Information Sciences 378 (2017) 484–497

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Fuzziness based semi-supervised learning approach for intrusion detection system

Rana Aamir Raza Ashfaq[a], Xi-Zhao Wang[a,*], Joshua Zhexue Huang[a], Haider Abbas[b], Yu-Lin He[a]

[a] College of Computer Science & Software Engineering, Shenzhen University, Shenzhen 518060, Guangdong, China
[b] King Saud University, Riyadh, Saudi Arabia

ARTICLE INFO

Article history:
Received 21 August 2015
Revised 17 March 2016
Accepted 6 April 2016
Available online 3 May 2016

Keywords:
Fuzziness
Divide-and-conquer strategy
Semi-supervised learning
Intrusion detection
Random weight neural network

ABSTRACT

Countering cyber threats, especially attack detection, is a challenging area of research in the field of information assurance. Intruders use polymorphic mechanisms to masquerade the attack payload and evade the detection techniques. Many supervised and unsupervised learning approaches from the field of machine learning and pattern recognition have been used to increase the efficacy of intrusion detection systems (IDSs). Supervised learning approaches use only labeled samples to train a classifier, but obtaining sufficient labeled samples is cumbersome, and requires the efforts of domain experts. However, unlabeled samples can easily be obtained in many real world problems. Compared to supervised learning approaches, semi-supervised learning (SSL) addresses this issue by considering large amount of unlabeled samples together with the labeled samples to build a better classifier. This paper proposes a novel fuzziness based semi-supervised learning approach by utilizing unlabeled samples assisted with supervised learning algorithm to improve the classifier's performance for the IDSs. A single hidden layer feed-forward neural network (SLFN) is trained to output a fuzzy membership vector, and the sample categorization (low, mid, and high fuzziness categories) on unlabeled samples is performed using the fuzzy quantity. The classifier is retrained after incorporating each category separately into the original training set. The experimental results using this technique of intrusion detection on the NSL-KDD dataset show that unlabeled samples belonging to low and high fuzziness groups make major contributions to improve the classifier's performance compared to existing classifiers e.g., naive bayes, support vector machine, random forests, etc.

© 2016 Elsevier Inc. All rights reserved.

1. Introduction

Intrusion detection (ID) is a process of monitoring, detecting, and analyzing the events that are considered as violation to the security policies of a networked environment [45]. Denning [12] introduced the concept of detecting cyber-based attacks on computer networks by providing a framework for intrusion detection system (IDS), which is based on the hypothesis that security violations can be detected by monitoring system audit records for abnormal patterns of system usage. Organizations deploy their own access controls to grant or restrict the level of access for their assets but this approach

* Corresponding author.
E-mail addresses: aamir@szu.edu.cn (R.A.R. Ashfaq), xizhaowang@ieee.org, xwang@szu.edu.cn (X.-Z. Wang), zx.huang@szu.edu.cn (J.-Z. Huang), hsiddiqui@ksu.edu.sa (H. Abbas), yulinhe@szu.edu.cn (Y.-L. He).

http://dx.doi.org/10.1016/j.ins.2016.04.019
0020-0255/© 2016 Elsevier Inc. All rights reserved.

does not guarantee the appropriate assurance and protection level for a particular resource [10]. This problem is evident through various security incidents around the world, for example, the compromise of Yahoo's and Amazon's websites after some sophisticated persistent attacks [6]. Intruders and attackers are always seeking to disrupt network traffic and degrade network performance with different types of attacks or intrusions. A network intrusion refers to a suspicious and sudden deviation from the normal behavior of the system, which destabilizes the security of the network system. According to Qui et al. [40], Hernndez-Pereira et al. [16] and Yan and Yu [56], intrusion can be depicted as the set of actions that attempt to compromise the confidentiality, integrity, or availability (CIA) of information resources; therefore, it is necessary to take different measures to minimize such risks.

The Internet has turned into an indispensable wellspring for exchanging information among users and organizations; therefore, security has become an essential aspect in this type of communication. IDSs are often used to sniff network packets by providing a better understanding of what is happening in a particular network. Two mainstream preferences for IDSs are (1) host-based IDSs, and (2) network-based IDSs. Correspondingly, the detection methods used in IDS are anomaly based and misuse based (also called signature or knowledge based), each having their own advantages and restrictions. In misuse-based detection, data gathered from the system is compared to a set of rules or patterns, also known as signatures, to describe network attacks. The core difference between these two techniques is that anomaly-based IDS uses collections of data containing examples of normal behavior and builds a model of familiarity, therefore, any action that deviates from the model is considered suspicious and is classified as an intrusion [20]. According to Mukkamala et al. [31], in misuse-based detection, attacks are represented by signatures or patterns. However, this approach does not contribute much in terms of zero-day attack detection. The main issue is how to build permanent signatures that have all the possible variations and non-intrusive activities to lower the false-negative and false-positive alarms.

