J Supercomput
https://doi.org/10.1007/s11227-018-2397-3

GARUDA: Gaussian dissimilarity measure for feature representation and anomaly detection in Internet of things

Shadi A. Aljawarneh[1] · **Radhakrishna Vangipuram**[2]

Abstract The objective of any anomaly detection system is to efficiently detect several types of malicious traffic patterns that cannot be detected by conventional firewall systems. Designing an efficient intrusion detection system has three primary challenges that include addressing high dimensionality problem, choice of learning algorithm, and distance or similarity measure used to find the similarity value between any two traffic patterns or input observations. Feature representation and dimensionality reduction have been studied and addressed widely in the literature and have also been applied for the design of intrusion detection systems (IDS). The choice of classifiers is also studied and applied widely in the design of IDS. However, at the heart of IDS lies the choice of distance measure that is required for an IDS to judge an incoming observation as normal or abnormal. This challenge has been understudied and relatively less addressed in the research literature both from academia and from industry. This research aims at introducing a novel distance measure that can be used to perform feature clustering and feature representation for efficient intrusion detection. Recent studies such as CANN proposed feature reduction techniques for improving detection and accuracy rates of IDS that used Euclidean distance. However, accuracies of attack classes such as U2R and R2L are not significantly promising. Our approach GARUDA is based on clustering feature patterns incrementally and then representing features in different transformation space through using a novel fuzzy Gaussian dissimilarity measure.

✉ Shadi A. Aljawarneh
saaljawarneh@just.edu.jo

Radhakrishna Vangipuram
radhakrishna_v@vnrvjiet.in

[1] Jordan University of Science and Technology, Irbid, Jordan

[2] Centre for Excellence in Networks and Security, Department of Information Technology, VNR Vignana Jyothi Institute of Engineering and Technology, Hyderabad, India

Published online: 07 May 2018

Experiments are conducted on both KDD and NSL-KDD datasets. The accuracy and detection rates of proposed approach are compared for classifiers such as kNN, J48, naïve Bayes, along with CANN and CLAPP approaches. Experiment results proved that proposed approach resulted in the improved accuracy and detection rates for U2R and R2L attack classes when compared to other approaches.

Keywords Anomaly detection · Feature representation · Intrusion · Dimensionality · Clustering · Distance measure

1 Introduction

Intrusion detection is a subproblem of anomaly detection which aims at determining whether the incoming traffic is normal or abnormal [1]. The inherent goal of any intrusion detection system is to automate the detection process which can efficiently detect anomalous traffic patterns as and when they take place in the network. To determine whether an incoming traffic is normal (safe) or anomalous (un-safe), most network intrusion detection systems apply data mining techniques such as classification (also called as the supervised learning technique in machine learning) and clustering (un-supervised learning). The fundamental task of any intrusion detection system is to compare the incoming traffic pattern to the existing traffic pattern and judge if the incoming traffic pattern is normal or abnormal. The accuracy or efficiency of this comparison implicitly depends on the underlying similarity measure (or distance measure) that is used to compare the similarity between traffic patterns. It is for this reason the accuracy and efficiency of network intrusion detection systems depend on the similarity or distance measure that is used to compare traffic patterns and how these similarity measures are used.

Most of the existing research contributions published in the literature do not explicitly specify what similarity measures have been used. This is also one of the fundamental drawbacks which make the task of comparing existing works difficult and inconsistent. This also limits the possibility of examining the scalability and trade-off of distance or similarity measure that is used. For example, the widely accepted and popularly known Euclidean distance measure (L_p-norm, $p = 2$) suffers from high dimensionality [2, 3] and sparse data. The second problem with Euclidean distance is that it has no finite upper distance bound [1, 4–9]. Similarly, when applying cosine measure on vectors expressed in terms of term frequency, the magnitude of vectors does not have any effect on similarity value [3]. Weller-Fahy et al. [1] present an exhaustive study on various similarity measures that have been used in studies within intrusion detection model and other applications not related to this area. An encyclopedia of distances [10] is a dictionary of distances published in 2006 which explores properties of distance functions and metrics followed by several types of distance functions related to natural sciences, computer-related distances, real-world distances, and applied and classical mathematics.

In general, a function is said to be a distance function (or distance measure) if it satisfies symmetry, non-negativity and reflexive properties [10]. Formally, let S be any

set. A function $\mathcal{F}_{\text{dist}}$: $S \times S \rightarrow R$ is called a distance function on S if and only if all three properties of the function $\mathcal{F}_{\text{dist}}$ hold good, i.e.,

1. $\mathcal{F}_{\text{dist}}\left(S_P, S_q\right) = \mathcal{F}_{\text{dist}}\left(S_q, S_P\right)$ (Symmetry property)
2. $\mathcal{F}_{\text{dist}}\left(S_P, S_q\right) \geq 0$ (Non-negative property)
3. $\mathcal{F}_{\text{dist}}\left(S_P, S_P\right) = 0$ (Reflexive property)

In addition to the above three properties, if a function $\mathcal{F}_{\text{dist}}$ satisfies another property called triangular inequality property [10], then it is termed as a distance metric.

This paper is outlined as follows: Sect. 2 reviews the related literature. In Sect. 3 the proposed distance metric is introduced, and Sect. 4 outlines algorithms for incremental clustering dimensionality reduction and intrusion detection using proposed distance metric. Section 5 outlines the experimental results and analysis of proposed intrusion detection algorithm approach. Section 6 concludes the research.

2 Related literature

With an increasing growth in the use of Internet and latest technological advancements, threats are also raising to an alarming level. Internet is serving as a bridge between business activities and end users. Business activities and information of various organizations are geographically widespread, and almost every business activity is Internet centric. The dependency of these activities over Internet is also continuously increasing. Internet of things (IoT) is new wave of connecting beyond mobiles, laptops, wearables, smart homes, cars, devices, sensors and healthcare sensors. As per the Gartener report by 2020 the total number of devices is estimated to cross 20.6 billion and by the year 2025 it is estimated to reach an unimaginable figure equal to one trillion [11].

IoT devices have a key role in bridging the gap between sensors and digital world, playing a deterministic role in the human life. The concept of full-length setup of smart cities, smart monitoring of health care, smart home, smart security, affluence of personal life is already on its way with accelerated development of IoT industry. With the rapid increase in the use of IoT, there are several direct applications which will have direct and indirect societal impacts. Few of them are listed in Fig. 1.

Internet of things (IoT) is a new paradigm [12] which addresses the integration of various physical devices or objects to Internet. In addition to bringing several benefits by deepening the use and availability of Internet-connected devices in our day-to-day activities, IoT throw several inherent security challenges that must be addressed. Although several approaches for intrusion detection models have been proposed using genetic algorithms, artificial neural networks, fuzzy techniques and data mining techniques, most of these techniques do not address a complete anomaly detection system that can improve accuracy, detection or false alarm rates. Recent attack Ransomware 'WannaCry' proved that there are still many vulnerabilities in the design of the intrusion detection systems which have influenced damage over 150 countries and recorded over 200,000 victims. Thus, there is an emerging need to design and come up with a robust anomaly detection system. This section reviews and compares different approaches and models of intrusion detection systems from various perspectives. These include comparison of experiment results achieved using different intrusion detection mod-

Fig. 1 IoT applications

els based on the choice of distance measure, feature selection, learning techniques (supervised learning or un-supervised learning) and hybrid techniques.

2.1 Distance measure

It is a fact that every intrusion or anomaly detection system implicitly requires using the distance measure in one stage or the other. In general, intrusion detection systems (IDS) require applying distance measure when performing (i) feature selection or (ii) supervised and un-supervised learning. However, most studies that address intrusion detection models and algorithms do not explicitly address the distance measure used to compare observations to judge if the observation is normal or an attack. This fact is coined in a detailed survey performed very recently in 2014 by Weller-Fahy et al. [1]. According to findings [1], measures for performing feature selection may be classified into (a) power distance and (b) distances based on distribution laws. The category of power distance includes either Euclidean or Mahalanobis [1]. Distance functions under second category include Kullback–Leibler distance (KLD) [13–15] and those based on entropy [16, 17]. Using KLD, the gain ratio of each feature is computed and feature selection is based on this gain value [13, 14]. Novel measures [18–21] for performing supervised classification are proposed to detect intrusions in networks. Gunupudi et al. [22, 23] proposed fuzzy membership functions for intrusion detection and dimensionality reduction that are based on clustering.

2.2 IDS with and without feature selection

Intrusion detection system (IDS) may be defined as the system used to supervise and recognize traffic data or behavior of the network user to identify and report intrusions.

Attacks can be classified into misuse or anomaly attacks. The ability of misuse-based IDS is limited to detecting only known intrusions. Anomaly-based IDS can detect new incoming attacks based on the normal traffic profile pattern. Most IDS can detect either of the two attacks. In general, an IDS should be able to successfully uncover both misuse and anomaly attacks. Zhang et al. [24] proposed two hierarchical intrusion detection frameworks called serial hierarchical IDS (SH-IDS) and parallel hierarchical IDS (PH-IDS) applying principles of artificial neural network and uses radial basis function (RBF). The limitation of SH-IDS is that it can recognize misuse attack with high accuracy, whereas anomaly attacks follow adaptive learning. Both these frameworks do not consider performing feature selection. To improve the performance of SH-IDS, Zhang et al. proposed PH-IDS. The evaluation methods considered are detection rate (DR) and false positive (FP). Experiments are conducted on KDD-99 dataset with 10,000 (normal) and 20,000 (attack) samples in training and testing datasets. Since the IDS is based on ANN, some neural network algorithms require normalizing the data before applying underlying algorithm. Backpropagation learning (BPL) is performed on un-normalized data, whereas RBF is applied on normalized data. The overall detection rate of misuse detection using BPL is 99.2 and 98% with RBF. For misuse detection, three attack classes (Probe, R2L and DoS) and one normal class is considered. For anomaly detection, the overall detection rate obtained is 93.7 and 99.2% using BPL and RBF, respectively.

Peddabachigari et al. [25] propose hybrid intelligent IDS model to detect misuse and anomaly attacks. The baseline techniques are support vector machine (SVM) and decision tree (DT). Hybrid model is obtained by combining SVM and DT. Experiments are conducted on dataset consisting of 11,982 (training—5092; testing—6890) records that are randomly chosen from KDD-Cup 99 dataset. Each record has 41 features, and no feature selection is applied. Samples are chosen proportional to the number of records in each class except for smallest class. The evaluation is based on accuracy parameter. The DT classifier percentage accuracies are 99.64, 99.86, 96.83, 68, 84.19 for Normal, Probe, DoS, U2R, R2L classes, respectively. Accuracies of SVM classifier are as follows: normal (99.64), Probe (98.57), DoS (99.92), U2R (40) and R2L (33.92). Hybrid IDS (SVM + DT) recorded accuracies are as follows: Normal (99.7), Probe (98.57), DoS (99.92), U2R (48) and R2L (37.8). The disadvantage is that both U2R and R2L attack accuracies are not appreciable.

Özyer et al. [26] apply classification and association rule mining techniques of data mining to propose intelligent IDS for anomaly and misuse detection. IDS [26] is based on iterative rule learning and uses genetic classifier that uses generated fuzzy rules. The evaluation method chosen is the detection rate of classifier and considers all 41 features and does not perform any feature elimination. The percentage detection rates obtained are 95.8, 54.1, 97.4, 10.9 and 6.9 for Normal, Probe, DoS, U2R and R2L classes, respectively, over a training dataset consisting of 494,014 samples and 311,029 samples in testing set.

A supervised NIDS for anomaly detection is proposed by Li et al. [27] based on TCM-KNN. The performance of IDS is evaluated considering TP and FP rates. Experiments are conducted on sample records extracted from KDD dataset, and the recorded TP rate is 99.7 without feature selection. The numbers of training and testing

samples considered are 49,402 and 12,350. The distance measure chosen is Euclidean distance for kNN classifier.

A genetic algorithm-based IDS model for anomaly detection is proposed in the work of Hansen et al. [28]. The IDS is evaluated considering detection rates, and even this approach does not address any feature selection technique. Giacinto et al. [29] and Hu et al. [30] are anomaly detection-based IDS that uses kNN, SVM ensemble and AdaBoost DT technique, respectively. The evaluation is carried out by considering classifier detection rates and applies no feature selection technique. Tajbakhsh et al. [31] propose an anomaly detection model that performs association-based classification by generating fuzzy association rules without applying feature selection. The baseline classifiers considered are SVM and *k*NN. For misuse-based detection, the detection rate and false-positive rate achieved are 91 and 3.34%, while for anomaly detection these rates are 80.6 and 2.95%, respectively. Specifically, for each attack class, these values are as follows: DoS (78.9%), Probe (88.5%), U2R (68.6%) and R2L (6.2%).

An artificial neural network (ANN)-based anomaly detection approach using fuzzy clustering called FC-ANN is proposed by Wang et al. [32] for improving detection precision rates of low-frequency attacks, namely R2L and U2R. FC-ANN anomaly detection model [32] applies fuzzy clustering on training set and generates subsets. These training subsets are used to train different ANNs, and a meta-learner is built using these ANNs to eliminate errors that may have been introduced by different ANNs. FC-ANN does not apply feature selection technique and considers precision, recall and F-measure for evaluating classifier performance. Lin et al. [33] propose a feature selection algorithm that applies simulated annealing (SA), support vector machine (SVM) and decision tree concept for building an anomaly detection model. Feature selection [33] is performed using SA and SVM techniques to generate features that can best improve the accuracy of decision tree classifier. The number of feature generated is 23 out of a total of 41 features in KDD dataset.

