologies to safeguard digital infrastructures against evolving threats.

1. INTRODUCTION

In the modern era, computer networks have evolved into expansive systems connecting diverse programs and devices to exchange, store, and process information. These networks drive significant business opportunities, with many companies relying on online presences and collaborative tools to enhance productivity. However, the growing complexity of these networks, coupled with the emergence of sophisticated cyber threats, underscores the critical need for robust security measures [1].

The proliferation of malicious software—encompassing spyware, adware, and spam—further exacerbates the situation, as the development of such software transitions from amateur efforts to lucrative criminal enterprises [2]. To combat these challenges, systems capable of classifying and neutralizing malicious activities have been developed. Traditional security systems, however, are often limited by their reliance on predefined rules, leaving them vulnerable to novel attacks and unpatched vulnerabilities [3].

The application of artificial intelligence (AI) introduces self-learning capabilities into network security, enabling systems to autonomously adapt and detect threats in real time [4]. This paper explores AI-driven methodologies in cybersecurity, with a particular focus on neural networks (NN) and support vector machines (SVM).

The subsequent sections provide an in-depth analysis of key approaches. Section 2 discusses multi-agent systems and their applications in network vulnerability analysis [5]. Section 3 examines production systems, while Section 4 highlights the application of Bayes' theorem in spam detection [6]. Section 5 delves into neural networks and their role in anomaly detection [7]. Finally, Section 6 evaluates SVMs, a promising yet underutilized AI method in network security [8].

*Corresponding author. Email: akram.kadhim87@gmail.com

2. MULTI-AGENT SYSTEMS

2.1 General Overview

The evolution of cybersecurity has traditionally focused on personal protection tools, such as antivirus software and firewalls, designed to secure individual devices. However, the increasing complexity of networked environments has shifted research towards the development of multi-agent systems (MAS), which offer a more comprehensive approach to network security [5].

In a multi-agent system, each agent is responsible for a specific part of the security task, and the overall solution emerges from the coordinated actions of these agents [9]. While some agents are equipped with intelligence capabilities, others rely on basic functionality. Communication between agents occurs through specialized protocols, ensuring seamless operation. Typically, a managing agent oversees the activities of others, providing a structured yet flexible architecture.

Key characteristics of multi-agent systems that make them highly effective for network security include:

1. *Flexibility:* Agents can replicate and deploy themselves on new network nodes, enabling multi-agent systems to adapt to changing network architectures and hardware configurations. This adaptability ensures continuous security, even in dynamic environments [5].
2. *Cost-effectiveness:* By distributing the system evenly across the network perimeter, MAS optimize the use of computational resources, reducing the strain on individual nodes [9].
3. *Increased Fault Tolerance:* Unlike centralized systems, MAS have no single point of failure. This decentralized structure complicates attackers' efforts, as compromising the network requires simultaneous targeting of multiple nodes [10].
4. *Centralized Administration Capability:* Despite their distributed nature, MAS allow centralized updates and configuration changes, which are communicated to all agents through their protocol infrastructure. This capability balances autonomy and control, ensuring effective management of the network [11].

The multi-agent approach has proven effective in safeguarding networks against a variety of threats. It is particularly well-suited for handling large, distributed environments where centralized systems may struggle to maintain efficiency and resilience [5] [9].

2.2 Network Security Systems

Multi-agent systems (MAS) have been extensively researched for their effectiveness in protecting networks from external threats. Their architecture, as explored by Zhang and Paxson and Shafiq et al., is designed to handle complex cybersecurity challenges while providing a clear visualization of the state of the network and its constituent agents.

Agents in a multi-agent system are categorized based on their functions and organized into specialized teams. For instance, Shafiq et al. classify agents into the following categories:

1. *Information Processing Agents (Samplers):* Responsible for gathering data, which is then analyzed to detect anomalies or potential misuse.
2. *Attack Detection Agents (Detectors):* Analyze data provided by samplers to identify unusual activities or security breaches.
3. *Filtering Agents (Filters):* Apply rules specified by detectors to filter out malicious traffic effectively.
4. *Investigation Agents:* Tasked with neutralizing the identified malicious agents by taking targeted action.

These agents collaborate through sophisticated interaction schemes to implement a comprehensive security mechanism. For example, when an attack is detected, the detector in the affected network initiates action by requesting relevant data from samplers across the network. This collaborative data exchange significantly enhances the likelihood of identifying and mitigating threats. Once the attack's origin is determined, the detector sends this information to its counterpart in the malicious agent's network for deactivation.

The MAS framework provides several advantages, as demonstrated in the research of Zhang and Paxson and Wang, Islam, and Jajodia:

- *Distributed Resilience:* The decentralized nature of MAS ensures that no single point of failure exists, making it challenging for attackers to compromise the entire system.
- *Efficient Resource Utilization:* Agents optimize the use of network resources by distributing tasks across multiple nodes.
- *Scalable Collaboration:* Teams of agents coordinate efficiently to manage large-scale attacks or complex multi-stage threats.