The KDDCUP'99 [18] was derived in 1999 from the DARPA98 network traffic dataset and a very popular benchmark dataset used in the International Knowledge Discovery in Databases (KDD) competition. From the literature, one can study that this dataset is widely used for the evaluation of anomaly based IDS. Many machine learning techniques, which may be either supervised or unsupervised, have been used to increase the efficacy of IDSs. Supervised learning techniques are applied to obtain the training data in which instances are tagged with labels and each label indicates the class of a particular instance. Many supervised algorithms, such as *k*-nearest neighbor (KNN) [24], neural network (NN) [29], and support vector machine (SVM) [30] have been extensively used to detect the intrusions. These algorithms build the model that separates a new unseen example or instance with the correct label. Many advantages and disadvantages related to supervised learning with IDS have been reported by many researchers. One of the shortcomings of supervised learning is the need for labeled instances. The only dataset is available for ID is the KDDCUP'99 dataset [18], and many new types of attacks have been developed. Therefore, this dataset is considered as obsolete, and for new types of examples its accuracy drops [22]. Many researchers are widely using the KDDCUP'99 dataset because it is the only dataset that is publically available for ID problem and useful information can still be extracted from it. Apart from its disadvantage, supervised learning has the advantage to achieve better accuracy to classify similar examples [22]. Unsupervised learning techniques deal with the learning tasks with unlabeled or untagged data. Clustering is the most popular unsupervised learning technique [25]. In clustering, the learning algorithm finds similarities among instances to build the clusters (i.e. group of instances). Instances that belong to the same cluster are assumed to having similar characteristics or properties and then are assembled into the same class. The disadvantage of unsupervised learning is the manually assignment of cluster numbers, which results in low accuracy in predictions. However, it has the advantage of detecting new examples better than supervised learning techniques, and considered to be more robust in IDSs. According to Laskov et al. [22], many new attacks have been developed, and the improper labeling of examples could make the unsupervised learning and SSL techniques the best choices for improving the accuracy of IDSs.

Regarding the aforementioned development in this area, the main objective behind our work is not just to seek for the smallest classification error but also to try to find a model that must be capable of incorporating new data that keeps its good generalization ability. We compute the fuzziness of every unlabeled sample outputted by the classier, and try to discover its relationship with misclassification. From literature, except for [51,53], we have not found any studies on generalization based on the fuzziness of a classifier. Therefore, based on our preliminary work [51] in which the sample categorization is performed according to the fuzziness quantity, we propose a new algorithm for the IDS. The experimental results demonstrate that samples belonging to the low and high fuzziness categories play an important role in improving the accuracy of IDSs.

The rest of the paper is organized as follows. Section 2 presents a prologue to the background of semi-supervised learning (SSL). Section 3 details the proposed fuzziness based algorithm using the neural network with random weights (NNR_w). The performance evaluation is presented in Section 4. Finally, Section 5 ends this paper with concluding remarks, and provides future directions for this research.

2. Semi-supervised learning (SSL)

SSL is an amalgamation of supervised and unsupervised learning techniques. The SSL technique deals with the learning tasks by utilizing both labeled and unlabeled data [65]. Labeled instances are, however, expensive and time-consuming to obtain and require the efforts of domain experts. Apart from this concern, unlabeled data can easily be obtained in many real world applications. SSL methods assign labels by considering unlabeled instances, together with the labeled instances,

and then build a better classifier. Many SSL methods such as self-training, co-training, transductive support vector machines (TSVMs), expectation maximization (EM) with generative mixture models, and graph-based methods have attracted much attention from researchers.

2.1. Self-training

Self-training is one of the earliest SSL techniques. A classifier C is used to classify the unlabeled data U and the most confident unlabeled samples with their predicted labels are incorporated into the training set Tr. The classifier C is again retrained, and the process is repeated. In the whole process, C uses its own prediction to learn or teach itself; hence, the process is named as self-teaching or bootstrapping. Yarowsky [57] used the self-training algorithm for the word sense disambiguation in which the decision was made regarding either the word *"plant"* represents a *living organism* or a *factory*. Riloff et al. [41] used the self-training algorithm to recognize the subjective nouns. Rosenberg et al. [42] utilized the self-training algorithm for object detection from the images, and also proved that, the SSL technique compares favorably with a state-of-the-art detector.

2.2. Co-training

Another well-known SSL technique is co-training [5], which assumes that input features can be split into two dissimilar views, V_1 and V_2, respectively. In other words, co-training deals with the task whose input space has two independent views. It works in an iterative manner, where two separate classifiers, *C1* and *C2*, are trained with Tr on V_1 and V_2, respectively and classify the U. Co-training starts its process at using a weak initial hypothesis over one feature set and labeling samples. These examples may be randomly distributed to the other classifier under the assumption of conditional independence, and the classification noise from the weak hypothesis would be brought to the other classifier. Thus, the algorithm can learn from labeled samples by an iterative manner between the two classifiers.

2.3. Generative models

Generative models are the oldest SSL methods, which assumes a model where $p(x|y)$ is an identifiable mixture distributions [64]. Nigam et al. [34] applied the EM algorithm to a mixture of multinomial for the text classification. EM is a generative model and uses the likelihood-based approach. Baluja [3] used the same algorithm in a face orientation discrimination task. Nigam and Ghani [33] conducted an experiment to compare co-training and the EM algorithm. They proposed a new algorithm called co-EM, which combines the characteristics of both co-training and EM. Fujino et al. [13] extended generative mixture models by adding the term *bias correction* and discriminative training using the maximum entropy principle.

From the literature [21], we can find that the earliest work on ID was based on SSL in which a partially observable Markov decision-making process (POMDM) was modeled in order to classify user behavior in Unix terminal and used the SSL mechanism called "Expectation-Maximization to learn conditional probability distribution".

2.4. Graph based methods

Graph based SSL methods define a graph on which nodes represent labeled and unlabeled examples and edges reflect the similarity among the examples. These types of methods are non-parametric, discriminative, and transductive [64]. According to Blum and Chawla [4], SSL can be portrayed as a graph mincut or st-cut. In a two-class problem, positive and negative examples act as sources and sinks, respectively. Hence, the nodes connecting to sources are labeled as positive, and those connecting to sinks are labeled as negative. The main problem with mincut is that, it provides only hard classification without confidence because it does not compute marginal probabilities but rather computes the mode. Pan et al. [37] proposed a novel SSL method for visual objects classification. In their proposed method they simultaneously maximize the separability between different classes and estimate the intrinsic geometric structure of the data by using both the labeled and unlabeled instances. Zhou et al. [63] proposed a general framework for SSL on a directed graph, and the structure of the graph along with the direction of the edges is considered. Their algorithm takes the directed graph and the label set as input, and a function is computed using the labeled vertices to classify the unlabeled vertices. Therefore, in the absence of labeled instances, this function can be used as a spectral clustering for the directed graph.