Baig et al. [34] model for anomaly detection uses naïve Bayes and ANN baseline classifiers and performs feature selection using Information Gain, Gain ratio and GMDH (iterated polynomial regression) methods. A feature representation approach called CANN is proposed by Lin et al. [35] for anomaly detection. CANN considers transforming each input observation of dataset to a single distance value by computing two distance values. CANN applies clustering on training and testing sets based on the number of decision classes in the dataset by applying *k*-means clustering technique. The performance of CANN is compared to SVM and *k*NN approaches on KDD dataset with 6 and 19 attributes. The distance measure used for anomaly detection using CANN approach is the Euclidean distance.

Gunupudi et al. [22, 36] propose an approach for intrusion detection which uses fuzzy membership function for feature clustering and representation. Recent surveys on application of text mining techniques for intrusion detection are discussed [37, 38]. An ensemble of SVM and ant colony optimization-based model for intrusion detection called combining support vectors with ant colony (CSVAC) is proposed in [39]. Aljawarneh et al. [40] propose a hybrid intrusion detection model, and experiments are conducted on NSL-KDD dataset.

A novel algorithm named optimum allocation-based least square support vector machine (OA-LSSVM) is proposed by Kabir et al. [41]. The model is suitable for

both static and incremental datasets [41] and suggests an approach for determining the sample size of the dataset using z-value also called standard normal variate by considering KDD dataset. Wang [42] approach for intrusion detection is based on the principle of feature augmentation and considers the NSL-KDD dataset for experiments. Feature transformation is achieved using logarithm marginal density ratio, and the resulting features are input for SVM classifier. The limitation of the approach [42] is its suitability and applicability only to the binary case of intrusion detection problems.

A recursive feature addition and bi-gram technique for feature reduction is proposed in [43]. The IDS built is tested on the ISCX2012 dataset for studying the capability of detecting intrusions [43]. Hamed [43] proposed a new evaluation metric by combining the metrics accuracy, detection rate and false alarm rates which shall be helpful to study and evaluate different systems. Intrusion detection system proposed by Akashdeep [44] is based on the feature reduction achieved using Information Gain and correlation, and the classification is achieved using ANN-based classifier. However, the manual preprocessing carried out is the major disadvantage of this approach followed by disadvantages due to Information Gain- and correlation-based approaches. The approach proposed by Sumaiya [45] fuses the Chi-square-based feature selection and multi-class support vector machine. The drawback is that the number of observations considered for experimentation is not optimal (33,300 records) and hence cannot be considered as a baseline for comparison. The choice of Chi-square feature selection is based on research study by Yang and Pedersen [46]. Further, CANN is performed on KDD dataset with 6, 19 and all the 41 features, whereas these features are not considered for experiments carried out in [45]. Instead, experiments [45] are conducted on 31 features. The next section introduces the proposed distance measure which is motivated from the research by Lin, Jiang and Lee [2, 3].

3 Proposed distance metric

In this section, we introduce the proposed distance measure to perform incremental feature clustering. The basic idea is to use these clusters generated from incremental feature clustering to achieve dimensionality reduction (or input transformation) by appropriately choosing the threshold value. The total number of clusters generated depends on the allowable dissimilarity limit (also called user threshold). The threshold and number of clusters have an inverse relation, i.e., a low threshold value chosen outputs more number of clusters, and a higher threshold value generates comparatively fewer clusters.

For designing the proposed distance measure, we choose the simple Gaussian function and use this function to design the proposed distance measure. Equation (1) denotes the simple Gaussian function $g(x, \mu)$, where 'k' defines the highest peak of the Gaussian function curve, 'μ' denotes the location of the center of this peak, and 'σ' is deviation which is used to control the width of the Gaussian curve, also called Gaussian RMS or deviation.

$$g\,(x, \mu) = k * \mathrm{e}^{-\left(\frac{x-\mu}{\sigma}\right)^2} \tag{1}$$

3.1 Dissimilarity measure for evolutionary clustering

Problem definition Given two vectors $\overrightarrow{X_P}, \overrightarrow{X_q}$ and threshold value represented by δ^U. Two vectors $\overrightarrow{X_P} = (X_{P_1}, X_{P_2}, X_{P_3}, \ldots, X_{P_m})$ and $\overrightarrow{X_q} = (X_{q_1}, X_{q_2}, X_{q_3}, \ldots, X_{q_m})$ are similar iff $\mathcal{F}_{\text{dist}}(\overrightarrow{X_P}, \overrightarrow{X_q}) \leq \delta^U$ where $\mathcal{F}_{\text{dist}}$ is any chosen distance function.

Let $\overrightarrow{X_P}$ and $\overrightarrow{X_q}$ denote any two vectors represented formally as $\overrightarrow{X_P} = (X_{P_1}, X_{P_2}, X_{P_3}, \ldots, X_{P_m})$ and $\overrightarrow{X_q} = (X_{q_1}, X_{q_2}, X_{q_3}, \ldots, X_{q_m})$ such that X_{P_i} is the posterior probability value and $X_{P_i} \in \{0, 1\}$. The dimensionality of both vectors $\overrightarrow{X_P}$ and $\overrightarrow{X_q}$ is equal to 'm'. Further, let δ^U be the user-defined allowable distance. The dissimilarity value between any two vectors $\overrightarrow{X_P}$ and $\overrightarrow{X_q}$ can be computed by applying the dissimilarity function represented using Eq. (2), with $\alpha = 0.3679$.

$$\mathcal{F}_{\text{dist}}\left(\overrightarrow{X_P}, \overrightarrow{X_q}\right) = \frac{1 - \mu\left(\overrightarrow{X_P}, \overrightarrow{X_q}\right)}{1 + \alpha} = \frac{1 - \mu\left(\overrightarrow{X_P}, \overrightarrow{X_q}\right)}{1.3679} \tag{2}$$

where

$$\mu\left(\overrightarrow{X_P}, \overrightarrow{X_q}\right) = \prod_{i=1}^{i=m} \exp^{-\left(\frac{X_{p_i} - X_{q_i}}{\sigma^f}\right)^2} \tag{3}$$

Equation (3) represents the fuzzy Gaussian membership function $\prod_{i=1}^{i=m} \exp^{-(\frac{X_{p_i} - X_{q_i}}{\sigma^f})^2}$ which is equal to the product of all exponential function values that are generated by considering different values for 'i' such that $1 \leq i \leq m$. The variable 'σ^f' used in Eq. (3) is the standard deviation value given by Eq. (4).

The expression for computing deviation is given by Eq. (4),

$$\sigma^f = \frac{\delta^U}{\sqrt{\ln_e\left(\frac{1}{\text{abs}(1-(1+\alpha)*\delta^U)}\right)}} \tag{4}$$

where δ^U is the allowable dissimilarity chosen between 0 and 1 and $\alpha = 0.3679$.

3.2 Derivation

We derive the expression for computing similarity between $\overrightarrow{X_P}$ and $\overrightarrow{X_q}$ by considering both $\overrightarrow{X_P}$ and $\overrightarrow{X_q}$ as single dimension vectors. The similarity between theses vectors is given by Eq. (5).

$$\text{Sim}\left(\overrightarrow{X_P}, \overrightarrow{X_q}\right) = \frac{\mu\left(X_{p_1}, X_{q_1}\right) + \alpha}{1 + \alpha} = \frac{\text{e}^{-\left(\frac{X_{p_1} - X_{q_1}}{\sigma^f}\right)^2} + \alpha}{1 + \alpha} \tag{5}$$

3.2.1 Case 1: Best case

In the best case, the distance between patterns is 0 (or similarity is 1), i.e., $|X_{p_1} - X_{q_1}| = 0$. This reduces to Eq. (6)

$$\text{Sim}\left(\overrightarrow{X_P}, \overrightarrow{X_q}\right) = \frac{1+\alpha}{1+\alpha} = 1 \tag{6}$$

3.2.2 Case 2: Worst case

In the worst case, $|X_{p_1} - X_{q_1}| = 1$. We have two possible analytical outcomes for this situation.

(i) $\sigma^f = 0$, then $\mu\left(\overrightarrow{X_P}, \overrightarrow{X_q}\right) = 0$.

The similarity value in this case is given by Eq. (7)

$$\text{Sim}\left(\overrightarrow{X_P}, \overrightarrow{X_q}\right) = \frac{0+\alpha}{1+\alpha} = \frac{\alpha}{1+\alpha} \tag{7}$$

We have for the worst case, $\text{Sim}\left(\overrightarrow{X_P}, \overrightarrow{X_q}\right) = 0$.

$$\text{i.e.} \quad \frac{\alpha}{1+\alpha} = 0 \xrightarrow{\text{yields}} \alpha = 0 \tag{8}$$

From Eq. (8), if $\alpha = 0$, then $\text{Sim}(\overrightarrow{X_P}, \overrightarrow{X_q}) = \mu(\overrightarrow{X_P}, \overrightarrow{X_q})$. This means that $\mu(\overrightarrow{X_P}, \overrightarrow{X_q})$ itself is considered as the similarity function which is not true and hence is ruled out.

(ii) $\sigma^f = 1$, then $\mu\left(\overrightarrow{X_P}, \overrightarrow{X_q}\right) = 0.3679$.

This gives Eq. (9)

$$\text{Sim}\left(\overrightarrow{X_P}, \overrightarrow{X_q}\right) = \frac{0.3679+\alpha}{1+\alpha} \tag{9}$$

We have for the worst case, $\text{Sim}\left(\overrightarrow{X_P}, \overrightarrow{X_q}\right) = 0$.

$$\text{i.e.,} \quad \frac{0.3679+\alpha}{1+\alpha} = 0 \xrightarrow{\text{yields}} \alpha = -0.3679 \tag{10}$$

By substituting the value for $\alpha = -0.3679$ in Eq. (5) and rationalizing both numerator and denominator, the expression for similarity is given by Eq. (11),

$$\text{Sim}\left(\overrightarrow{X_P}, \overrightarrow{X_q}\right) = \frac{\mu\left(\overrightarrow{X_P}, \overrightarrow{X_q}\right) + 0.3679}{1.3679} \tag{11}$$

The dissimilarity between $\overrightarrow{X_P}$ and $\overrightarrow{X_q}$ may hence be obtained using Eq. (12).

$$\mathcal{F}_{\text{dist}}\left(\overrightarrow{X_P}, \overrightarrow{X_q}\right) = 1 - \text{Sim}\left(\overrightarrow{X_P}, \overrightarrow{X_q}\right) \tag{12}$$

$$= \frac{1 - \mu\left(\overrightarrow{X_P}, \overrightarrow{X_q}\right)}{1.3679} \tag{13}$$

In general, Eq. (14) defines the generalized expression for finding the distance between two vectors, $\overrightarrow{X_P}$ and $\overrightarrow{X_q}$

$$\mathcal{F}_{\text{dist}}\left(\overrightarrow{X_P}, \overrightarrow{X_q}\right) = \frac{1 - \prod_{i=1}^{i=c} \exp^{-\left(\frac{X_{p_i} - X_{q_i}}{\sigma^f}\right)^2}}{1.3679} \tag{14}$$

3.3 Standard deviation

Let δ^U denote the allowable dissimilarity value between any two vectors $\overrightarrow{X_P}$ and $\overrightarrow{X_q}$. The expression for deviation is derived by considering single dimension vectors. Then, for any given dimension (say, ith dimension), we have the distance between X_{p_i} and X_{q_i} given by Eq. (15)

$$\delta^U = \sqrt{\left(X_{p_i} - X_{q_i}\right)^2} \tag{15}$$

Using the proposed distance function, the fuzzy distance between $\overrightarrow{X_P}$ and $\overrightarrow{X_q}$ is given by Eq. (16)

$$\mathcal{F}_{\text{dist}}\left(\overrightarrow{X_P}, \overrightarrow{X_q}\right) = \frac{1 - \text{e}^{-\left(\frac{|X_{p_i} - X_{q_i}|}{\sigma^f}\right)^2}}{1.3679} \tag{16}$$

From Eqs. (15) and (16), we have Eq. (17)

$$\frac{1 - \text{e}^{-\left(\frac{\delta^U}{\sigma^f}\right)^2}}{1.3679} = \delta^U \tag{17}$$

Solving Eq. (17), deviation is given by Eq. (18),

$$\sigma^f = \frac{\delta^U}{\sqrt{\ln_e\left(\frac{1}{\text{abs}(1 - 1.3679 * \delta^U)}\right)}} \tag{18}$$

3.4 Transformed dissimilarity (or threshold value)

Equation (15) gives the distance between two single dimension vectors, $\overrightarrow{X_P}$ and $\overrightarrow{X_q}$. So, we have Eq. (19) from Eq. (15)

$$\delta^U = |X_{p_i} - X_{q_i}| \tag{19}$$

The expression for transformed threshold value is represented using Eq. (20) and can be obtained using Eqs. (16) and (19),

$$\delta^f = \frac{1 - \mathrm{e}^{-\left(\frac{\delta^U}{\sigma^f}\right)^2}}{1.3679} \tag{20}$$

3.5 An example

To understand the computation of dissimilarity using the proposed measure, the working of the proposed distance measure is explained below with three examples. The distance computed using the proposed distance measure and the Euclidean distance measure is considered for explanation and comparison.