By integrating MAS with intelligent data-processing algorithms, organizations can significantly enhance the overall effectiveness of their network security systems. This collaborative and adaptive approach has proven to be a robust solution for modern cybersecurity challenges [11].

2.3 Vulnerability Analysis Systems

A critical component of any computer network security system is the regular assessment of its vulnerabilities. Depending on the required quality and depth of the assessment, two primary methods—scanning and probing—are employed [10] [1].

1. Scanning

Scanning is a passive analysis method designed to identify vulnerabilities without initiating any attacking actions. Initially, the system collects information about the current state of the network, including open ports and associated data blocks, such as headers and banners that contain standard responses from network services. This information is then compared against a database of known vulnerabilities to determine whether any security gaps exist.

Recent advancements in scanning techniques leverage knowledge of the physical topology of network devices. For example, scanning tools can construct a network graph during the information-gathering phase. These tools generate an attack graph that models the network, helping to assess potential vulnerabilities and enabling predictions about possible multi-step complex attacks [10]. Expert systems and intelligent tools are often used to analyze these graphs, providing deeper insights into the network's security posture [3].

2. Probing

Probing is an active analysis method that simulates attacks on the system being assessed. It often follows scanning and uses the attack graph created during the scanning phase. While probing is slower and more complex than scanning, it delivers more precise results, identifying vulnerabilities that passive methods might miss. For example, probing can detect vulnerabilities exploited in DDoS (Distributed Denial of Service) attacks, which involve multiple malicious nodes distributed across various subnets [9].

Probing systems can also employ the multi-agent approach, which facilitates simulation modeling of interactions between the defense system and the simulated attack at the packet level. This simulation-based method offers several advantages:

- *Cost Efficiency:* Reduces expenses by eliminating the need for real networks during testing.
- *Adaptability:* Simplifies testing under new conditions by allowing the modification of agent scenarios for emerging distributed attack types [5] [11].

2.4 Application of Multi-Agent Systems

Multi-agent approaches are integral to both scanning and probing systems. In the context of probing, agents can simulate attacker and defender behaviors, enhancing the accuracy of attack modeling and defense evaluation. For instance, attack modeling systems employing multi-agent frameworks can efficiently simulate DDoS attacks and other complex threats, enabling the development of robust mitigation strategies [12] [13].

Vulnerability analysis systems and their methodologies are comprehensively explored in [1] and [14], while the integration of multi-agent frameworks for attack modeling is detailed in [15] and [16].

3. PRODUCTION SYSTEMS

Production systems are rule-based systems where decisions are made based on predefined conditions. These systems use rules in the form of:

- **IF <condition> THEN <action>**

The left-hand side, <condition>, specifies the criteria that must be satisfied for the rule to apply. These conditions can combine multiple simple statements using logical operators such as AND, OR, and NOT. The right-hand side, <action>, defines the operation to be executed if the conditions are met.

3.1 Heuristics and Expert Systems

Production systems often serve as heuristics—rules derived from experience that do not require exhaustive data but can still produce practical solutions. These heuristics, commonly used in expert systems, are built from years of domain-specific knowledge [4] [17]. Expert systems, also referred to as knowledge-based systems, leverage this expertise to address problems in a range of fields, including cybersecurity.

3.2 Components of a Production System

Any production system comprises three fundamental components [7]:

1. **Knowledge Base**:
   - Contains domain-specific rules and facts.
   - Serves as the repository for decision-making logic.
2. **Working Memory**:
   - Temporarily stores information about the current problem.
   - Allows the system to process and adapt dynamically based on new data inputs.
3. **Inference Mechanism**:

- Enables the application of rules to the data in working memory.
- Determines the sequence of actions or conclusions drawn by the system.

The structure of a production system is illustrated in Figure 1.

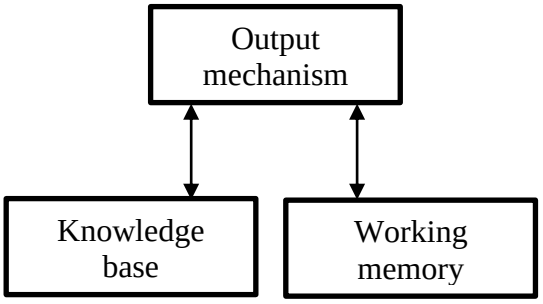

Fig. 1. Structure of a Production System

3.3 Applications in Cybersecurity

Production systems are widely used in cybersecurity for tasks such as intrusion detection and vulnerability assessment. For example, rules in an intrusion detection system may include conditions like:
"IF unexpected traffic on port 80 AND packet size exceeds threshold THEN flag as suspicious activity."
By codifying such rules, production systems provide a logical and adaptable framework for identifying potential threats. While their reliance on predefined rules can limit flexibility, their simplicity and clarity make them essential tools in many cybersecurity applications [18].