Recently, Pan et al. [36] presented a new type of SSL mechanism based on regularization term that utilizes the label information of a dataset for various kernel learning algorithms. Their proposed method provides an efficient use of label information by adding a new regularization term to the objective functions of the optimization problems. Chen et al. [8] proposed a novel SSL method for clustering, where they evaluated the application of spectral graph transduction and Gauss random fields for the detection of unknown attacks. One can study in the literature [14,26,52,62] that many SSL methods related to clustering have been proposed. In [14], the authors proposed a novel SSL technique called stable semi-supervised discriminant learning (SSDL) for dimensionality reduction. In this approach, they constructed two adjacency graphs to learn the intrinsic structure that characterizes the local topology, and geometrical properties of the similarity and the diversity of the data. They incorporated it into an objective function of linear discriminant analysis (LDA).

2.5. Transductive support vector machines (TSVMs)

TSVM originated from the intention to work only on observed data that builds the connection between $p(x)$ and the discriminative decision boundary by not putting the density in the high density region. TSVM is an extension of the SVM but with unlabeled data aiming to find labels for unlabeled examples so that the linear boundary has the maximum margin on both the original labeled examples and the newly labeled examples [64]. Vapnik in [50] introduced the notion of transductive inference, which is regarded as an approach of SSL. Many success rates have been reported in these studies, but the method has faced some criticism because it cannot perform well under some circumstances.

From the literature [28,47], one can find that many SSL techniques have been proposed for improving the accuracy of a classifier. Chen et al. [9] proposed a variant of Laplacian smooth twin SVM (Lap-STSVM) and experimentally demonstrated the accuracy level comparable to that of Lap-SVM at less computational cost. Recently Qi et al. [39] proposed a new SSL method called semi-supervised learning using privileged information (Semi-LUPI). Their method improves the classifier performance by utilizing the geometry information of the marginal distribution embedded in unlabeled data and the privileged information to improve the efficiency of the learning. Maulik and Chakaraborty [27] proposed a novel semi-supervised SVM classification technique that exploits both labeled and unlabeled data points by considering the problem of pixel classification of remote sensing images and applied the marginal maximization principle to both labeled and unlabeled patterns. They experimentally confirmed that their learning scheme removes unnecessary points to a great extent from the unlabeled data and increases the accuracy level.

In the next section, we present our proposed algorithm, which relies on fuzziness outputted by the classifier on a group of samples (unlabeled). Our algorithm can be utilized in an effective way for the improvement of classifier performance and to have lower computational cost.

3. Proposed fuzziness based algorithm using NNR_w for IDS

In this section, we first discuss the fuzziness and introduce a fast learning mechanism for a single hidden layer feedforward neural network (SLFN), i.e., neural network with random weights (NNR_w), and then propose an algorithm for IDS.

3.1. Fuzziness

The term *fuzziness* refers to the unclear boundary between two linguistic terms and is based on the membership function of fuzzy sets. It was first mentioned by Zadeh [60] in 1965. Zadeh also generalized the probability measure of an event to a fuzzy event and suggested using entropy in information theory to interpret the uncertainty associated with a fuzzy event. Authors in [11] considered fuzziness to be a type of uncertainty and also defined a quantitative measure of fuzziness with non-probabilistic entropy analogous to Shannon's information entropy. They also proposed three properties that fuzziness should hold. These properties depict that, the fuzziness degree should attain its maximum when the membership degree of every element is equal and its minimum when every element either belongs to the fuzzy set or absolutely not. In this study, We consider fuzziness as a type of cognitive uncertainty, coming from the transition of uncertainty from one linguistic term to another, where a linguistic term is a fuzzy set defined in a certain universe of discourse. As stated in the literature [43], the fuzziness of a fuzzy set can be measured by a function $F \rightarrow [0, 1]^X$ satisfying the following axioms.

- 1 : $F(\mu) = 0$ if and only if μ is a crisp set.
- 2: $F(\mu)$ gets its maximum value if and only if $\mu(x) = 0.5\ \forall x \in X$.
- 3: if $\mu \leq_S \sigma$ then $F(\mu) \geq F(\sigma)$.
- 4 : $F(\mu) = F(\mu')$, where $\mu'(x) = 1 - \mu(x)$ for $\forall x \in X$.
- 5 : $F(\mu \cup \sigma) + F(\mu \cap \sigma) = F(\mu) + F(\sigma)$.

Axioms (1–3) were already proposed by De Luca and Termini [11]. In order to measure the fuzziness, it is necessary to know when a fuzzy set is less than another. The sharpened order $\leq_S$ is proposed in [11] as

- $\mu \leq_S \sigma \Leftrightarrow \min(0.5, \mu(x)) \geq \min(0.5, \sigma(x))\ \&\ \max(0.5, \mu(x)) \leq \max(0.5, \sigma(x))$.

Definition 1. Let $V = \{\mu_1, \mu_2, \ldots, \mu_n\}$ be a fuzzy set. According to De Luca and Termini [11], the fuzziness of V can be defined as

$$F(V) = -\frac{1}{n}\sum_{i=1}^{n}(\mu_i \log \mu_i + (1-\mu_i)\log(1-\mu_i)) \tag{1}$$

Many equations similar to Eq. (1) can be constructed, e.g., when $n = 2$ and V is normalized i.e., $\mu_1 + \mu_2 = 1$, we can have

$$F_1(V) = 1 - \mu_1^2 - (1-\mu_1)^2 \tag{2}$$

$$F_2(V) = \begin{cases} \dfrac{\mu_1}{1-\mu_1} & 0 \leq \mu_1 \leq 0.5 \\ \dfrac{1-\mu_1}{\mu_1} & 0.5 \leq \mu_1 \leq 1 \end{cases} \tag{3}$$