Example 1 Consider two feature patterns, $\overrightarrow{X_P} = \langle 1.0, 0.6, 0.4\rangle$ and $\overrightarrow{X_q} = \langle 0.0, 0.6, 0.4\rangle$, and an allowable dissimilarity $\delta^U = 014$.

3.5.1 Case 1: Euclidean distance measure

In this case, Euclidean distance is equal to 1.0049 and exceeds 1. Euclidean distance has lower limit (0), but has no upper limit (analytically, ∞). If we can restrict the upper bound to 1, then we can define the similarity between two feature patterns within finite bounds. Example 2 below uses the normalized Euclidean distance to determine the similarity between patterns and then compares to the proposed measure.

3.5.2 Case-2: Proposed distance measure

Using the proposed measure, we have $\mathcal{F}_{\text{dist}}(\overrightarrow{X_P}, \overrightarrow{X_q}) = 0.71$ and $\mathcal{F}_{\text{dist}}(\overrightarrow{X_P}, \overrightarrow{X_q}) > \delta^f = 0.14$. Hence, the two patterns are dissimilar. The upper bound distance obtained using the proposed measure always holds between 0 and 1. It requires no normalization.

Example 2 Consider two feature patterns $\overrightarrow{X_P} = \langle 0.5, 0.7, 0.5\rangle$, $\overrightarrow{X_q} = \langle 0.4, 0.6, 0.4\rangle$, and $\delta^U = 0.14$.

3.5.3 Case 1: Euclidean distance measure

Using Euclidean distance measure without normalization, the distance between $\overrightarrow{X_P}$ and $\overrightarrow{X_q}$ is equal to 0.1732. This shows that given patterns are dissimilar. However, these two patterns are actually similar w.r.t their behavior trends as shown in Fig. 2. This behavior trend similarity cannot be determined by Euclidean distance even for low dimensionality, if it is not normalized.

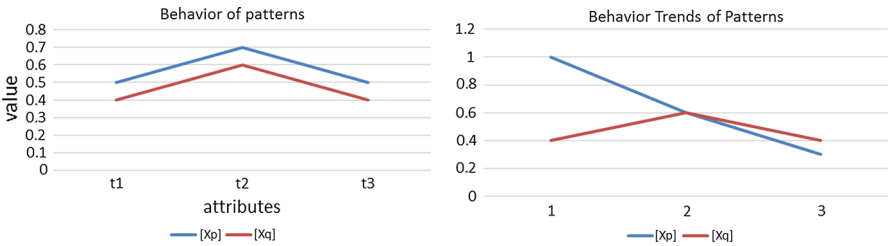


Fig. 2 Behavior trends of two feature patterns

Now, consider the normalized Euclidean distance measure. This is equal to 0.1 (i.e., $0.1732/\sqrt{3}$). This proves two patterns are similar. It is to be noted that the un-normalized Euclidean distance fails in the previous case for the reason that it has no upper bound. However, in the latter case, the distance is normalized and hence patterns are similar. This is one of the problems which are overcome using the proposed measure.

3.5.4 Case 2: Proposed distance measure

Given $\delta^U = 0.14$. Applying Eqs. (18) and (20) we get $\sigma^f = 0.3036$ and $\delta^f = 0.14$. In this example, $\delta^f = \delta^U = 0.14$. The distance between $\overrightarrow{X_P}$ and $\overrightarrow{X_q}$ in the transformed space is 0.075141, i.e., $\mathcal{F}_{\text{dist}}(\overrightarrow{X_P}, \overrightarrow{X_q}) = 0.075 < \delta^f = 0.14$. Patterns $\overrightarrow{X_P}$ and $\overrightarrow{X_q}$ are similar as the dissimilarity condition $\mathcal{F}_{\text{dist}}(\overrightarrow{X_P}, \overrightarrow{X_q}) \leq \delta^f$ holds good.

Example 3 Let $\overrightarrow{X_P}$ and $\overrightarrow{X_q}$ be two patterns given by $\overrightarrow{X_P} = \langle 0.8, 0.6, 0.4 \rangle$ and $\overrightarrow{X_q} = \langle 0.4, 0.6, 0.4 \rangle$, and an allowable dissimilarity, $\delta^U = 0.14$.

3.5.5 Case 1: Euclidean distance measure

Using normalized Euclidean distance, the distance between two patterns is equal to 0.23. The patterns $\overrightarrow{X_P}$ and $\overrightarrow{X_q}$ are considered dissimilar.

3.5.6 Case 2: Proposed distance measure

Applying Eqs. (18) and (20) we have $\sigma^f = 0.3036$ and $\delta^f = 0.14$. The distance between $\overrightarrow{X_P}$ and $\overrightarrow{X_q}$ in transformed space is 0.32, i.e., $\mathcal{F}_{\text{dist}}(\overrightarrow{X_P}, \overrightarrow{X_q}) = 0.32 > \delta^f$. $\overrightarrow{X_P}$ and $\overrightarrow{X_q}$ are hence dissimilar.

4 Algorithms

Some of the terms and definitions used in network intrusion and anomaly detection are described below.

1. Dataset

A dataset is defined as a collection of all observations. All observations of dataset are usually represented as a matrix called observation matrix such that rows represent observations and columns represent features.

2. Observations

An observation is usually a data entity which can be a network packet, traffic pattern or network process.

3. Feature

The attributes that best describe an observation are called the feature. Each observation of a dataset is defined over a set of attributes called features. For example, in KDD dataset a feature can be a time stamp, source or destination IP address and length of a network packet.

4. Preprocessing

Preprocessing is usually performed to transform the existing feature value to a different value to eliminate the bias or inconsistency due to one or more feature. For example, some ANN algorithms require input data to be normalized before applying actual algorithm. Similarly, some algorithms might require data to be transformed to a different projection space before applying.

5. Feature selection

Feature selection is the process of selecting subset feature set from the original set of features. For example, NSL-KDD dataset is obtained by considering important 19 features out of the total 41 features [35].

6. Feature extraction

Feature extraction is a dimensionality reduction technique that essentially aims at finding optimal transformation matrix for a given set of observations.

4.1 Incremental feature clustering

Algorithm 1: Incremental feature clustering

Input : User threshold, Observation matrix with m-dimensions
Output : Soft Clusters

δ^U : user threshold
δ^f : transformed threshold
σ^f : standard deviation in transformed space
m : dimensionality
f_i : i[th] feature
f_{i_c} : Probability of i[th] feature w.r.t c[th] decision label
$\overrightarrow{f_t}$: m-dimension feature pattern, $\overrightarrow{f_t}=<f_{i_1}, f_{i_2}, \ldots\ldots f_{i_m}>$
$\mathcal{F}_{dist}$: distance function between two feature patterns $\overrightarrow{f_t}$ and $\overrightarrow{f_J}$
σ^0 : Initial standard deviation, m-dimensional vector
$\sigma^{(g)}$: Deviation, $<\sigma^{g1}, \sigma^{g2}, \sigma^{g3}, \ldots\ldots\ldots, \sigma^{gm}>$
$m^{(g)}$: mean of g[th] cluster, $<m^{g1}, m^{g2}, m^{g3}, \ldots\ldots\ldots, m^{gm}>$
i : iterative index variable, varies from 1 to m
g : number of clusters, initially g = 0
C_g : g[th] cluster

Begin

1. Read the allowable dissimilarity value, δ^U
2. Determine the initial deviation value, σ^f and transformed threshold value, δ^f using Eq. (18) and Eq. (20) respectively.
3. Choose the first feature pattern (say $\overrightarrow{f_1}$). Initially, g = 0. Generate the first cluster by placing the first feature pattern, say, $\overrightarrow{f_1}$ in this cluster. Set g = g+1. Call it C_g. Now, C_g contains only $\overrightarrow{f_1}$.
4. Initialize mean and deviation of generated cluster (first cluster).
   4.1 Mean of the first cluster is an m-dimensional vector and is same as the first feature pattern. i.e. $\overrightarrow{m^{(g)}}=\overrightarrow{f_1}$
   4.2 Initial standard deviation of the new generated cluster, $\overrightarrow{\sigma^{(g)}}=<\sigma^{(f)}, \sigma^{(f)}, \sigma^{(f)}, \ldots\ldots\ldots m\ times>$
5. If no other feature pattern exists then go to step-10 else go to step-6.
6. Choose the feature pattern that is not yet clustered, say $\overrightarrow{f_p}$. Determine the distance between this feature pattern, $\overrightarrow{f_p}$ and mean of each of the existing clusters denoted using, $\overrightarrow{m^{(g)}}$ with the proposed distance measure. i.e compute $\mathcal{F}_{dist}(\overrightarrow{f_p}, \overrightarrow{m^{(g)}})$.
7. If ($\mathcal{F}_{dist}(\overrightarrow{f_p}, \overrightarrow{m^{(g)}}) \leq \delta^f$)
   Add $\overrightarrow{f_p}$ to existing cluster and go to step-8
   else
   Set g= g+1. Create a new cluster. Call it C_g and repeat the process in step-4.
8. Update the mean of the cluster after adding the feature pattern to the cluster. The new mean shall be the average of the existing feature pattern.
9. Go to step-5.
10. At the end of incremental clustering, we finally have 'g' clusters generated and their respective mean vectors.
11. Compute the respective standard deviation vector for each of these generated clusters by considering only those feature patterns that exist in respective clusters.
12. Update the deviation of final clusters. The final deviation of each generated cluster shall be the sum of the initial chosen deviation and the respective deviation obtained in step-11.

End

Our approach for feature clustering is motivated from the contribution of [40], and we use the same approach but with the proposed distance measure. In [40], the clustering process is based on the similarity value computed using the basic Gaussian fuzzy membership function. The limitation is that the deviation must be specified by the user. For different values of deviation, we get separate set of clusters. Our feature clustering approach clusters all feature patterns into subset clusters and is based on the proposed distance measure that requires specification of threshold. For feature

clustering, it is just sufficient to specify the allowable dissimilarity constraint and the required deviation value can be obtained using Eq. (18). Feature clustering is performed in an incremental manner. The dimensionality of feature pattern is equal to the number of decision class labels. Initially, the clustering process starts by considering the first feature pattern and then incrementally clusters each feature pattern using the proposed distance measure and considering deviation value that is computed based on the allowable dissimilarity limit. At the end of clustering process, we have a set of clusters with each cluster consisting subset of these feature patterns.

Incremental feature pattern clustering algorithm outlined (Algorithm 1) uses feature patterns as input and outputs feature pattern clusters. Information from these generated clusters is used to obtain the transformed observation matrix. The process of obtaining feature pattern corresponding to a given feature is explained in Sect. 4.4. Algorithm 2 discussed below uses Algorithm 1.

4.2 Algorithm for dimensionality reduction and feature representation

This subsection outlines algorithm for projecting input observation (or observation matrix) on to a different dimension space. A high value of dissimilarity threshold results in less number of clusters, and a low value of dissimilarity threshold generates comparatively more number of clusters. The idea behind the proposed dimensionality reduction method is to express each observation in terms of clusters. Naturally, the generated number of clusters is less than the actual number of features.

Algorithm 2: Dimensionality reduction/ Feature transformation

Input : Observation matrix with decision class label
Output : Observation matrix with reduced dimensionality

Begin

1. Read the input observation matrix with 'o' instances, 'f' features and 'd' class labels. Represent the above information in the form of a matrix. Call the matrix as instance-feature matrix (also called as observation -feature matrix) denoted by $[\text{O-F}]_{|o|x|f|}$.
2. Obtain feature pattern vector for every feature using procedure outlined in sub-section D.
   2.1 Given a class label, say 'd', the conditional probability that feature, 'f' belongs to the decision class, 'd' must be computed for each decision class. (computed using equation 25)
   2.2 The set of all values computed in step 2.1 for a given feature w.r.t each class label is represented as a vector called as feature pattern (or feature pattern vector)
3. Apply incremental clustering (algorithm-1) with feature patterns as input and generate feature clusters (say 'g' clusters). The mean and deviation of individual clusters are the corresponding cluster representatives.
4. Compute the distance, i.e $\mathcal{F}_{dist}$ between each feature pattern to every generated cluster. Represent these feature-cluster dissimilarities in the form of a matrix called the soft feature-cluster dissimilarity matrix, $[\text{F-C}]_{fxg}$. Alternately, soft matrix can represent similarity values.
5. Transform the original observation-feature matrix by multiplying matrices $[\text{O-F}]_{oxf}$ and $[\text{F-C}]_{fxg}$. The result is a soft matrix (or soft observation-cluster matrix) denoted by $[\text{O-C}]_{oxg}$.
6. Matrix $[\text{O-C}]_{oxg}$ obtained in step-5 is transformed representation of input observation matrix. A high value of allowable dissimilarity yields fewer number of clusters while a low dissimilarity threshold results large number of clusters.
7. Output the reduced dimensionality matrix, $[\text{O-C}]_{oxg}$.