- **Challenges and Future Directions**

Despite their utility, production systems face challenges in addressing dynamic and evolving cyber threats. Their dependency on static, predefined rules can make them less effective against novel attack methods. Integrating machine learning techniques, such as neural networks or support vector machines, with production systems may enhance their adaptability and real-time decision-making capabilities [7].

- **Production Systems in Network Security**

In a production system, the knowledge base stores rules (productions) that describe the subject area, while the working memory dynamically holds facts relevant to the current situation. The contents of the working memory evolve as rules are applied: facts are either added or, less frequently, removed. The inference mechanism enables logical reasoning, matching rules with facts and modifying the working memory accordingly [4] [17].
When a production's conditions are satisfied, one of two events occurs:

1. *Acquisition of New Knowledge:* A fact from the rule's right-hand side is added to the working memory.
2. *Execution of an Action:* A specific operation, such as changing a network configuration, is performed.

- **Application in Network Security**

Production systems are widely used in network security for detecting known vulnerabilities based on formal indicators established by experts. Developers regularly update the heuristic databases of these systems to ensure they remain effective against emerging threats [1]. For example, the free utility AVZ includes a heuristic system check feature that detects spyware and viruses by analyzing indirect indicators such as registry entries, disk files, and memory usage [2].

- **Example Production Rule**

IF

- A process uses libraries for network operations, AND
- The number of detected signatures typical for sending emails exceeds a threshold,

THEN

- Record in the working memory the fact that "the program is working with email."

This example illustrates how a composite condition can generate new facts for further processing by other rules, enhancing the system's ability to detect and classify network activities.

- **Advantages and Limitations**

❖ **Advantages**:

- *Resilience:* Production systems are robust against known threats, offering accurate detection based on expert-curated rules [18].

- *High Accuracy:* These systems excel in identifying well-documented vulnerabilities, such as those in outdated software or commonly exploited protocols [7].

❖ **Limitations**:

- *Zero-Day Vulnerability:* Production systems struggle to address zero-day attacks, which exploit newly discovered vulnerabilities that remain unpatched.
- *Frequent Updates:* The increasing volume of threats requires constant updates to the rule set by experts, making the process labor-intensive and reactive rather than proactive [3].

- **Future Directions**

To overcome these challenges, integrating production systems with machine learning methods, such as neural networks and support vector machines, could enhance adaptability and allow these systems to identify patterns beyond predefined rules. For instance, AI-powered production systems could dynamically adapt to new threats, minimizing reliance on static heuristics [4] [17].

4. BAYES' THEOREM

Another approach to handling incompletely defined information is probabilistic modeling of the subject area. In this field, systems based on Bayes' Theorem have gained widespread popularity. The theorem is articulated through Bayes' formula:

$$P(H \mid X) = (P(X \mid H)P(H))/P(X) \quad (1)$$

Where:
P(H/X) represents the probability of hypothesis *H* given the occurrence of cause *X*;
P(X/H) denotes the probability of the presence of cause *X* assuming that hypothesis *H* is true;
P(H) indicates the prior probability of hypothesis *H*;
and *P(X)* signifies the probability of the occurrence of cause *X*.

This fundamental formula serves as the foundation for many contemporary artificial intelligence systems designed to operate under conditions of uncertainty [7] [4]. Such systems provide probabilistic assessments and typically do not replace human experts; instead, they assist in decision-making processes.
In practice, when there are *n* hypotheses, Bayes' Theorem is utilized in its general form:

$$P(H_i|X) = \frac{P(X|H_i)P(H_i)}{\sum_{k=1}^{n} P(X|H_k)P(H_k)} \quad (2)$$

Where:
$P(H_i/X)$ represents the probability of the truth of hypothesis H_i given the specified cause *X*;
$P(H_i)$ denotes the prior probability of hypothesis H_i;
$P(X/H_i)$ indicates the probability of the presence of cause *X* if hypothesis H_i is true;
and *n* represents the number of possible hypotheses.
If the cause can be represented as a vector:

$$\mathbf{X} = (X_1, X_2, \dots, X_m),$$

Each component of which has a conditional probability relative to the hypothesis Hi, denoted as $P(\mathbf{X}_j/H_i)$. To compute the conditional probabilities $P(X/H_i)$, the "naive" (*Classification systems based on this assumption are called Naive Bayes classifiers*) assumption of conditional independence among the components of vector **X** is employed. In this case, the conditional probability is calculated using the formula [19]:

$$P(\mathbf{X}|H_i) = \prod_{j=1}^{m} P(X_j|H_i). \quad (3)$$

Consider an example of a spam filter (*A spam filter is a type of email message filtering that screens out unwanted emails, most often of an advertising nature*) based on Bayes' Theorem [6]. During the training phase, a collection of electronic messages is divided into two classes: spam and legitimate correspondence. For each word, the frequency of its occurrence in both classes of emails is computed.
Let $F_S(W_i)$ denote the number of spam emails containing the word W_i, and $F_{NS}(W_i)$ represent the number of legitimate emails containing the word W_i. In this scenario, two hypotheses are present: H_S (the email is spam) and H_{NS} (the email is legitimate). Thus, the probability that the occurrence of the word W_i in an email indicates spam can be calculated using the formula:

$$P(W_i|H_S) = \frac{F_S(W_i)}{F_S(W_i) + F_{NS}(W_i)}, \quad (4)$$

The probability that the word W_i does not indicate spam in an email is given by:

$$P(W_i|H_{NS}) = \frac{F_{NS}(W_i)}{F_S(W_i) + F_{NS}(W_i)}. \quad (5)$$

The vector *W* includes all the words in a new email. Therefore, the probability that the new email is spam can be calculated using Bayes' theorem as follows:

$$P(H_S|\mathrm{W}) = \frac{P(\mathrm{W}|H_S)P(H_S)}{P(\mathrm{W}|H_S)P(H_S) + P(\mathrm{W}|H_{NS})P(H_{NS})}. \quad (6)$$

Considering formula (1) and assuming that the prior probabilities of both hypotheses are equal, we obtain:

$$P(H_S|\mathrm{W}) = \frac{\prod_{j=1}^{m} P(W_i|H_S)}{\prod_{j=1}^{m} P(W_j|H_S) + \prod_{j=1}^{m} P(W_j|H_{NS})}. \quad (7)$$

Bayes' Theorem forms the foundation of many modern spam filters by applying probabilistic reasoning to classify emails as either spam or legitimate. Classification is typically based on a user-defined threshold, often set between **0.6 and 0.8**. Once a classification decision is made, the probabilities associated with the words contained in the email are updated in the system's database [6].

4.1 Effectiveness of Bayesian Spam Filters

Bayesian spam filters are known for their straightforward implementation and efficiency. After being trained on a sufficiently large dataset of emails, these filters can effectively identify and block up to 95–97% of spam messages. Moreover, they are adaptable, allowing retraining as new types of spam emerge. This flexibility has made Bayesian methods a mainstay in modern email filtering systems [4].

4.2 Challenges in Modern Spam Detection

Despite their success, traditional Bayesian filters face challenges from evolving spam techniques. For instance, spammers often embed advertising content within images, accompanied by nonsensical or absent text. This bypasses text-based filters, requiring more advanced tools like optical character recognition (OCR) to extract textual information from images. Alternatively, older methods, such as blacklists and regular expressions, remain useful since these types of spam often follow stereotypical patterns (*Kaspersky Lab, 2021*).

Kaspersky Lab has developed technologies that recognize text within embedded images, enabling the extracted content to be analyzed by Bayesian filters. Such advancements illustrate the ongoing evolution of spam filtering technologies [2].

4.3 Bayesian Networks: An Evolution of Bayes' Theorem

An advanced application of Bayes' Theorem is the Bayesian network, which models probabilistic and causal relationships among variables. Structurally, a Bayesian network is a directed graph where each node contains specific probability values, representing the joint probability distribution. These networks provide a visual and analytical framework for complex probabilistic reasoning [20].

To create a functional Bayesian network, the model must be trained on expertly prepared datasets. During training, algorithms such as gradient descent and the Expectation-Maximization (EM) algorithm are employed to minimize errors and optimize the network's performance. These methods ensure the network's reliability in real-world applications [4] [20].

4.4 Practical Implementation Challenges and Real-World Applicability

The methods presented in this study—Bayes' Theorem, Bayesian networks, and heuristic-based production systems—offer significant potential for enhancing cybersecurity. However, their practical implementation faces certain challenges that must be addressed for effective real-world deployment.

4.5 Practical Implementation Challenges

1. Scalability and Computational Resources

Bayesian networks and production systems require significant computational power for processing large datasets, particularly in real-time scenarios. For example, the inference mechanisms in Bayesian networks involve calculating probabilities across multiple nodes, which can be resource-intensive for large-scale networks [20].

2. Adaptability to Evolving Threats

While production systems excel in detecting known vulnerabilities, they struggle against zero-day attacks, which exploit unpatched vulnerabilities in software. Regular updates to the knowledge base are necessary, but this process is labor-intensive and may not keep pace with rapidly emerging threats [3].

Similarly, traditional Bayesian filters, though effective against textual spam, require additional tools like optical character recognition (OCR) to handle image-based spam or other novel evasion techniques used by attackers [21].

3. Training Data and Expertise Requirements

Bayesian networks require expertly curated datasets for training to minimize errors during operation. Preparing such datasets is a time-consuming process, and errors during training can reduce the network's effectiveness (*Russell & Norvig, 2021*). Moreover, the use of advanced algorithms like gradient descent and Expectation-Maximization (EM) adds complexity to the training process.