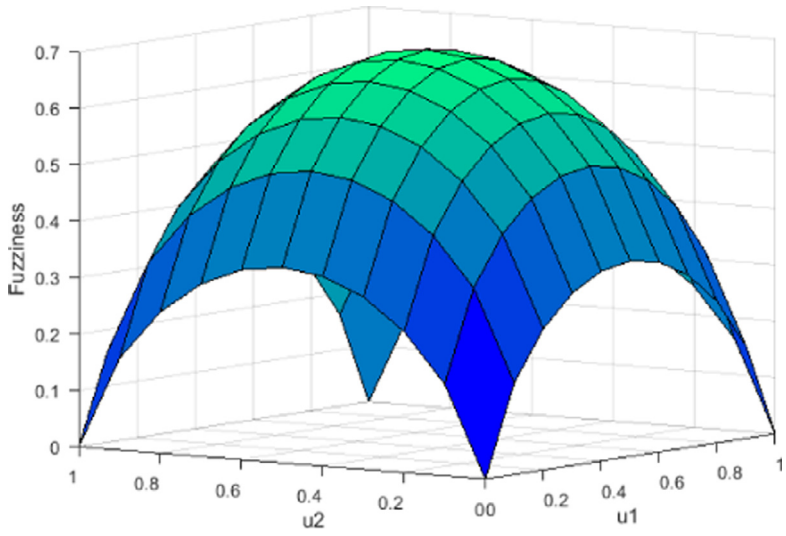


Fig. 1. Fuzziness equation for binary class.

It is easy to verify that Eqs. (1)–(3) satisfy the above mentioned axioms $(1-5)$. The fuzziness of fuzzy set defined by Eq. (1) attains its maximum when the membership degree of every element is $\mu_i = 0.5$ for every $i = 1, 2, \ldots, n$ and minimum when every element belongs to the fuzzy set or absolutely not for every $\mu_i = 0$ or $\mu_i = 1$, $i(1 \le i \le n)$.

For a binary class problem, Eq. (1) can be plotted as Fig. 1, where we can see that the fuzziness of V for a binary class attains its maximum at (0.5, 0.5) and minimum at four points i.e., (0, 0), (1, 1), (0, 1) and (1, 0).

Now we associate fuzziness with the output of a classifier. One can study in the literature [19,23,54,58,59] that many classifiers have an output analogous to fuzzy vector in which each vector's component represents the membership degree of testing sample belongs to a particular class C. For a given training set $\{X_i\}_{i=1}^{N}$, a fuzzy partition of these samples assigns the membership degree of every sample to the C classes. Thus the partition can be denoted by the membership vector $U = (U_{ij})_{(C\times N)}$. The elements of the membership matrix U have to conform the following property.

$$\sum_{i=1}^{C} \mu_{ij} = 1, 0 < \sum_{j=1}^{N} \mu_{ij} < N, \mu_{ij} \in [0, 1]$$

Where $\mu_{ij} = \mu_i(x_j)$ represents the membership of the jth sample x_j belongs to the ith class. When a classifier completes its training process, its membership matrix U can be obtained. The classifier will give an output in the form of fuzzy vector for every jth sample during the testing phase. Based on Eq. (1), for every jth sample the fuzziness of every output vector can be obtained by following Eq. (4).

$$F(\mu_j) = -\frac{1}{C}\sum_{i=1}^{C} (\mu_{ij} \log \mu_{ij} + (1-\mu_{ij}) \log(1-\mu_{ij})) \tag{4}$$

3.2. Neural network with random weights (NNR_w)

Schmidt et al. [46] are the pioneers who earlier studied the impact of random initialization on generalization performance of SLFN in 1992. They experimentally demonstrated that SLFN can obtain a better performance by choosing random weights associated with the input layer and by analytically computing the weights of the output layer. This is the first research regarding the non-iterative training of NN using randomization. The researchers also concluded that, in SLFN, the weights of the output layer are significantly most important than the weights found in the hidden layer. Yam et al. [55] used this approach for initializing the weights of NN before training it with back-propagation (BP). Similarly, in [7,15,49,61], authors introduced NN with a randomly initialized hidden layer and trained using pseudo-inverse. From the literature [2,17,38,44], one can see that several ideas have been proposed for randomization of the hidden layer in NN, such as the Random Vector Functional Link (RVFL) network, which incorporates random hidden-layer weights and biases, and the direct connection between the input layer and the output layer. Schmidt et al. [46] did not propose a name for their proposed SLFN; hence, to recognize their work, we use the neural network with random weights (NNR_w). The NNR_w discussed in our study has no direct connection between the nodes of the input layer and the output layer, and we consider only the biases for the hidden layer nodes. The output of SLFN with L hidden layer nodes can be represented as

$$f(\mathbf{x}) = \sum_{i=1}^{L} \tilde{\beta}_i g(\mathbf{w}_i, b_i, \mathbf{x}), \mathbf{x} \in \mathbb{R}^n \tag{5}$$

In Eq. (5), $\mathbf{w}_i \in \mathbb{R}^n$ and $b_i \in \mathbb{R}$ represent input weights and biases at hidden layer nodes respectively, $\tilde{\beta}_i \in \mathbb{R}^m$ is the output weight and $\tilde{\beta}_i g(\mathbf{w}_i, b_i, \mathbf{x})$ is the output of ith hidden node w.r.t input $\mathbf{x}$.