End

4.3 Algorithm for anomaly detection

Algorithm 3: *Anomaly and Intrusion Detection based on evolutionary clustering and proposed feature transformation*

Input : Proposed Classifier, features, observations
Output : Classification label

1. Read the input dataset.
2. Read the allowable threshold.
3. Divide the dataset to training and testing groups.
4. Pre-process training and testing groups. Pre-processing the training group involves following tasks
   4.1 Generate feature pattern vectors for each class label.
   i.e Given a class, then the probability that the specific feature belongs to the given class must be computed.
   4.2 Apply evolutionary clustering (Algorithm-1) for a chosen dissimilarity constraint.
   4.3 Apply dimensionality reduction algorithm.
   4.4 Generate soft (or hard) feature-cluster matrix.
   4.5 Obtain the transformed observation matrix from the soft or hard matrix generated in 4.4.
   4.6 Store the resultant observation matrix.
5. Repeat steps 4.5 to 4.6 for testing set.
6. Apply classification algorithms, e.g: kNN, SVM, C4.5 etc considering the transformed observation matrix along with class label.
7. Evaluate classifier performance considering parameters such as classification accuracy, detection rates or false rates

4.4 Feature pattern computation from feature value

Let $[O\text{–}F]_{|o|\times|f|}$ be the observation matrix where columns represent features and rows represent observations. Suppose that $|o|$ is the number of observations, $|f|$ be the number of features, and $|d|$ be the number of decision class labels in the dataset. Notations O, F and D denote observation, feature and decision label sets with cardinalities $|o|$, $|f|$ and $|d|$, respectively. We use Eqs. (21)–(23) to represent observation, feature and decision class sets, respectively,

$$O = \{O_1, O_2, O_3, \ldots, O_o\} \tag{21}$$

$$F = \{F_1, F_2, F_3, \ldots, F_f\} \tag{22}$$

$$D = \{D_1, D_2, D_3, \ldots, D_d\} \tag{23}$$

Let F_i denote ith feature from the feature set F and f_{io} represent the value of feature f_i in Oth observation. The notation $\overrightarrow{X_i}$ represents the feature pattern vector corresponding to feature F_i. Feature pattern vector is computed for each feature F_i in the feature set F, and the dimensionality of feature pattern vector is equal to $|d|$. The feature pattern vector $\overrightarrow{X_i}$ is represented using Eq. (24) where each X_{id} is the probability that the feature F_i belongs to decision class label D_d.

$$\overrightarrow{X_i} = (X_{i1}, X_{i2}, X_{i3}, \ldots, X_{id}) \tag{24}$$

The value for X_{id} can be obtained using Eq. (25)

$$X_{id} = \frac{\sum_{i=1}^{i=|o|} f_{ji} X \mathcal{M}_d^j}{\sum_{i=1}^{i=|o|} f_{ji}} \tag{25}$$

where f_{ji} indicates jth feature value in the ith observation from the observation matrix and $\mathcal{M}_d^j$ is equal to 1, if the jth feature denoted by F_j belongs to the decision class D_d and is equal to 0 and if F_j does not belong to decision class D_d.

5 Experimental results and analysis

5.1 Description of benchmark datasets

Experiments are performed using two widely accepted benchmark datasets: (i) KDD-99 and (ii) NSL-KDD. One reason for choosing KDD and NSL-KDD datasets is the un-availability of other freely or publicly available datasets for evaluating the performance of network intrusion detection systems. Another reason for choosing these datasets is that the performance results achieved by various intrusion detection systems or models over these two datasets can be better evaluated and compared with existing contributions. This is also because most of the significant research contributions in the literature use these two benchmark datasets for their experimental studies. One more advantage of performing experimentation using these two datasets is that it helps researchers to perform analysis of different intrusion detection models and eventually come out with innovative approaches and techniques that can address improved computational efficiency and detection accuracies when compared to the existing network intrusion detection models.

5.1.1 KDD-Cup 99 dataset

KDD-Cup 99 dataset is the most widely used benchmark dataset for evaluating various intrusion and anomaly detection models. KDD-Cup 99 dataset consists of 41 features. This dataset is collected considering protocols such as TCP, UDP and ICMP. Attacks associated with each protocol are listed in Table 1. KDD-Cup 99 dataset has 23 attack names, and each attack is associated with one of the four attack classes named as denial-of-service (DoS), Remote2Local (R2L), User2Root (U2R) and Probing attacks. Table 2 presents the number of attack instances in the KDD-Cup 99 dataset w.r.t each attack type. For example, in attack class name, say 'Neptune,' there are 1, 01,201 attack instances in the training dataset and 2, 80,790 attack instances w.r.t smurf. Table 3 presents attack classes (DoS, U2R, R2L, Probe) and their respective attacks in KDD-Cup 99 dataset. Table 4 depicts total records in whole KDD dataset, corrected KDD dataset and 10% KDD dataset, respectively. The number of observations in original KDD is 4,898,430. Out of these 9, 72,780 are normal instances, and the remaining 3,925,650 are attack instances. In the corrected KDD dataset 60,593 are normal traffic patterns out of the total 311,029 observations.

Table 1 Attacks associated with each protocol

Protocol name	Names of the attacks under each protocol
UDP	Normal, teardrop, satan, nmap, rootkit
TCP	Normal, neptune, guess_passwd, land, portsweep, buffer_overflow, phf, warezmaster, ipsweep, multihop, warezclient, perl, back, ftp_write, loadmodule, satan, spy, imap, rootkit
ICMP	Normal, portsweep, ipsweep, smurf, satan, pod, nmap

Table 2 Attack-wise number of instances in KDD-Cup 99

Attack name	Attacks in training dataset	Attack	Attacks in training dataset
neptune	101,201	ftp_write	8
portsweep	1040	imap	12
satan	1589	guess_passwd	53
warezclient	1020	nmap	231
back	2203	smurf	280,790
ipsweep	1247	warezmaster	20
buffer_overflow	30	perl	3
multihop	7	spy	2
loadmodule	9	pod	264
teardrop	979	land	21
rootkit	10	phf	4

Table 3 Attack classes of KDD-Cup 99 dataset

DoS (6)	U2R (4)	R2L (8)		Probe (4)
back	buffer overflow	ftp_write	multihop	portsweep
land	rootkit	guess_passwd	phf	satan
neptune	perl	imap		ipsweep
smurf	load module	spy		namp
teardrop		warezclient		
pod dos		warezmaster		

Table 4 Attack-wise number of samples in KDD-Cup 99

KDD dataset	Total	DoS	Probe	R2L	U2R	Normal
Whole KDD	4,898,430	3,883,370	41,102	1126	52	972,780
Corrected KDD	311,029	229,853	4166	16,347	70	60,593
10% KDD	494,020	391,458	4107	1126	52	97,277

Table 5 Number of instances of each attack class in training and testing sets of KDD-Cup 99

Dataset label	DoS (%)	Probe (%)	U2R (%)	R2L (%)	Total attack (%)	Total normal (%)
Training data	79.24	0.83	0.01	0.23	80.31	19.69
Testing data	73.90	1.34	0.07	5.20	81.51	19.49

Table 6 Details of records in 10% KDD training set

	Original records	Distinct records	Reduction rate (%)
Attack	3,925,650	262,178	93.32
Normal	972,781	812,814	16.44
Total	4,898,431	1,074,992	78.05

Table 7 Details of records in 10% KDD test dataset

	Original records	Distinct records	Reduction rate (%)
Attack	250,436	29,378	88.26
Normal	60,591	47,911	20.92
Total	311,027	77,289	75.15

The total number of observations in 10% KDD dataset [47, 48] proposed by Portnoy et al. in the year 2001 is 4, 94,020, and out of these 97,277 are normal with the rest of observations being attack instances.

It can be seen from Table 4 that total instances of DoS, Probe and Normal are 99.76% of whole dataset, while U2R attacks are only 0.0011% of the whole KDD dataset. The sample of U2R attacks is very less and hence many anomaly detection algorithms fail to detect U2R attacks successfully.

Table 5 depicts percentage of observations with respect to training and testing sets. The training data contains 80.31% attacks and 19.69% normal input observation samples. Similarly, testing data contains 81.51% (attack) and 19.49% (normal) samples.

Tables 6 and 7 show details of records in 10% KDD train and test sets, respectively. In the training set, there are 3,925,650 attacks (original records) out of which only 2, 62,178 are unique and 8, 12,814 records are normal out of 972,781 records. Similarly, out of the total 3, 11,027 records in test set, only 77,289 are unique and the remaining is redundant.

Even though the KDD data are widely used test bed for the design of the anomaly detection system, it also suffers from few issues as follows:

(a) The synthesized data in the KDD-Cup 99 do not seem to be correlated with the real-time data.

(b) The tool used for the traffic collection TCPDump suffers from overloading, and majority of data components have chances of being dropped. No mention of dropped data is discussed in the dataset literature.
(c) The attacks in the dataset are not discussed in detail. For example, Probing may not necessarily be an attack. It can be considered as an attack only when a specific threshold is overcome.
(d) Tables 6 and 7 give details of the percentage of duplicate records in KDD-Cup 99 train and test datasets. It proves that KDD suffers from redundant data records which substantially influences the detection accuracy [20].
(e) Tavallaee et al. [49] perform statistical analysis of KDD dataset and point out drawbacks of KDD dataset.

5.1.2 NSL-KDD dataset

Experiments are also performed using NSL-KDD dataset which is considered as another benchmark dataset and has been suggested to overcome problems pointed by McHugh [50]. NSL-KDD dataset has following advantages over KDD-99 dataset.

(a) One problem with KDD dataset is the presence of redundant records in the training set. There is a scope for the classifier to have a biasing behavior toward more frequent records present in the training set. This problem is overcome in NSL-KDD dataset.
(b) The second problem with KDD dataset is from redundant records present in testing set of KDD-Cup 99 dataset. The presence of these redundant records in the testing set can make classifiers decision biased toward more frequent records. NSL-KDD dataset has no redundancy with respect to its testing set.
(c) The number of records in the training set and testing set are proportionately reasonable.
(d) Records are of different difficulty level.

The main reason for selecting NSL-KDD dataset is that it overcomes most drawbacks of the original KDD dataset; however, it still suffers from problems pointed by McHugh [50]. It also helps researchers to evaluate the performance of different anomalous detection models. Also, the relative number of records available with NSL-KDD dataset is also reasonable to conduct experiments. Eventually, experiments conducted and evaluated shall also be consistent with other intrusion detection models.

5.2 Performance evaluation

For evaluating the performance of the proposed approach for intrusion detection, i.e., GARUDA, we consider accuracy rate, detection rate and false alarm rate of classifiers. Consider the confusion matrix given in Table 8.

From this, we can obtain accuracy, detection and false alarm rates of classifier as given by Eqs. (26)–(28).

Table 8 Confusion matrix

	Predicted class label	
	Positive	Negative
Actual class label		
Positive	True positive	False negative
Negative	False positive	True negative

5.2.1 Accuracy

Given parameters true positive (TP), true negative (TN), false negative (FN) and false positive (FP), the accuracy of a classifier can be obtained using Eq. (26)

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{26}$$

5.2.2 Detection rate

Given true positive (TP) and false negative (FN), the detection rate is computed using Eq. (27)

$$\text{Detection rate} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{27}$$

5.2.3 False alarm rate

Given false positive (FP) and true negative (TN), fall out also called false alarm rate can be computed using Eq. (28)

$$\text{Fall positive rate} = \frac{\text{FP}}{\text{FP} + \text{TN}} \tag{28}$$

All experiment results discussed in this research are performed on an Intel core-i5 3470, 3.24 GHz CPU with 4 GB of memory running on windows operating system using the IDS model implemented in Java and using Weka [51, 52] for classifiers Bayes net, naïve Bayes, J48 and kNN.

5.3 Study I: KDD-Cup 99 with 19 features

The first experiment is based on KDD-Cup 99 dataset by considering nineteen (19) features out of the total 41 features. These 41 features are selected based on the work of Xue-qin [53]. Recent work by Lin et al. [35] proposed a novel approach for feature representation, CANN (combining cluster centers and nearest neighbors), for intrusion detection and used KDD-Cup 99 dataset with 494,020 samples by considering only nineteen features out of 41 based on the work of Xue-qin [53]. Experiments are conducted by considering classifiers *k*NN, SVM, J48 and CANN to evaluate the

Table 9 Confusion matrix of SVM

	Normal	U2R	DoS	R2L	Probe	Accuracies
Normal	1068	0	96,210	0	0	0.981
U2R	1	0	51	0	0	0
DoS	0	0	391,458	0	0	0.794
R2L	20	0	1106	0	0	0
Probe	0	0	4107	0	0	0
Overall accuracy						0.7945

Table 10 Confusion matrix of kNN ($k = 1$) classifier

	Normal	U2R	DoS	R2L	Probe	Accuracies
Normal	96,749	19	355	127	28	0.997
U2R	24	16	1	11	0	0.333
DoS	194	5	391,251	3	5	0.999
R2L	62	8	6	1048	2	0.881
Probe	55	0	7	0	4045	0.991
Overall accuracy						0.99815

Table 11 Confusion matrix of J48 classifier

	Normal	U2R	DoS	R2L	Probe	Accuracies
Normal	97,913	12	31	20	22	0.998
U2R	33	16	1	2	0	0.571
DoS	38	0	391,416	1	3	1.00
R2L	69	0	0	1056	1	0.979
Probe	44	0	16	0	4047	0.994
Overall accuracy						0.99941

respective classifier performance on this dataset. The idea behind CANN approach for feature representation is to transform all observations in the dataset to a single numeric value. Experiments using CANN approach for intrusion detection are performed on KDD-Cup 99 dataset with 19 features by transforming all samples to a single dimension value achieving dimensionality reduction. Percentage accuracy achieved using CANN ($k = 1$) is 99.46, while for SVM it is 95.37 over KDD-Cup 99 dataset with 19 features. Tables 9, 10 and 11 give confusion matrix for classifiers kNN, J48 and SVM classifiers using the proposed feature representation approach for a user-defined allowable dissimilarity limit equal to 0.0005. Using the proposed feature representation approach and similarity function, the accuracy achieved with kNN ($k = 1$) is 99.81% which is 0.35% higher than CANN. For J48 classifier, accuracy obtained is equal to 99.94% which is higher than CANN accuracy.