4. Integration with Legacy Systems
Incorporating these AI-driven methods into existing network infrastructures presents compatibility challenges. Legacy systems may lack the processing power or architectural flexibility needed to support advanced AI techniques like Bayesian networks or production systems with dynamic heuristics [1].

4.6 Real-World Applicability

Despite these challenges, the presented methods have proven applicability in addressing various cybersecurity problems:
1. Spam Detection and Filtering
Bayesian filters remain a cornerstone in email filtering, capable of blocking up to 95–97% of spam messages with high accuracy. The addition of OCR technology and updated rule sets extends their utility to modern threats, such as image-based spam [6] [2].
2. Intrusion Detection
Production systems are widely used for intrusion detection by applying heuristic rules to monitor and identify suspicious activities. For example, heuristic-based systems can flag processes using libraries for unusual network operations or a high volume of email signatures as potential threats [18].
3. Decision Support Systems
Bayesian networks provide robust support for decision-making in complex environments by modeling probabilistic and causal relationships. Their applications include assessing risk levels in real-time and predicting potential attack vectors based on historical data [3].
4. Multi-Layered Defense Strategies
By combining heuristic-based production systems with machine learning models like neural networks and support vector machines, organizations can create adaptive, multi-layered defense mechanisms capable of handling both known and emerging threats [3].

4.7 Future Considerations

To enhance real-world applicability, the following strategies can be adopted:

- *Hybrid Systems:* Integrating machine learning techniques, such as neural networks, with Bayesian methods to improve adaptability and threat detection.
- *Cloud-Based Deployment:* Utilizing cloud infrastructure to overcome computational constraints and improve scalability.
- *Automation:* Leveraging automated tools for dataset preparation and rule updates to reduce dependency on expert input.
- *Cross-System Integration:* Developing frameworks that facilitate seamless integration with legacy systems for faster deployment.

5. ARTIFICIAL NEURAL NETWORKS

An artificial neural network (ANN) serves as a simplified model of the brain and consists of a set of neurons interconnected in a specific manner [4] [22].
Neural networks are adept at addressing various practical tasks, primarily those related to pattern recognition and classification. One of the notable advantages of ANNs is their ability to autonomously acquire knowledge during the training process, as well as their capacity for generalization.
The fundamental component of the network is the artificial neuron (see Fig. 2), which represents a mathematical model of a biological nerve cell.

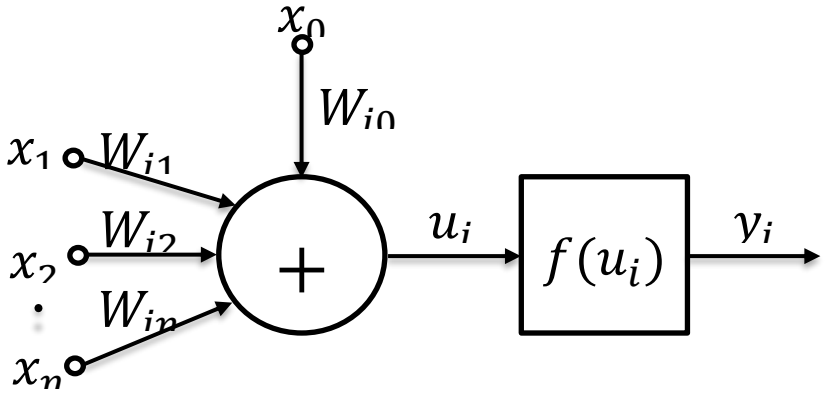

Fig. 2. Neuron Model

In this model, the input signals x_j (where j=1, 2, …, n, are summed within the *i-th* neuron, taking into account the corresponding weight coefficients w_{ij}. Additionally, the sum includes x_0— the bias term—which determines the reduction or increase of the input signal by a specified amount.
The cumulative value is then fed into a functional block $f(u_i)$, the output of which represents the output signal of the neuron. Thus, the operation of the *i-th* neuron can be described by the following function:

$$f(u_i) = f\left(\sum_{j=1}^{n} w_{ij}x_j + x_0\right) \tag{8}$$

This function illustrates the fundamental process of how the neuron computes its output based on the weighted sum of its inputs, adjusted by the bias.

$$y_{i=f\left(\sum_{j=1}^{n} w_{ij}x_j + w_{i0}x_o\right)} \tag{9}$$

Based on the form of the function $f(u_i)$, referred to as the activation function, several types of neurons can be distinguished. The most commonly used type is the sigmoid neuron, whose activation function is defined as follows:

$$f(u_i) = \frac{1}{1 + e^{-u_i}} \tag{10}$$

This function maps the input u_i to an output value between 0 and 1, making it particularly suitable for binary classification tasks. The sigmoid function's smooth gradient allows for effective learning in neural networks, facilitating the adjustment of weights during training through gradient descent methods.

$$f(u) = \frac{1}{1 + e^{-\beta u}} \tag{11}$$

Individual neurons are organized into networks with diverse architectures. Currently, multilayer feedforward networks are widely used. In these networks, the outputs of neurons in one layer serve as inputs for the subsequent layer (see Figure 3).
Multilayer feedforward networks consist of an input layer, one or more hidden layers, and an output layer. The input layer receives the initial data, while the hidden layers process this information through the activation functions of their respective neurons. Finally, the output layer produces the network's predictions or classifications. This architecture allows for complex mappings between inputs and outputs, enabling the networks to learn intricate patterns in data.