Therefore, for a given dataset $\{(\mathbf{x}_i, \mathbf{t}_i)\}_{i=1}^{N} \subset \mathbb{R}^n \times \mathbb{R}^m$, where $\mathbf{x}_i$ is an input vector, and t_i is corresponding observed vector. SLFN with L hidden nodes approximating these N training samples with zero error means that their exist $\tilde{\beta}_i$, $\mathbf{w}_i$, and b_i where $i = 1, \ldots, L$ such that

$$\sum_{i=1}^{\mathrm{L}} \tilde{\beta}_i g(\mathbf{w}_i, b_i, \mathbf{x}_j) = \mathbf{t}_j, \; j = 1, \ldots, N \tag{6}$$

Hence the Eq. (6) can be written compactly as

$$\mathbf{H}\tilde{\beta} = \mathbf{T} \tag{7}$$

where $\mathbf{H}_{N\times L} = \begin{pmatrix} g(\mathbf{w}_1, b_1, \mathbf{x}_1) & g(\mathbf{w}_2, b_2, \mathbf{x}_1) & \cdots & g(\mathbf{w}_L, b_L, \mathbf{x}_1) \\ g(\mathbf{w}_1, b_1, \mathbf{x}_2) & g(\mathbf{w}_2, b_2, \mathbf{x}_2) & \cdots & g(\mathbf{w}_L, b_L, \mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ g(\mathbf{w}_1, b_1, \mathbf{x}_N) & g(\mathbf{w}_2, b_2, \mathbf{x}_N) & \cdots & g(\mathbf{w}_L, b_L, \mathbf{x}_N) \end{pmatrix}$, $\tilde{\beta}_{L\times m} = \begin{pmatrix} \tilde{\beta}_1 \\ \tilde{\beta}_2 \\ \vdots \\ \tilde{\beta}_L \end{pmatrix}^T$ *and* $\mathrm{T}_{N\times m} = \begin{pmatrix} \mathbf{t}_1 \\ \mathbf{t}_2 \\ \vdots \\ \mathbf{t}_N \end{pmatrix}^T$

$\mathbf{H}$ is a hidden layer output matrix with respect to the input vectors x_i, where $i = 1, \ldots, N$, and $g(z) = \frac{1}{1+e^{-z}}$ is sigmoid activation function. Therefore, Eq. (7) becomes a system of linear equations, which in most cases can be transferred to a regular system of linear equations.

$$\mathbf{H}^T\mathbf{H}\tilde{\beta} = \mathbf{H}^T\mathrm{T} \tag{8}$$

Suppose that $\mathbf{H}^T\mathbf{H}$ is non-singular, the solution of system according to Eq. (8) can be expressed as

$$\tilde{\beta} = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathrm{T} \tag{9}$$

Algorithm $\mathrm{NNR_w}$: For a given dataset $\mathrm{X} = \{(\mathbf{x}_i, \mathbf{t}_i) | \mathbf{x}_i \in \mathbb{R}^n, \mathbf{t}_i \in \mathbb{R}^m, i = 1, \ldots, N\}$ and a hidden node output function $g(\mathbf{w}, b, \mathbf{x})$, and number of hidden nodes L

1. Randomly select the input parameters $\mathbf{w}_i$ and b_i where $i = 1, \ldots, L$
2. Compute the hidden layer output matrix $\mathbf{H}$.
3. By using Eq. (9), calculate the output weight $\tilde{\beta}$

The parameters $\mathbf{w}_i$ and b_i at the hidden layer of SLFNs are assigned randomly and independently according to Igelnik and Pao [17], where specified ranges of these random parameters are suggested.

In this paper, we specify the ranges of the random parameters in the form of $[0, \theta]$, where the scope parameter θ is data dependent and should be adjusted to gain favorable performance for a given dataset.

3.3. Fuzziness based divide-and-conquer strategy

Recently Wang et al. [51] proposed a new algorithm based on a divide-and-conquer strategy. In their methodology, all testing samples *Ts* were categorized into three groups according to the magnitude of fuzziness, and the group with highest accuracy was incorporated into the original training set *Tr*. Retraining was performed with the new training set *Tr*'. Their proposed technique is regarded as an approach to SSL in which some samples with unknown labels having low fuzziness participate in the training process. The key steps of the algorithm proposed by Wang et al. [51] are mentioned in Table 1.

3.4. Proposed algorithm

Based on our previous work [51], we extend the divide-and-conquer methodology and propose a new algorithm for IDSs by using $\mathrm{NNR_w}$. The proposed algorithm used is described in Table 2. For a given dataset *Tr* of labeled examples, a dataset

Table 1
Fuzziness based divide-and-conquer methodology.

For a given training set *Tr*, testing set *Ts*, and classifier *C*,
1. Obtain the fuzzy membership vector output by the classifier *C* on *Ts*.
2. Compute the fuzziness for every output vector and also obtain the training accuracy $\mathrm{Tr}_{Accuracy}$ and testing accuracy $\mathrm{Ts}_{Accuracy}$, respectively.
3. Group the output samples of *Ts* into low fuzziness group G_{low}, mid fuzziness group G_{mid}, and high fuzziness group G_{high}.
4. Add the group having the highest accuracy into *Tr* and obtain new training set *Tr*'.
5. Retrain the classifier *C* on *Tr*' and get the training accuracy $Tr'_{Accuracy}$ and testing accuracy $\mathrm{Ts}_{Accuracy}$.
6. Compare the accuracies obtained in steps 3 and 5.

Table 2
Proposed algorithm using NNR_w.

Input:
- Tr : Labeled dataset $(x_i, y_i | 1 \le i \le N)$.
- U : Unlabeled dataset $(u_i, | 1 \le i \le U)$.
- Ts : Test dataset $(t_i, y_i | 1 \le i \le K)$.
- Classifier: C_{NNR_w}.
- L : Number of hidden nodes.
- Hidden-node output function: $g(z) = \frac{1}{1+e^{-z}}$.

Output:
- $\mathbf{Ts}_{Accuracy}$: Testing accuracy.

Process:
- $F' = C_{NNR_w}\ (Tr)$.
- Generate F' (U).
- Obtain membership vector V of every unlabeled example from F' (U).
- Compute fuzziness $F(V)$ of every sample in U.
- Sample categorization FG_{low}, FG_{mid}, and FG_{high}.
- $Tr_{new} = Tr + (FG_{low} + FG_{high})$.
- $F' = C_{NNR_w}(Tr_{new})$.
- Generate $F'(Ts)$.