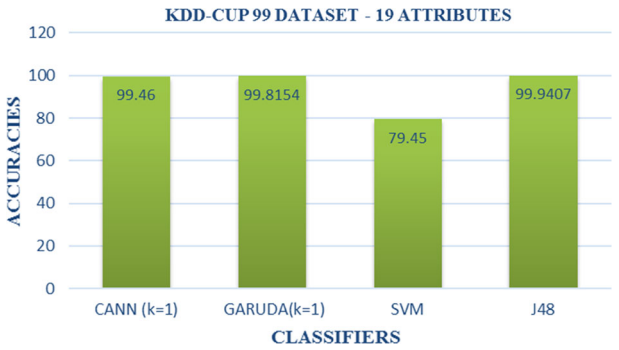


Fig. 3 Comparison of classifier accuracies over KDD-99 with 19 features

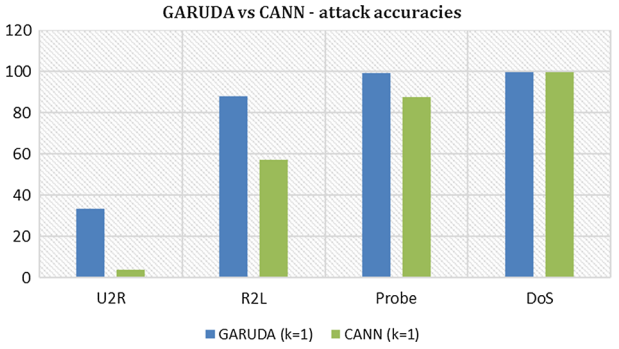


Fig. 4 Attack accuracies for GARUDA and CANN

The advantage of our approach is proved with the ability of the classifier to detect U2R and R2L attacks which are very much better compared to CANN, i.e., U2R and R2L attack accuracies using CANN ($k = 1$) are equal to 3.85 and 57.02%, respectively. Using the proposed feature representation approach with GARUDA measure, attack accuracies for U2R and R2L classes are equal to 33.33 and 88.1%, respectively. Using J48 classifier, U2R and R2L attack accuracies are equal to 57.1 and 97.9%, respectively.

The graph in Fig. 3 shows the comparison of accuracies of CANN ($k = 1$), GARUDA ($k = 1$), SVM and J48 classifier.

Figure 4 shows the graph plotted by considering individual attack class accuracies w.r.t KDD dataset. The recorded attack class accuracy values using GARUDA are comparatively better to CANN.

The graph plotted in Fig. 5 shows the comparison of attack class accuracies of GARUDA to kNN and CANN. It is proved that the accuracy using GARUDA is comparatively much better and promising than those achieved using kNN and CANN approaches.

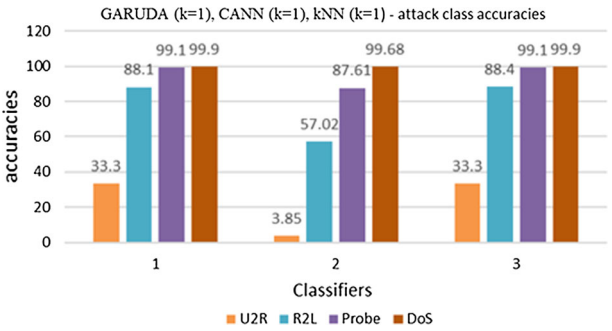


Fig. 5 GARUDA versus *k*NN and CANN

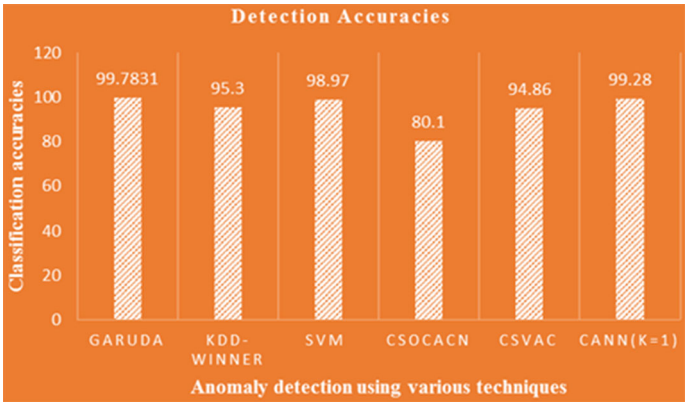


Fig. 6 Comparison of detection accuracies of anomaly detection techniques with the proposed approach

5.4 Study II: KDD-Cup 99 with 41 features

In the second experiment, we considered KDD-Cup 99 dataset with 41 features. The dataset has a total number of 494,021 samples. Experiments are conducted considering classifiers such as *k*NN, SVM and J48, and accuracies are recorded for this dataset. Using the proposed approach with GARUDA ($k = 1$) the accuracy achieved for U2R and R2L accuracies are 65 and 96.1%, while accuracies with *k*NN ($k = 1$) classifier for U2R and R2L are 68.1 and 95.3%.

ROC values for R2L and U2R attack classes using the proposed feature representation with GARUDA and *k*NN ($k = 5$) classifier are 0.9967 and 0.9519, while these values are 0.9967 and 0.9339 for kNN classifier with all 41 features. The accuracy improvement is seen in the U2R attack class. The ROC values for U2R attack class using J48 classifier with and without feature representation approach are 89.6 and 89.26%, respectively. Detection accuracy of the proposed approach is also compared to accuracies obtained using SVM, CSOCACN, CSVAC, CANN ($k = 1$) and plotted as a graph shown in Fig. 6. The percentage detection accuracy of GARUDA ($k = 1$) is 99.78, while recent approaches such as CSVAC [39] and CANN ($k = 1$) [35] have detection accuracies equal to 94.86 and 99.28, respectively.

The bar graph shown in Fig. 7a shows the comparison of kNN classifier accuracies with feature representation using GARUDA using KDD dataset with 41 attributes. The values of k are set to 1, 3 and 5. It is noticed that the accuracy of U2R attack class is highest for $k = 5$ and is equal to 83.33%. For R2L attack class, highest accuracy is noticed for $k = 2$ (96.1%) and is almost the same when $k = 1$ (96.1%). On average, the accuracies recorded using proposed feature representation approach have been better when compared to other approaches.

Figure 7b shows the comparison of accuracies of classifiers J48, Bayes net, naive Bayes and SVM for Normal, U2R, DoS, R2L and Probe classes. The dataset considered is KDD dataset with 41 attributes. Feature reduction is performed using the proposed approach with GARUDA, and then these classifiers are applied for this resulting dataset. It can be seen that the J48 classifier performed better to other classifiers.

Figure 7c shows the comparison of classifier accuracies of J48 and kNN using the proposed feature reduction approach. The similarity threshold is set to 0.9995, and clusters are generated. These clusters are used to transform the input dataset. Classifiers J48 and kNN are then applied over this transformed dataset, and accuracies are recorded. It can be seen that J48 and kNN classifier performance is the same for Normal and DoS attack class. For U2R attack class, J48 achieved 78.9% accuracy, whereas accuracies achieved for kNN classifier are equal to 65, 77.3 and 83.3% for values of k equal to 1, 3 and 5. Highest accuracy for kNN classifier is seen for $k = 5$. In similar lines, for R2L attack class, J48 achieved 96.4% accuracy, whereas accuracies achieved for kNN classifier are equal to 96.1, 96.5 and 95% for values of k equal to 1, 3 and 5. Highest accuracy for kNN classifier is seen for $k = 3$.

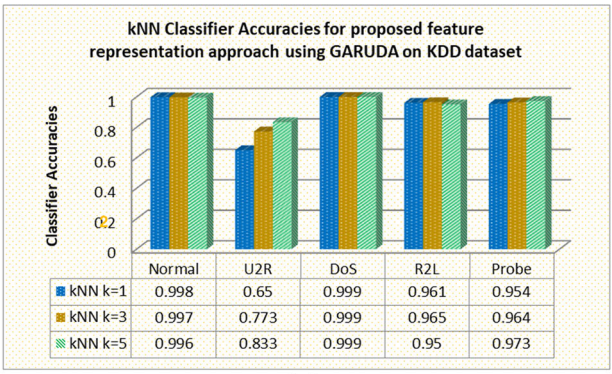
It may hence be concluded that for detecting R2L attack class by applying kNN with the proposed feature representation using GARUDA, the value of k may be fixed to 3 and for U2R attack class detection the value of k may be set to 5. In general, the performance of kNN for $k = 3$ is the same as J48 which is visible from Fig. 7c.

5.5 Study III: NSL-KDD with 41 features

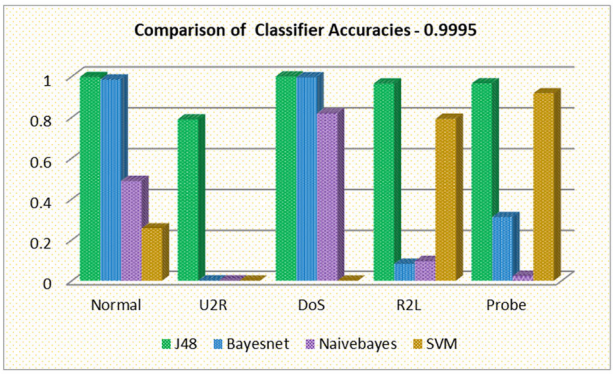
The third experiment is conducted on NSL-KDD dataset with 41 features. Experiments are conducted using various classifiers, and the results are evaluated to analyze the attack accuracies of various classifiers. The total number of records in NSL-KDD dataset is 125,973 samples. NSL-KDD dataset consisting of 125,973 samples over 41 features is considered for experimental evaluation. The percentage classification accuracies achieved for naïve Bayes, Bayesian, SVM and kNN are 83.89, 95.68, 96.92 and 99.68, respectively. Using the proposed feature representation approach, GARUDA ($k = 1$), the percentage accuracy obtained is 99.24 which is higher than naïve Bayes, Bayesian and SVM, and nearer to kNN ($k = 1$).

The bar graph in Fig. 8 shows the overall classification accuracies of various classifiers on the NSL-KDD dataset with 41 features.

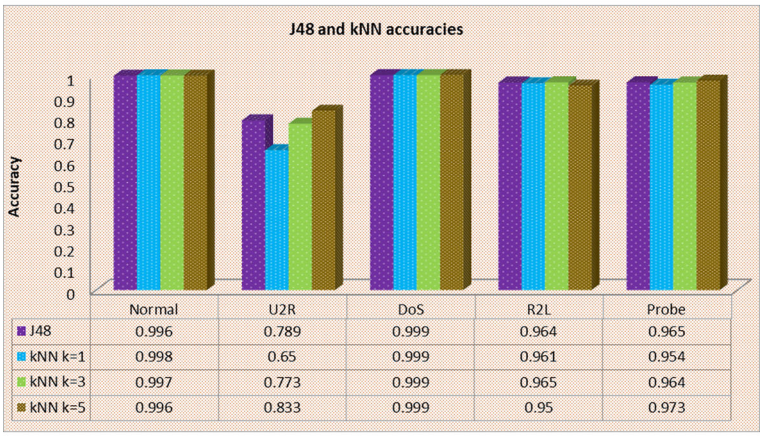
Tables 12 and 13 give the confusion matrix of the proposed approach with GARUDA ($k = 1$) and SVM classifier for NSL-KDD dataset. The fourth experiment is conducted by considering the NSL-KDD dataset with 19 features suggested in [35, 53]. Experiment results are discussed in the next subsection by considering various classifiers.