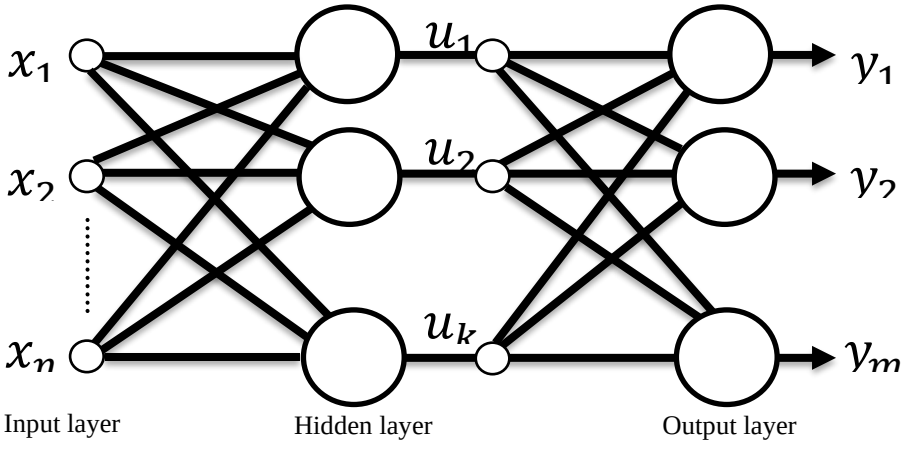

Fig. 3. Two-Layer Neural Network

The application of neural networks (NN) for solving any task encompasses two stages: the training phase and the recognition phase. During the training phase, a training dataset consisting of pre-selected and prepared input and output vectors is fed into the NN. Depending on the chosen learning algorithm (for instance, the backpropagation method or the conjugate gradient method), the weight coefficients are adjusted. As a result, when the NN receives a training vector as input, it produces a specified output vector that indicates the class of the input vector.
In the recognition phase, an unknown input vector is presented to the NN. The output of this phase is a vector representing the recognition result, which categorizes the input vector into one of the known classes.
Thus, when utilizing NNs in the domain of network security, any action performed by a user or an application must be represented as a feature vector, which is inputted into the NN. As the signals traverse through the network, the output yields a vector that determines whether the action is malicious.

Let us examine the application of NNs in a small practical task within the field of information security (example adapted from [23]. The objective is to detect access to a database by software that is distinct from the user's automated workstation (AW), or to identify anomalies in the user's behavior. The NN has four inputs:

1. *Volume of Information:* Measured in kilobytes, this represents the amount of data retrieved from the database during a control period. The obtained value must be normalized since the amount read from the database is not known in advance and varies for each task and user. Normalization can be achieved by evaluating the traffic on a scale of 1 to 10 (where 0 represents zero volume and 10 signifies maximum traffic volume).
2. *Transaction Count:* This refers to the number of transactions processed per minute.
3. *Data Modification Operations:* This is the count of data modification operations per minute. In this example, the AW utilizes "short transactions," typically involving 1–2 data modification operations within a single transaction.
4. *Dictionary Access Indicators:* Most client workstations do not access the database dictionary, distinguishing them from development and administration tools. These indicators will be binary (0 – no access, 1 – access) and will be several in number, with one for each of the database dictionary tables.

This structured approach allows the NN to effectively discern patterns and detect potentially anomalous activities associated with database access, thereby enhancing the security of the information system.

In this example, a two-layer feedforward neural network is utilized, comprising one hidden layer with two neurons and an output layer with a single neuron. The training of the neural network can be accomplished using existing software packages (e.g., Deductor Lite (*BaseGroup Labs*); MATLAB Neural Network Toolbox (*The MathWorks*) or well-known algorithms (e.g., the backpropagation method). To achieve quality training, approximately 300 training examples are necessary [23]. It is important to note that preparing the training dataset is a complex step in the overall process.

The output of the neural network can be interpreted as a percentage indicating the similarity of current actions to those typically performed by a hacker. This methodology can also be applied to identify various types of attacks and adapt to new threat vectors.

Another example of utilizing neural networks in network security systems is the neuro analyzer included in the AVZ antivirus utility [2]. The neuro analyzer facilitates the examination of suspicious files and is employed in detecting keyloggers.