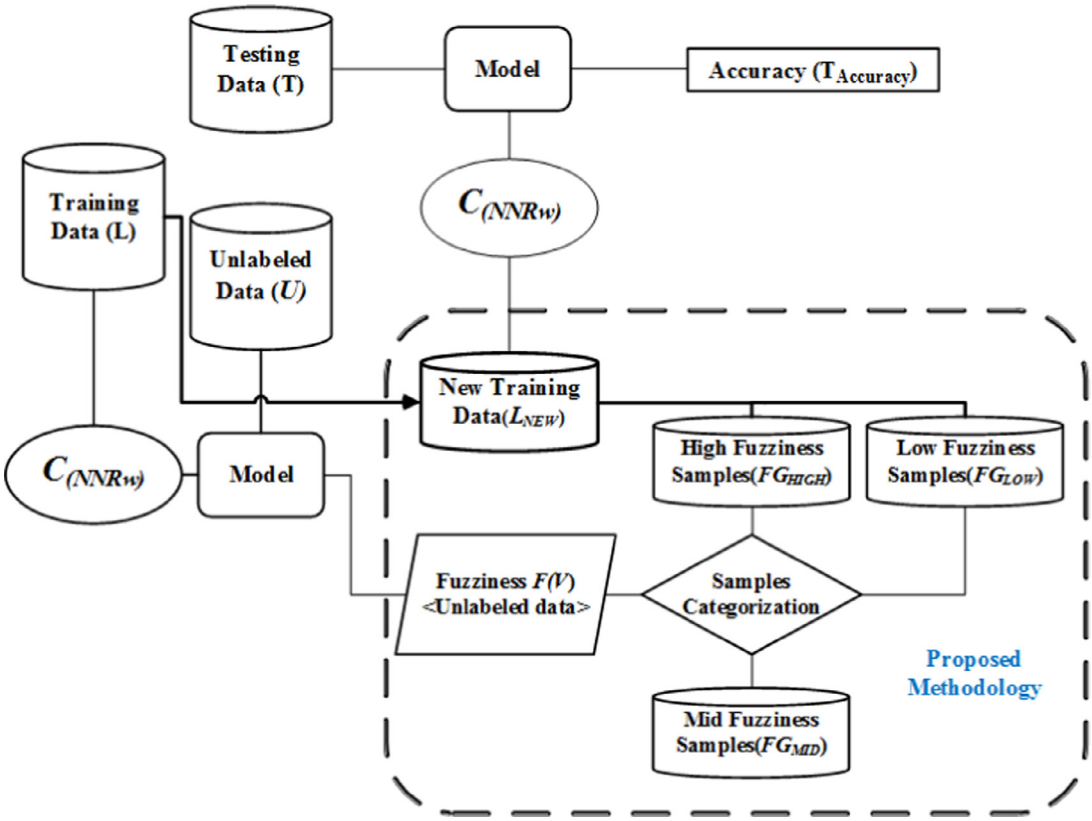


Fig. 2. Flow chart of proposed methodology.

U of unlabeled examples, and a testing dataset Ts. The algorithm first uses Tr to train the classifier NNR_w by using L hidden nodes. The hidden node output function used by the algorithm is the sigmoid activation function. Second, a membership vector V is obtained on every unlabeled sample by examining U using classifier C_{NNR_w}. The membership vector of each unlabeled sample that we get during this process is further utilized to obtain the fuzziness $F(V)$ by using Eq. (1). Based on the fuzziness value we categorize the samples into three groups, i.e., low fuzziness group FG_{low}, mid fuzziness group FG_{mid}, and high fuzziness group FG_{high} respectively, and extract those samples that belong to the FG_{low} and FG_{high} fuzziness groups. These two groups are further incorporated with Tr to obtain an updated dataset Tr_{new} for retraining the C_{NNR_w}. The flow chart of the proposed algorithm is depicted in Fig. 2.

4. Performance evaluation

Tavallaee et al. [48] statistically discovered and figured out some shortcomings in the original KDDCUP'99 dataset that adversely affect the performance of the evaluated system and provide inefficient anomaly detection schemes. They proposed an enhanced dataset called NSL-KDD [35] to counter these issues and provided a more efficient scheme for comparing various ID models. They also mentioned some advantages of the new dataset over the original dataset: (1) non-availability of redundant records in the training dataset so that a classifier has low bias toward frequent records and reduces the complexity level, (2) no duplicate records in the testing datasets, and (3) the number of records in both training and testing datasets are quite sensible, making it reasonable to run the experiment on a complete set without the need to randomly select a small subset or portion. They also compared the performance of the NSL-KDD dataset on different classifiers i.e., naive bayes, SVM, and random forests etc., where they used *KDDTrain_20percent* dataset for training and two datasets i.e., $KDDTest^{+}$ and $KDDTest^{-21}$, for testing.

4.1. Data specification

The NSL-KDD [35] dataset incorporates 41 *input features* as in original KDDCUP'99 [18] dataset and a *class* label. These features are essentially classified into three different categories, (1) features that are extracted from TCP/IP connection without inspecting the payload called *basic features*, (2) features for accessing the payload of TCP packet and necessary to observe the suspicious behavior within the payload segment called *content features*, (3) traffic features utilize with 2 s temporal window (*time based traffic features*), and historical window instead of time (*host based traffic features*) that are designed to assess attack within interval longer than 2 s. Features $1-9$ represent the basic features, $10-22$ illuminate contents features, $23-31$ and $32-41$ concentrate on time based traffic features and host based traffic features respectively. The description and overall picture of the dataset according to the input features are listed in Table 3.