(a)



(b)



(c)

Fig. 7 **a** kNN accuracies with feature representation using GARUDA. **b** Comparison of various classifier accuracies. **c** Comparison of various classifier accuracies

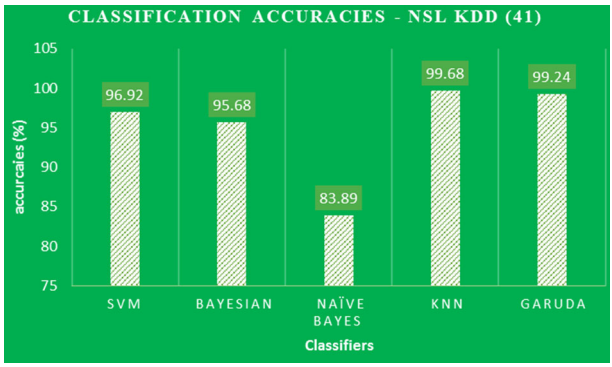


Fig. 8 Classification accuracies—NSL-KDD (41) dataset

Table 12 Confusion matrix for GARUDA ($k = 1$)

	Normal	U2R	DoS	R2L	Probe	Accuracies
Normal	67,022	100	56	146	19	99.7
U2R	54	45,690	0	183	0	54.5
DoS	49	0	945	1	0	99.3
R2L	103	217	0	11,335	1	94.4
Probe	28	0	0	0	24	97.2
Overall accuracy						99.24

Table 13 Confusion matrix for SVM—41 features

	Normal	U2R	DoS	R2L	Probe	Accuracies
Normal	65,988	510	289	546	10	0.964
U2R	1127	44,792	4	4	0	0.699
DoS	289	2	702	0	2	0.951
R2L	1062	4	1	10,589	0	0.676
Probe	18	1	8	0	25	0.989
Overall accuracy						96.92

5.6 Study IV: NSL-KDD with 19 features

In this experiment, we consider the NSL-KDD dataset with 19 features [53]. Experiments are conducted using classifiers such as kNN and J48 using the proposed approach and CANN with Gaussian measure [23]. Table 14 gives the confusion matrix for CANN using Gaussian measure [23].

The percentage accuracies on NSL-KDD dataset with 19 features for classes Normal, U2R, DoS, R2L and Probe are 89.74, 8, 89.07, 33.59 and 68.24. The detection accuracies are 88.46, 3.84, 34.77, 69.73 and 90.31 for Normal, U2R, R2L, Probe and DoS classes, respectively. The overall percentage accuracy is 86.98.

Table 14 Confusion matrix for CANN ($k = 1$)

	Normal	U2R	DoS	R2L	Probe
Normal	59,628	19	4217	612	2867
U2R	43	2	6	1	0
DoS	3547	4	41,479	25	872
R2L	576	0	30	346	43
Probe	2648	0	834	46	8128

Table 15 Confusion matrix for GARUDA ($k = 1$)

	Normal	U2R	DoS	R2L	Probe
Normal	66,256	35	407	168	477
U2R	41	10	0	0	1
DoS	239	0	45,502	1	185
R2L	129	0	0	854	12
Probe	344	1	246	11	11,054

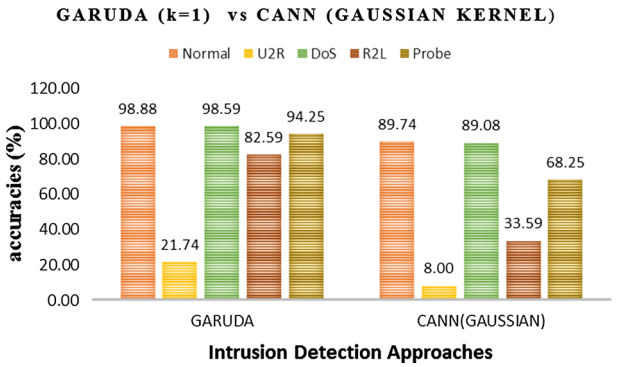


Fig. 9 GARUDA versus CANN (Gaussian kernel)

The confusion matrix for GARUDA is shown in Table 15. Percentage classification accuracies obtained for classes Normal, DoS, R2L, Probe and U2R are 98.88, 98.59, 82.59, 94.25 and 21.74, respectively. The overall percentage classification accuracy using GARUDA is 98.17. The percentage detection accuracies are equal to 98.38, 99.07, 94.83, 19.23 and 85.82 for Normal, U2R, R2L, Probe and DoS classes, respectively. The graph shown in Fig. 9 depicts the comparison of class-wise classification accuracies of CANN with Gaussian [23] and the proposed approach.

Figure 10 depicts the comparison of U2R and R2L attack class accuracies of CANN with Gaussian [23] and the proposed approach.

The graph shown in Fig. 11 depicts the comparison of class-wise detection accuracies of CANN with Gaussian [23] and the proposed approach.

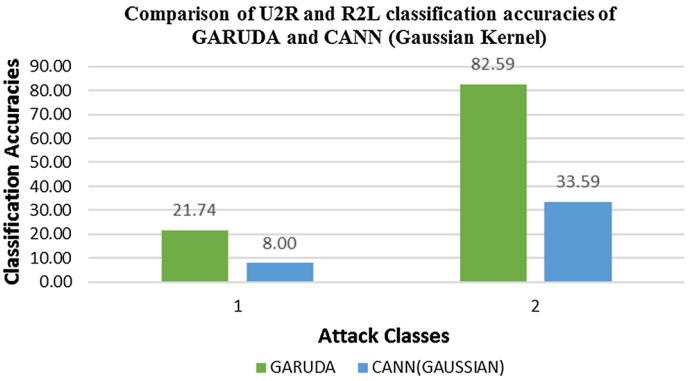


Fig. 10 GARUDA versus CANN (Gaussian kernel)

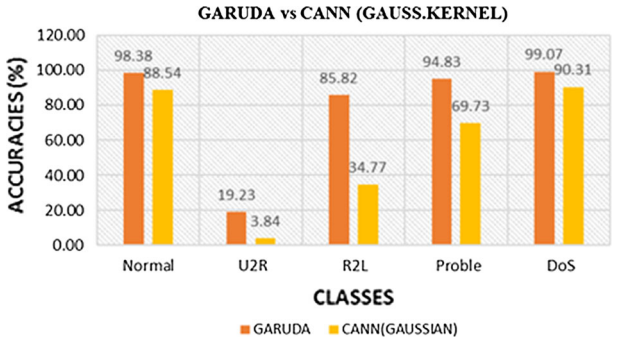


Fig. 11 Detection accuracies of all five classes

5.7 ROC values for NSL-KDD dataset (41 attributes and 494,021 samples)

Experiments are performed by considering NSL-KDD dataset, and ROC values for normal and attack classes are recorded applying feature representation with GARUDA. The graph shown in Fig. 12 shows the comparison of ROC values of classifiers J48, *k*NN, Bayes net, naive Bayes, SVM classifiers w.r.t U2R and R2L attack classes and feature representation using GARUDA for chosen similarity threshold equal to 0.999999. ROC values for all five classes of NSL-KDD dataset for chosen similarity threshold equal to 0.999999 are recorded and plotted as a bar chart in Fig. 13. Highest values of ROC for attack classes R2L (0.9966) and U2R (0.9518) classes are obtained for $k = 5$. For $k = 1$, the corresponding values are 0.9947 and 0.9366.

ROC values (for similarity threshold equal to 0.9995) obtained using feature representation with GARUDA w.r.t classifiers J48, kNN and naïve Bayes for attack classes R2L and U2R over NSL-KDD dataset are shown as a bar graph in Fig. 14.

ROC values achieved for classifiers J48, kNN ($k = 1, 3, 5$), Bayes net, naïve Bayes and SVM using the proposed approach of feature representation with GARUDA for similarity threshold equal to 0.9995 are plotted as a bar chart in Fig. 15. It can be seen that SVM has low ROC values for R2L and U2R attack classes when compared to

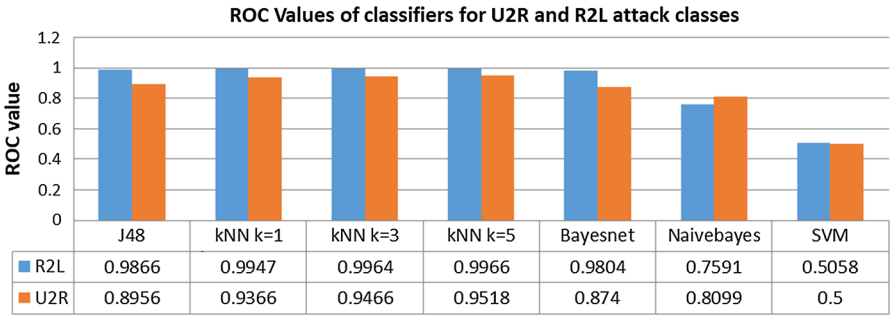


Fig. 12 ROC values for U2R and R2L attack classes of NSL-KDD dataset (41) with feature representation using GARUDA

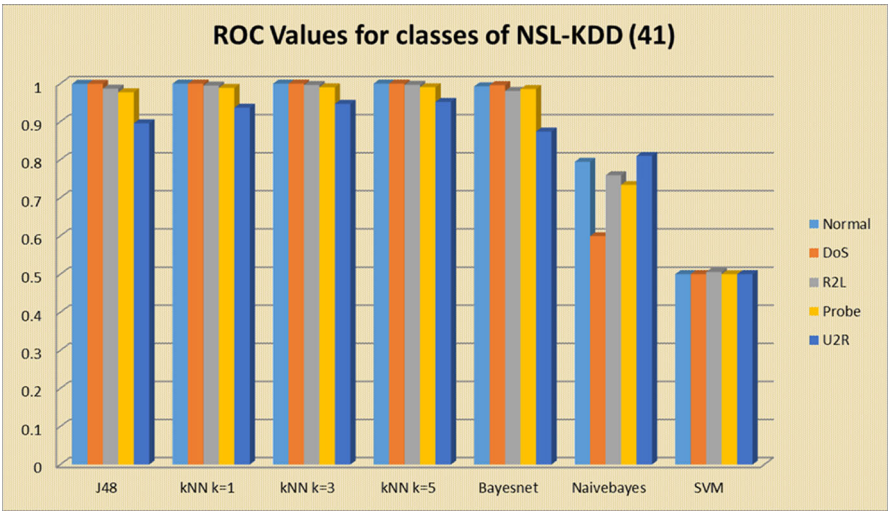


Fig. 13 ROC values for NSL-KDD dataset using feature representation with GARUDA

all other classifiers. Highest ROC value for R2L and U2R attack classes is recorded for kNN ($k = 5$) classifier, which is equal to 0.9966 and 0.9518, respectively. The performance of J48 classifier is also better when compared to Bayes net, naïve Bayes and SVM classifiers.

5.8 ROC values for KDD dataset (41 attributes and 494,021 samples)

Experiments are performed by considering KDD dataset, and ROC values for normal and attack classes are recorded applying feature representation with GARUDA. The bar graph shown in Fig. 16 plots ROC values of classifiers J48, kNN ($k = 1, 3, 5$), Bayes net, naïve Bayes and SVM for normal class. Feature representation is achieved using GARUDA measure for chosen similarity threshold equal to 0.9995, and the number of clusters formed is equal to 38. From Fig. 16, it is visible that the highest

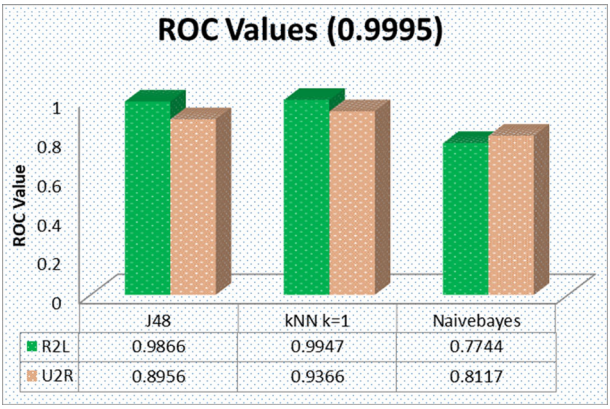


Fig. 14 ROC values for similarity threshold—0.9995 with feature representation using GARUDA

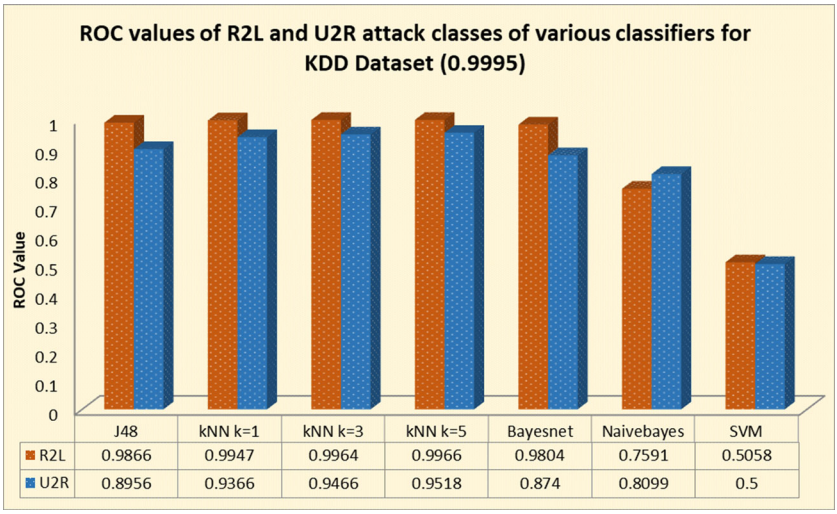


Fig. 15 ROC values for various classifiers with feature representation using GARUDA

ROC value of 0.9998 is achieved for kNN ($k = 5$) and the lowest value of 0.5002 is achieved for SVM classifier. For J48, ROC value is equal to 0.9991 which is almost the same as achieved for kNN. The uncovered ROC is equal to 0.0002 and 0.0009, respectively, for kNN and J48 classifiers.