The adoption of neural network technologies enhances security systems' learning capabilities and provides high recognition accuracy. However, a drawback lies in the complexity of analysis, resulting in the trained neural network being perceived by users as a "black box" with a specified number of inputs and outputs.

In contrast to production systems, the storage of a neural network on a computational machine requires significantly less memory, and determining malicious actions demands fewer computational resources. The impact of these advantages is further amplified when considering that developers strive to minimize the size of updates for their security systems.

6. SUPPORT VECTOR METHOD

The Support Vector Machine (SVM) method was described in the works of V. N. Vapnik [8] [24]. SVM is a mathematical approach for deriving a function that addresses classification tasks.

The concept of this method originated from the geometric interpretation of the classification problem. Suppose we have two sets of points that can be separated by a hyperplane (or a line in a two-dimensional space). There exists an infinite number of such hyperplanes (see Figure 4a).

To determine the optimal hyperplane, we select one such that the distances from the closest points of both classes to the hyperplane are equal (see Figure 4b). The nearest points, which act as vectors, are referred to as support vectors. The search for the optimal hyperplane translates into a quadratic programming problem subject to a set of linear inequality constraints.

In the 1990s, the SVM method was enhanced with the development of efficient algorithms for finding the optimal hyperplane, as well as methods for generalizing it to nonlinear cases and situations involving more than two classes [8] [24].

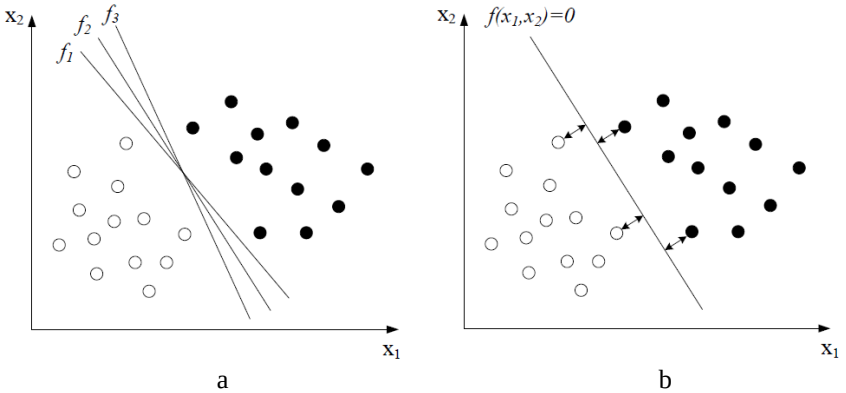

Fig. 4. Illustration of the Fundamental Concept of Support Vector Machines (SVM)

The Support Vector Machine (SVM) method has proven effective in various applications, including handwritten text recognition, face recognition, and text classification tasks. Current research is exploring the use of this method in network security systems. For instance, the methodology for identifying unwanted software based on the metric distance from the geometric center of feature vectors in computer network events using SVM is detailed in reference [25].
To detect attacks, it is essential to formulate a feature vector similar to the one created for artificial neural networks. Subsequently, the SVM classifier can be trained using specialized software, such as SVM Light [26]. The outcome will be a function capable of classifying feature vectors, thereby recognizing whether the current action of software or a user fall into a legitimate or prohibited class.
The methods for employing and training SVMs in the realm of network security remain inadequately explored. However, it is evident that this approach possesses significant power and promising potential for development, particularly in the context of safeguarding computer networks.

7. CONCLUSION

The landscape of network security is undergoing a significant transformation, driven by the increasing complexity and frequency of cyber threats. This article has examined the critical role of artificial intelligence (AI) technologies, particularly neural networks (NN) and support vector machines (SVM), in bolstering the effectiveness of network security systems. The findings indicate that these methodologies not only enhance the detection and classification of malicious activities but also offer adaptive learning capabilities that can be crucial in responding to emerging threats
Neural networks demonstrate remarkable proficiency in recognizing patterns in vast datasets, enabling the identification of anomalous behavior indicative of potential security breaches. However, the complexity of preparing training datasets remains a significant challenge that requires ongoing attention. Similarly, support vector machines have shown promise in classifying network activities by employing advanced mathematical techniques, thus facilitating the identification of unwanted software.
The integration of AI into new antivirus utilities and network security frameworks points to a growing trend towards multi-layered defense mechanisms that leverage the learning and adaptive capabilities of AI. As organizations continue to face sophisticated cyber threats, the potential for AI methodologies to revolutionize network security practices is substantial. It is imperative for future research to focus on refining these technologies and exploring their applicability in diverse contexts within network security.
In summary, the approaches and techniques discussed in this article highlight that the potential of AI in network security has not yet been fully realized. Continued research and development are essential to unlock new applications of AI methodologies, ensuring robust protection for computer networks in an increasingly hostile digital environment..