A *class* label, which specifies the status of an instance is either normal or attack, there are different types of attacks listed in original KDDCUP'99 dataset. Detail of these attacks with their categories is listed in Table 4.

Table 3
Description of input features.

	#	Input feature	Data type		#	Input feature	Data type
Basic features	1	duration	Continuous	Time based traffic features	23	Count	Continuous
	2	protocol_type	Symbolic		24	srv_count	Continuous
	3	service	Symbolic		25	serror_rate	Continuous
	4	flag	Symbolic		26	srv_error_rate	Continuous
	5	src_bytes	Continuous		27	rerror_rate	Continuous
	6	dst_bytes	Continuous		28	srv_rerror_rate	Continuous
	7	land	Symbolic		29	same_srv_rate	Continuous
	8	wrong_fragment	Continuous		30	diff_srv_rate	Continuous
	9	urgent	Continuous		31	srv_diff_host_rate	Continuous
	10	hot	Continuous				
Contents features	11	num_failed_logins	Continuous	Host based traffic features	32	dst_host_count	Continuous
	12	logged_in	Symbolic		33	dst_host_srv_count	Continuous
	13	num_compromised	Continuous		34	dst_host_same_srv_rate	Continuous
	14	root_shell	Continuous		35	dst_host_diff_srv_rate	Continuous
	15	su_attempted	Continuous		36	dst_host_same_src_port_rate	Continuous
	16	num_root	Continuous		37	dst_host_srv_diff_host_rate	Continuous
	17	num_file_creations	Continuous		38	dst_host_serror_rate	Continuous
	18	num_shells	Continuous		39	dst_host_srv_serror_rate	Continuous
	19	num_access_files	Continuous		40	dst_host_rerror_rate	Continuous
	20	num_outbound_cmds	Continuous		41	dst_host_srv_rerror_rate	Continuous
	21	is_hot_login	Symbolic				
	22	is_guest_login	Symbolic				

Table 4
Attack types.

Deniel of service (DoS)	User to root (U2R)	Remote to local (R2L)	Probing (PROBE)
Back	Perl	FTP write	IP sweep
Ping of death	Buffer overflow	Guess password	NMAP
Neptune	Load module	IMAP	Port sweep
Smurf	Rootkit	Multi HOP	Satan
Land		Phf	
Teardrop		SPY	
		Wareclient	
		Warezmaster	

Table 5
Symbolic features with values.

Symbolic features	Nominal values	#. of distinct categories
protocol_type	tcp, udp, icmp	3
service	smtp, ntp_u, shell, kshell, imap4, urh_i, netbios_ssn, ftp_u, mtp, uucp, nnsp, echo, tim_i, ssh, iso_tsap, time, netbios_ns, systat, hostnames, login, efs, supdup, http_8001, courier, ctf, finger, nntp, ftp_data, red_i, ldap, http, ftp, pm_dump, exec, klogin, auth, netbios_dgm, ...	66
flag	RSTR, S3, SF, RSTO, SH, OTH, S2, RSTOS0, S1, S0, REJ	11
land	0 and 1	2
logged_in	0 and 1	2
is_host_login	0 and 1	2
is_guest_login	0 and 1	2

Table 6
Clustering of *flag* feature.

Flag cluster	Flag category	Description
FG1	S0	Connection attempt seen, no reply
	REG	Connection attempt rejected
FG2	S1	Connection established but not terminated
	SF	Regular establishment and termination
	OTH	No SYN seen, midstream traffic
FG3	S2	Connection established and closed attempt seen by originator
	RSTO	Connection established, originator aborted
FG4	S3	Connection established and close attempt seen by responder
	RSTR	Connection established, responder aborted
FG5	RSTOSO	Originator sent a SYN followed by a RST, SYN ACK not seen by the responder
	SH	Originator sent a SYN followed by a FIN, SYN ACK not seen by the responder
FG6	RSTRH	Responder sent a SYN ACK followed by a RST, SYN not seen by the originator
	SHR	Responder sent a SYN ACK followed by a FIN, SYN not seen by the originator

4.2. Data preprocessing

NNR_w cannot process symbolic data or discrete data therefore, different techniques can be used to convert symbolic data into continuous data without affecting the performance. One can see in Table 3 that several symbolic features exist in the dataset. Some associated values of these features are listed in Table 5.

In Table 5, some features like *land, logged_in, is_host_login* and *is_guest_login* have values 0 or 1, therefore, we can handle these features in the same way as continuous features. Other features like *protocol_type, service* and *flag* have more than two different values. The *protocol_type* feature has 3 distinct values, *flag* feature has 11 distinct values, and *service* feature has 66 distinct values. Many researchers considered different approaches that are Indicator/Dummy Variables [32] and Conditional Probability [1] to handle symbolic features. We use the scheme proposed by Neter [32] in our experiment to handle such type of features. There are many symbolic values of *flag* and *service* features. The scheme proposed by Neter [32] will increase the dimensionality of dataset, therefore, based on domain knowledge, we also use clustering technique proposed by Hernndez-Pereira et al. [16] to group similar categories for different symbolic features. Hence, the *flag* and *service* features are further clustered to reduce the dimensionality before transforming these categories into indicator variable. The *flag* feature describes the status of the connection. The values of *flag* feature used in dataset are mentioned in Table 5. According to Hernndez-Pereira et al. [16], the categories of these features are further clustered into different groups as mentioned in Table 6.

The *service* feature describes the availability of services for a connection. The values of *service* attribute used in dataset are further clustered according to Hernndez-Pereira et al. [16] as mentioned in Table 7.