ROC values obtained for Normal, DoS, R2L, Probe and U2R classes using J48 and kNN ($k = 5$) classifiers applying the proposed feature representation approach for similarity threshold equal to 0.9995 are plotted as a bar chart in Fig. 17. Using the proposed approach, ROC value for R2L attack class is 0.9966 for kNN ($k = 5$) and 0.9866 for J48 classifier. For U2R attack class, ROC value using kNN and J48 classifiers is 0.9518 and 0.8956, respectively. It is evident that ROC values for kNN classifier are better than for J48 classifier for all the five classes of KDD dataset.

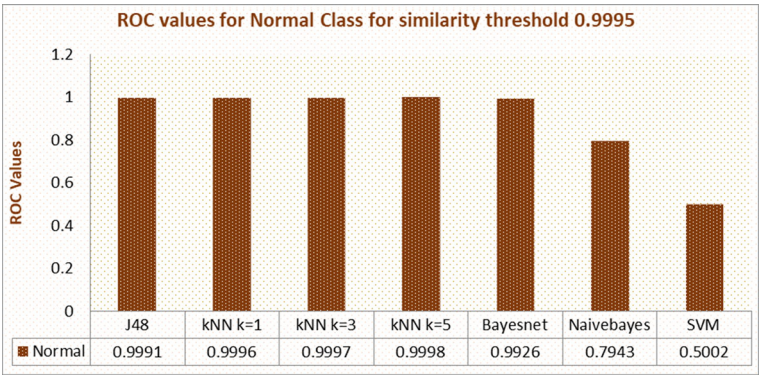
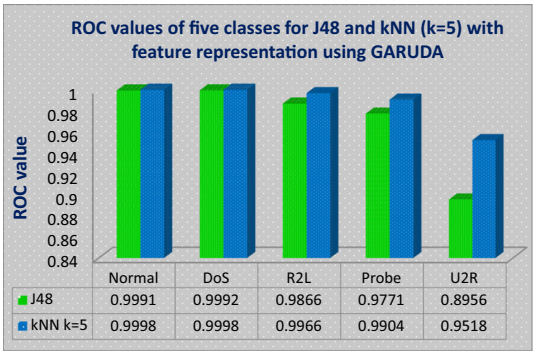


Fig. 16 ROC values for normal class with feature representation using GARUDA

Fig. 17 ROC values for five classes of KDD-41 for J48 and kNN classifiers using GARUDA



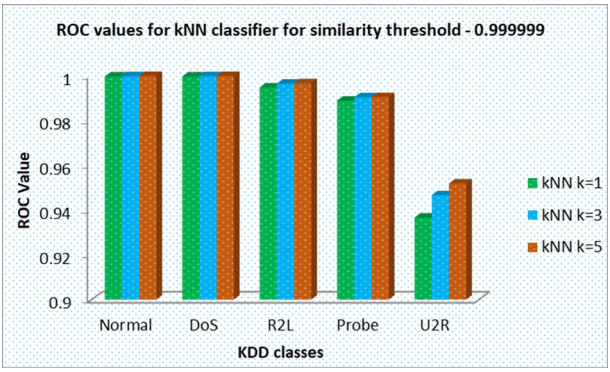


Fig. 18 ROC values of five classes for similarity threshold equal to 0.999999

Figure 18 depicts ROC values of Normal, DoS, R2L, Probe, U2R classes plotted as a bar chart for kNN classifier ($k = 1, 3, 5$). The chosen similarity threshold is equal to 0.999999. For Normal class and attack class (DoS) the ROC values are almost the same for $k = 1, 3, 5$. Slight variation is recorded and is visible when $k = 1, 3, 5$ for

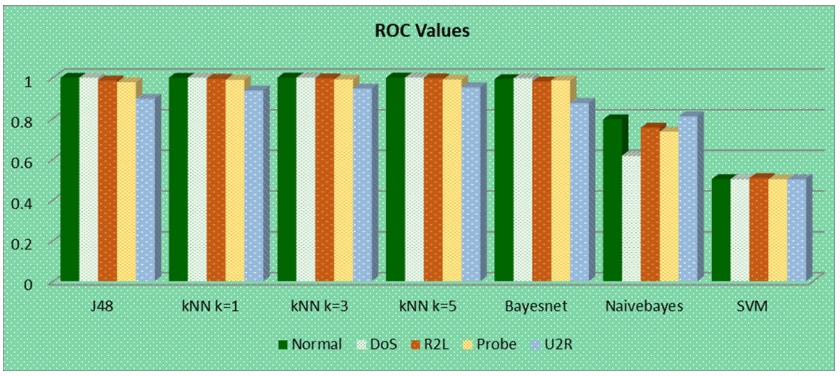


Fig. 19 ROC values of five classes for similarity threshold equal to 0.9999

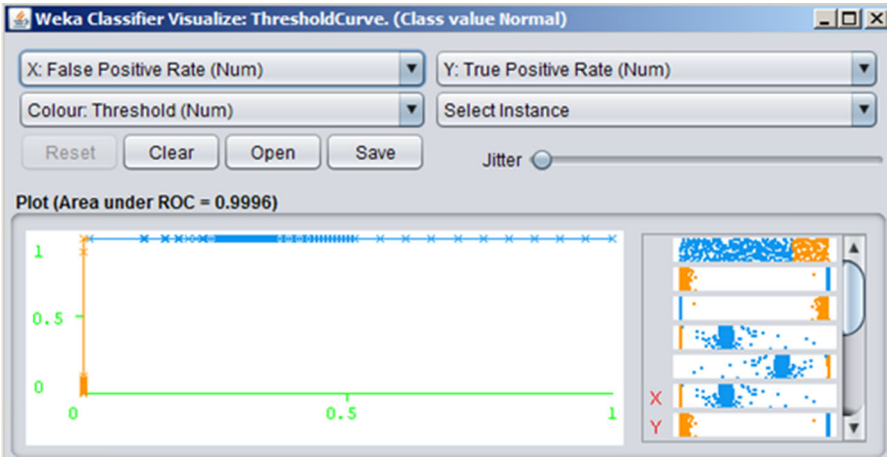


Fig. 20 ROC curve for Normal class

attack class U2R. Recorded ROC values of all classes for kNN are seen to be maximum for value of $k = 5$.

Figure 19 presents ROC values of various classes obtained using classifiers J48, kNN, Bayes net, naïve Bayes and SVM for similarity threshold equal to 0.9995. It is visible that SVM recorded lowest ROC values for all five classes (Normal, DoS, R2L, Probe and U2R) when compared to other classifiers. A relatively similar performance is achieved for J48 and kNN classifiers. In general, ROC values for U2R and R2L attack classes recorded for 0.9995, 0.9999 and 0.999999 are better using the proposed feature representation approach for kNN and J48 classifiers. The lowest performance is recorded for SVM w.r.t U2R and R2L attack classes.

Figure 20 shows visualization of ROC curve for Normal class using kNN ($k = 1$) classifier. The horizontal and vertical axis represents true-positive rate (TPR) and false-positive rate (FPR). The similarity threshold for feature clustering is set to 0.9995. For this threshold, clusters are generated and kNN classifier is applied for k equal to 1.

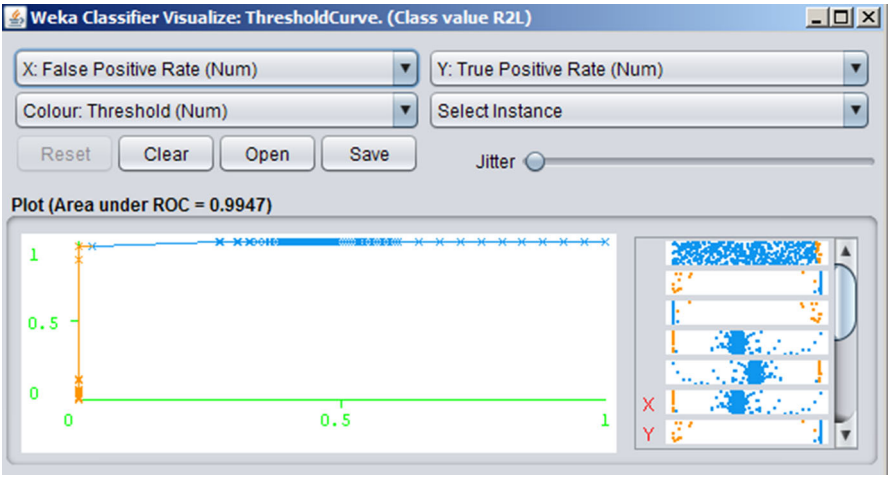


Fig. 21 ROC curve for R2L attack class

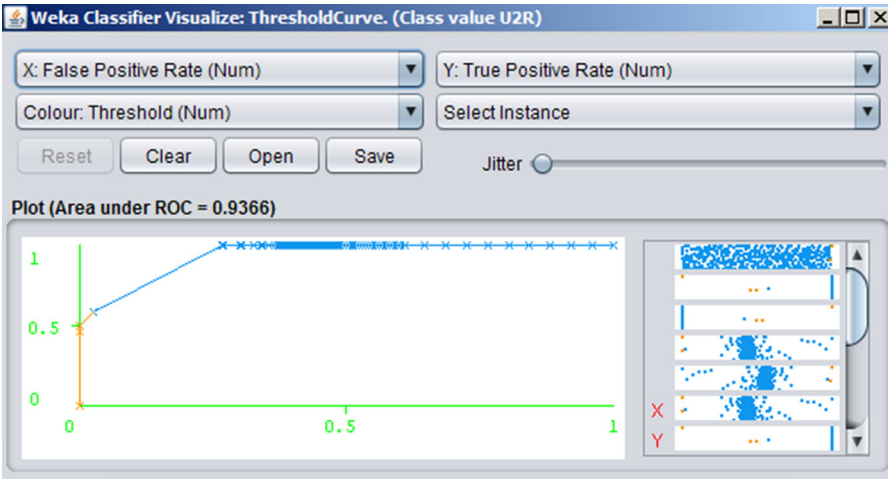


Fig. 22 ROC curve for U2R attack class

Area under ROC curve is equal to 0.9995. The ROC curve shown in Figs. 20, 21, 22, 23 and 24 is obtained using Weka tool [51, 52].

Figure 21 shows visualization of ROC curve for R2L attack class using kNN ($k = 1$) classifier. Area under ROC curve is equal to 0.9947.

Figure 22 shows visualization of ROC curve for U2R attack class using kNN ($k = 1$) classifier. Area under ROC curve is equal to 0.9366.

The ROC curve for R2L attack class is shown in Fig. 23 choosing kNN classifier with $k = 5$ for similarity threshold equal to 0.999999 for feature reduction. The area under ROC is 0.9966 for R2L class. Figure 24 depicts ROC curves for four classes: Normal, Probe, DoS and U2R.

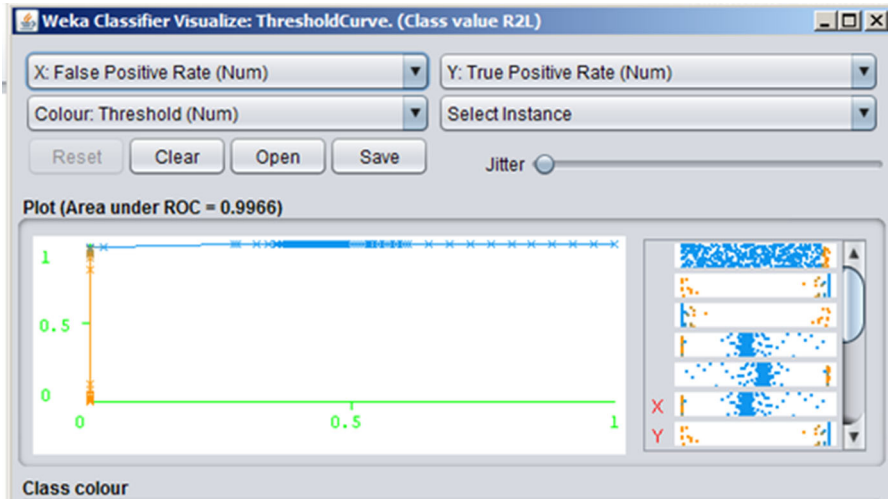


Fig. 23 ROC curve for R2L attack class for threshold—0.999999

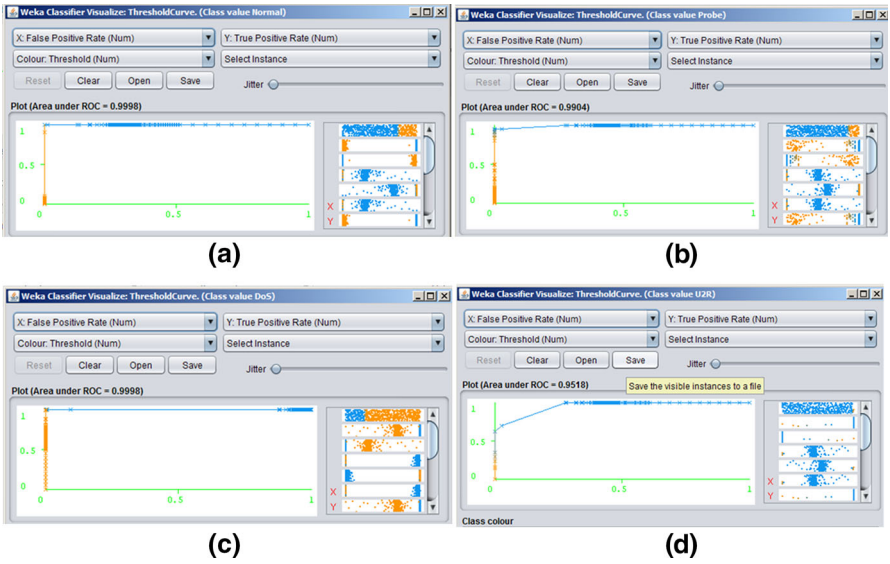


Fig. 24 ROC curves for Normal (**a**), Probe (**b**), DoS (**c**) and U2R (**d**) classes