Conflicts Of Interest

The author disclosure statement confirms the absence of any conflicts of interest.

Funding

No grant or sponsorship is mentioned in the paper, suggesting that the author received no financial assistance.

Acknowledgment

The author extends appreciation to the institution for their unwavering support and encouragement during the course of this research.

References

[1] K. Scarfone and P. Mell, Guide to Intrusion Detection and Prevention Systems (IDPS), National Institute of Standards and Technology (NIST), 2021.
[2] K. Lab., Kaspersky Security Bulletin: Predictions for 2023, 2022.
[3] M. Z. Alom and et al., A Comprehensive Review of Deep Learning-Based Intrusion Detection Systems, vol. 21(3), IEEE Communications Surveys & Tutorials, 2019, pp. 2891-2923.
[4] S. Russell and P. Norvig, Artificial Intelligence: A Modern Approach, vol. 4th ed., Pearson Education, 2021.
[5] J. Zhang and and Paxson, Detecting Stealthy Network Attacks Using Multi-Agent Systems, vol. 15, IEEE Transactions on Information Forensics and Security, 2020, p. 245–257.
[6] P. Graham, A Plan for Spam." Paul Graham's Essays", 2020..
[7] I. Goodfellow, Y. Bengio and A. Courville, Deep Learning, MIT Press, 2016.
[8] V. Vapnik, Statistical Learning Theory, New York: Wiley, 1998.
[9] M. Z. Shafiq and et al., A Multi-Agent Framework for Cyber Defense against Distributed Denial of Service (DDoS) Attacks, vol. 13(2), ACM Transactions on Autonomous and Adaptive Systems, 2018, pp. 1-27.
[10] L. Wang, T. Islam and S. Jajodia, Modeling Attack Graphs for Network Security Risk Assessment, vol. 15(4), IEEE Transactions on Dependable and Secure Computing, 2018, pp. 556-570.

[11] M. Omar and R. Oludare, AI-Driven Multi-Agent Systems for Network Intrusion Detection, vol. 5(1), Journal of Artificial Intelligence Research and Applications, 2021, p. 80–95.
[12] S. Natarajan and et al., Advanced AI Techniques for Cybersecurity in Modern Networked Systems, vol. 14(3), IEEE Systems Journal, 2020, pp. 4422-4432.
[13] B. Pavloski and et al., Cyber-Physical Security Agents for Smart Grid Network Protection, vol. 10(4), IEEE Transactions on Smart Grid, 2019, pp. 3438-3447.
[14] S. Amiri and A. Rahmani, Cybersecurity in Smart Grid Networks: Challenges and Solutions, Springer, 2019.
[15] M. H. Bhuyan, D. K. Bhattacharyya and J. Kalita, Network Anomaly Detection: Methods, Systems, and Tools, vol. 16(1), IEEE Communications Surveys & Tutorials, 2017, pp. 303-336.
[16] M. A. Javed and et al., Modeling Multi-Stage Network Attacks Using Intelligent Agents, vol. 74, Computers & Security, 2018, pp. 235-249.
[17] T. M. Mitchell, Machine Learning, vol. 2nd ed., McGraw-Hill Education, 2020.
[18] Y. LeCun, Y. Bengio and G. Hinton, Deep Learning, vol. 521(7553), Nature, 2015, pp. 436-444.
[19] K. P. Murphy, Machine Learning: A Probabilistic Perspective, MIT Press, 2012.
[20] D. Koller and N. Friedman, Probabilistic Graphical Models: Principles and Techniques, MIT Press, 2009.
[21] K. Lab., Spam Evolution in 2020 Report, 2021.
[22] J. Schmidhuber, Deep Learning in Neural Networks: An Overview, vol. 61, Neural Networks, 2015, pp. 85-117.
[23] C. C. Aggarwal, Neural Networks and Deep Learning: A Textbook, Springer, 2018.
[24] V. N. Vapnik, The Nature of Statistical Learning Theory, Springer-Verlag, 1995.
[25] P. Lashkov, C. Schäfer and I. Kotenko, Intrusion Detection in Unlabeled Data with Quarter-Sphere Support Vector Machine.
[26] J. Thorsten, Making Large-Scale SVM Learning Practical, Lehrstuhl VIII Dortmund: Artificial Intelligence, 1998.
[27] J. Kim, H. Lee and H. K. Kim, Deep Learning-Based Detection and Mitigation of Distributed Denial of Service Attacks in the IoT Environment, vol. 8, IEEE Access, 2020, pp. 207994-208003.
[28] C. Liu and et al., A Machine Learning-Based Framework for Assessing Security Vulnerabilities in Computer Networks, vol. 95, Journal of Network and Computer Applications, 2018, pp. 16-32.
[29] B. Labs, Data Analysis Technologies. [Online resource].
[30] T. MathWorks, MATLAB and Simulink for Technical Computing.