6 Conclusions

Similarity-based incremental feature pattern clustering for feature reduction is one of the techniques that we have addressed in this research. To be specific, we concentrate on input observation transformation by projecting observation matrix on to a new dimension space. Another important contribution is the proposed dissimilarity measure that can be used to find similarity degree between any two vectors whose element

values are between 0 and 1. Experiments are performed on KDD-Cup 99 dataset with 19 and 41 attributes and NSL-KDD dataset with 19 and 41 attributes. Our approach for intrusion detection applies incremental feature clustering technique and transforms the initial observation matrix. It is better to choose a low dissimilarity threshold to obtain the optimal observation matrix for efficient intrusion detection. The optimal matrix obtained applying the proposed approach is used for evaluating classification and detection accuracies of classifiers such as *k*NN, J48, SVM, naive Bayes and Bayes net. Recorded experiment results are compared to recent approaches such as CANN, CLAPP, CSVAC and CSOCACN. The results show that the proposed approach has better classification accuracies and detection rates compared to other classifiers. The proposed approach using GARUDA measure has better detection rates on low-frequency U2R and R2L attack classes when compared to other approaches. Better classification and detection accuracies for U2R and R2L attack classes are recorded for kNN and J48 classifier, and lowest classification and detection accuracies are recorded for SVM classifier. One drawback of the proposed approach is it did not improve class accuracies obtained using SVM classifier. As a future extension for the present research, there is a scope and possibility to come up with new measures and classifiers that can improve the classification and detection accuracies of low-frequency attacks such as U2R and R2L.

References

1. Weller-Fahy DJ, Borghetti BJ, Sodemann AA (2015) A survey of distance and similarity measures used within network intrusion anomaly detection. IEEE Commun Surv Tutor 17(1):70–91
2. Lin YS, Jiang JY, Lee SJ (2014) A similarity measure for text classification and clustering. IEEE Trans Knowl Data Eng 26(7):1575–1590
3. Jiang JY, Cheng WH, Chiou YS, Lee SJ (2011) A similarity measure for text processing. In: 2011 International Conference on Machine Learning and Cybernetics, Guilin, pp 1460–1465
4. Yoo JS, Shekhar S (2009) Similarity-profiled temporal association mining. IEEE Trans Knowl Data Eng 21(8):1147–1161
5. Radhakrishna V, Kumar PV, Janaki V (2016) A novel similar temporal system call pattern mining for efficient intrusion detection. J Univers Comput Sci 22(4):475–493. https://doi.org/10.3217/jucs-022-04-0475
6. Radhakrishna V, Aljawarneh SA, Veereswara Kumar P et al (2017) ASTRA—a novel interest measure for unearthing latent temporal associations and trends through extending basic Gaussian membership function. Multimed Tools Appl. https://doi.org/10.1007/s11042-017-5280-y
7. Radhakrishna V, Veereswara Kumar P, Janaki V (2017) SRIHASS—a similarity measure for discovery of hidden time profiled temporal associations. Multimed Tools Appl. https://doi.org/10.1007/s11042-017-5185-9
8. Radhakrishna V, Aljawarneh SA, Kumar PV, Janaki V (2017) A novel fuzzy similarity measure and prevalence estimation approach for similarity profiled temporal association pattern mining. Future Gener Comput Syst. https://doi.org/10.1016/j.future.2017.03.016 **(ISSN 0167-739X)**
9. Radhakrishna V, Aljawarneh SA, Kumar PV et al (2016) A novel fuzzy Gaussian-based dissimilarity measure for discovering similarity temporal association patterns. Soft Comput. https://doi.org/10.1007/s00500-016-2445-y
10. Deza M, Deza E (2009) Encyclopedia of distances. Springer, Berlin
11. Zarpelao BB, Miani RS, Kawakani CT, de Alvarenga SC (2017) A survey of intrusion detection in Internet of Things. J Netw Comput Appl 84:25–37
12. Aljawarneh SA, Vangipuram R, Puligadda VK, Vinjamuri J (2017) G-SPAMINE: an approach to discover temporal association patterns and trends in internet of things. Future Gener Comput Syst 74:430–443

13. Sindhu SSS, Geetha S, Kannan A (2012) Decision tree based light weight intrusion detection using a wrapper approach. Exp Syst Appl 39(1):129–141
14. Lima CFL, de Assis FM, de Souza CP (2012) A comparative study of use of shannon, rényi and tsallis entropy for attribute selecting in network intrusion detection. In: Yin H, Costa JAF, Barreto G (eds) Intelligent Data Engineering and Automated Learning-IDEAL 2012. Lecture Notes in Computer Science, vol 7435. Springer, Berlin, Heidelberg
15. Singh S, Silakari S (2009) An ensemble approach for feature selection of cyber attack dataset. Int J Comput Sci Inf Secur 6(2):297–302
16. Chen RC, Cheng KF, Chen YH, Hsieh CF (2009) Using rough set and support vector machine for network intrusion detection system. In: 2009 First Asian Conference on Intelligent Information and Database Systems, Dong Hoi, pp 465–470
17. Devarakonda N, Pamidi S, Valli Kumari V, Govardhan A (2011) Outliers detection as network intrusion detection system using multi layered framework. In Advances in computer science and information technology, vol 131. Springer, Berlin, pp 101–111
18. Mabu S, Chen C, Lu N, Shimada K, Hirasawa K (2011) An intrusion-detection model based on fuzzy class-association-rule mining using genetic network programming. IEEE Trans Syst Man Cybern Part C (Appl Rev) 41(1):130–139
19. Shyu M-L, Sainani V (2009) A multiagent-based intrusion detection system with the support of multi-class supervised classification. In: Data mining and multiagent integration. Springer, Berlin, pp 127–142
20. Su M-Y, Yu G-J, Lin C-Y (2009) A real-time network intrusion detection system for large-scale attacks based on an incremental mining approach. Comput Secur 28(5):301–309. https://doi.org/10.1016/j.cose.2008.12.001
21. He X, Parameswaran S (2008) MCAD: multiple connection based anomaly detection. In: 11th IEEE Singapore International Conference on Communication Systems, Guangzhou, 2008, pp 999–1004
22. Gunupudi RK, Nimmala M, Gugulothu N, Gali SR (2017) CLAPP: a self-constructing feature clustering approach for anomaly detection. Future Gener Comput Syst 74:417–429
23. Kumar GR, Mangathayaru N, Narasimha G (2015) An improved k-means clustering algorithm for intrusion detection using Gaussian function. In: Proceedings of the International Conference on Engineering and MIS 2015 (ICEMIS'15). ACM, New York, Article 69. http://dx.doi.org/10.1145/2832987.2833082
24. Zhang C, Jiang J, Kamel M (2005) Intrusion detection using hierarchical neural networks. Pattern Recognit Lett 26(6):779–791
25. Peddabachigari S, Abraham A, Grosan C, Thomas J (2007) Modeling intrusion detection system using hybrid intelligent systems. J Netw Comput Appl 30(1):114–132
26. Özyer T, Alhajj R, Barker K (2007) Intrusion detection by integrating boosting genetic fuzzy classifier and data mining criteria for rule pre-screening. J Netw Comput Appl 30(1):99–113
27. Li Y, Guo L (2007) An active learning based TCM-KNN algorithm for supervised network intrusion detection. Comput Secur 26(7):459–467
28. Hansen JV, Lowry PB, Meservy RD, McDonald DM (2007) Genetic programming for prevention of cyberterrorism through dynamic and evolving intrusion detection. Decis Support Syst 43(4):1362–1374
29. Giacinto G, Perdisci R, Del Rio M, Roli F (2008) Intrusion detection in computer networks by a modular ensemble of one-class classifiers. Inf Fusion 9(1):69–82
30. Hu W, Hu W, Maybank S (2008) AdaBoost-based algorithm for network intrusion detection. IEEE Trans Syst Man Cybern Part B (Cybern) 38(2):577–583
31. Tajbakhsh A, Rahmati M, Mirzaei A (2009) Intrusion detection using fuzzy association rules. Appl Soft Comput 9(2):462–469
32. 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
33. Lin S-W, Ying K-C, Lee C-Y, Lee Z-J (2012) An intelligent algorithm with feature selection and decision rules applied to anomaly intrusion detection. Appl Soft Comput 12(10):3285–3290
34. Baig ZA, Sait SM, Shaeen A (2013) GMDH-based networks for intelligent intrusion detection. Eng Appl Artif Intell 26(7):1731–1740
35. Lin W-C, Ke S-W, Tsai C-F (2015) CANN: an intrusion detection system based on combining cluster centers and nearest neighbors. Knowl-Based Syst 78:13–21
36. Kumar GR, Mangathayaru N, Narasimha G (2017) A feature clustering based dimensionality reduction for intrusion detection (FCBDR). IADIS Int J Comput Sci Inf Syst 12(1):26–44

37. Kumar GR, Mangathayaru N, Narasimha G (2015) Intrusion detection using text processing techniques: a recent survey. In: Proceedings of the International Conference on Engineering and MIS 2015 (ICEMIS '15). ACM, New York, Article 55
38. Kumar GR, Mangathayaru N, Narasimha G (2015) An approach for intrusion detection using text mining techniques. In: Proceedings of the International Conference on Engineering and MIS 2015 (ICEMIS '15). ACM, New York, Article 63
39. Feng W, Zhang Q, Hu G, Huang JX (2014) Mining network data for intrusion detection through combining SVMs with ant colony networks. Future Gener Comput Syst 37(2014):127–140
40. Aljawarneh S, Aldwairi M, Yassein MB (2017) Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model. J Comput Sci. https://doi.org/10.1016/j.jocs.2017.03.006 **(ISSN 1877-7503)**
41. Kabir E, Hu J, Wang H, Zhuo G (2018) A novel statistical technique for intrusion detection systems. Future Gener Comput Syst 79(1):303–318. https://doi.org/10.1016/j.future.2017.01.029 **(ISSN 0167-739X)**
42. Wang H, Gu J, Wang S (2017) An effective intrusion detection framework based on SVM with feature augmentation. Knowl-Based Syst 136:130–139. https://doi.org/10.1016/j.knosys.2017.09.014 **(ISSN 0950-7051)**
43. Hamed T, Dara R, Kremer SC (2018) Network intrusion detection system based on recursive feature addition and bigram technique. Comput Secur 73:137–155. https://doi.org/10.1016/j.cose.2017.10.011 **(ISSN 0167-4048)**
44. Manzoor I, Kumar N (2017) A feature reduced intrusion detection system using ANN classifier. Expert Syst Appl 88(C):249–257. https://doi.org/10.1016/j.eswa.2017.07.005
45. 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 **(ISSN 1319-1578)**
46. Yang Y, Pedersen JO (1997) A comparative study on feature selection in text categorization. In: Fisher DH (ed) Proceedings of the Fourteenth International Conference on Machine Learning (ICML '97), Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp 412–420
47. Portnoy L, Eskin E, Stolfo SJ (2001) Intrusion detection with unlabeled data using clustering. In: Proceedings of ACM CSS Workshop on Data Mining Applied to Security (DMSA-2001), Philadelphia, PA, USA
48. Eskin E, Arnold A, Prerau M, Portnoy L, Stolfo SJ (2002) A geometric framework for unsupervised anomaly detection: detecting intrusions in unlabeled data. In: Data mining for security applications. Kluwer, Boston
49. Tavallaee M, Bagheri E, Lu W, Ghorbani AA (2009) A detailed analysis of the KDD CUP 99 data set. In: Proceedings of the 2009 IEEE Symposium on Computational Intelligence in Security and Defense Applications (CISDA 2009), pp 1–6
50. McHugh J (2000) Testing intrusion detection systems: a critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory. ACM Trans Inf Syst Secur 3(4):262–294
51. Hall M, Frank E, Holmes G, Pfahringer B, Reutemann P, Witten IH (2009) The WEKA data mining software: an update. SIGKDD Explor Newslett 11(1):10–18. https://doi.org/10.1145/1656274.1656278
52. https://www.cs.waikato.ac.nz/ml/weka/index.html
53. Xue-qin Z, Chun-hua G, Jia-jun L (2006) Intrusion detection system based on feature selection and support vector machine. In: 2006 First International Conference on Communications and Networking in China, Beijing, pp 1–5