It is necessary to scale the data for NN. We performed necessary scaling to normalize the data. We used the *KDDTrain_20percent* dataset for training and $KDDTest^{+}$ and $KDDTest^{-21}$ for testing. During the experimental phase, two subsets are extracted from the training file, where *Tr* is a set of labeled examples used to train the classifier, and *U* is a set of unlabeled examples used to predict the labels. For the purpose of this simulation, the size of *Tr* is taken much smaller than that of *U* so that the efficiency of the proposed scheme can be tested properly. The division of training samples and unlabeled samples is based on the ratio of 10:90, where 10% is labeled data *Tr*, and remaining 90% is unlabeled data *U*. However, we used the testing datasets, $KDDTest^{+}$ and $KDDTest^{-21}$, as a whole to evaluate the performance of proposed technique.

Table 7
Clustering of *service* feature.

Service cluster	Service category	Description
SG1	telnet, ssh, ...	Services used to remotely access other machines
SG2	ftp, tftp, ...	Services used for file transfer
SG3	smtp, imap4, ...	Services used in mail transfer
SG4	http, ...	Services used in web applications
SG5	svstat, netstat, ...	Services used to get system parameters and statistics
SG6	host name, domain, ...	Services used in names servers
SG7	eco_i, tim_i, ecr_i, urp_i, ...	Services used in ICMP protocol
SG8	Remaining services	All other services

Table 8
List of experiments.

#. of experiments	Techniques for handling symbolic features
Experiment-1	Indicator/dummy variable
Experiment-2	Clustering technique + indicator variable

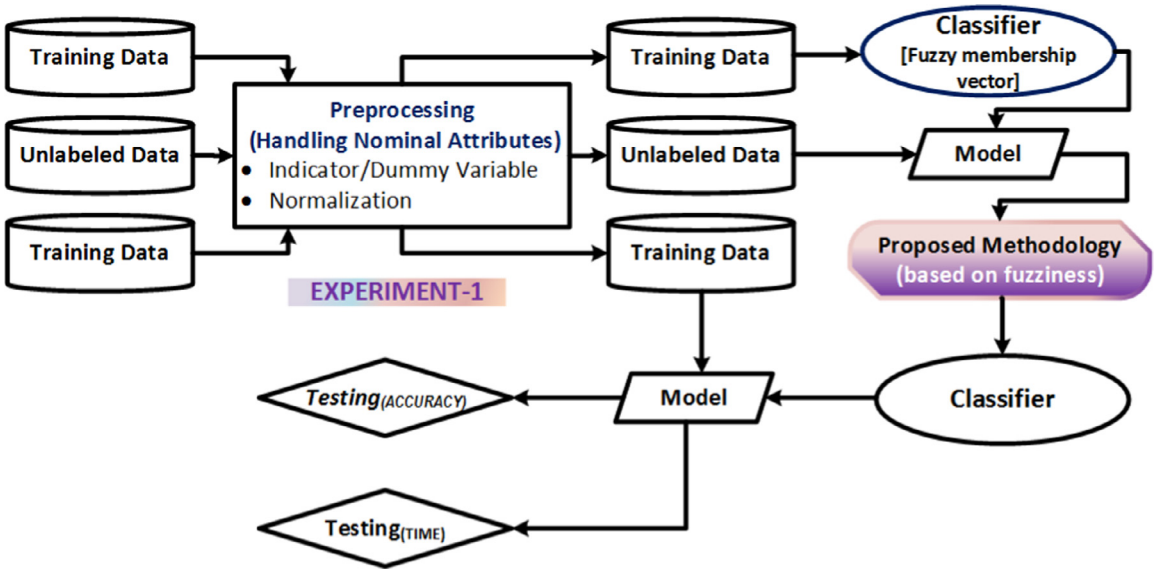


Fig. 3. Flow chart of Experiment-1 by using proposed methodology.

4.3. Experimental results

After required data preprocessing and necessary data scaling, we conduct the experiment in two modes as mentioned in Table 8 to evaluate the performance of the proposed methodology.

Experiment-1: In the first experiment, we convert all symbolic attributes into indicator variables as discussed earlier. The flow chart of experiment by using proposed strategy is depicted in Fig. 3.

We test our methodology and obtain the testing accuracy on both $KDDTest^{+}$ and $KDDTest^{-21}$ dataset. In first phase, we train the classifier NNR_w with original training set *Tr* and obtain the three groups of samples i.e., low, mid, and high, which are based on fuzzy quantity after utilizing the unlabeled set *U*. In second phase, we retrain the classifier with new training set *Tr*', where the samples belonging to low and high fuzziness groups are also incorporated with *Tr*. Testing accuracies on both datasets are depicted in Table 9. During the experiment, we also tested the impact of different initialization intervals on the overall performance of NNR_w. The initialization interval in this experiment was $[0, \theta]$, $1 \leq \theta \leq 10$. The input weights $\mathbf{w}_i$ and biases b_i at the hidden layer of NNR_w were the random variables that followed a *uniform* distribution over the interval $[0, \theta]$. Hence, a smaller interval, i.e., [0,1], leads to better accuracy as shown in Fig. 5(a).

For further verification of the results, the accuracies are computed by incorporating other categories with *Tr* and obtaining the following results as shown in Table 10.

Table 9
Testing accuracy on $KDDTest^{+}$ and $KDDTest^{-21}$.

Datasets	Proposed algorithm by incorporating $Tr + \text{FG}_{low} + \text{FG}_{high}$	
	$\text{Test}_{acc}(\%)$	Test_{time} (s)
$KDDTest^{+}$	82.41	0.012
$KDDTest^{-21}$	67.06	0.014

Table 10
Testing accuracy on $KDDTest^{+}$ and $KDDTest^{-21}$ dataset after incorporating other fuzziness groups into the training set.

Fuzziness categories	Test_{acc}	
	$KDDTest^{+}(\%)$	$KDDTest^{-21}(\%)$
FG_{low}	78.87	60.14
FG_{mid}	78.29	59.06
FG_{high}	78.81	59.99
$FG_{low} + FG_{mid}$	78.65	59.97
$FG_{mid} + FG_{high}$	78.79	59.75

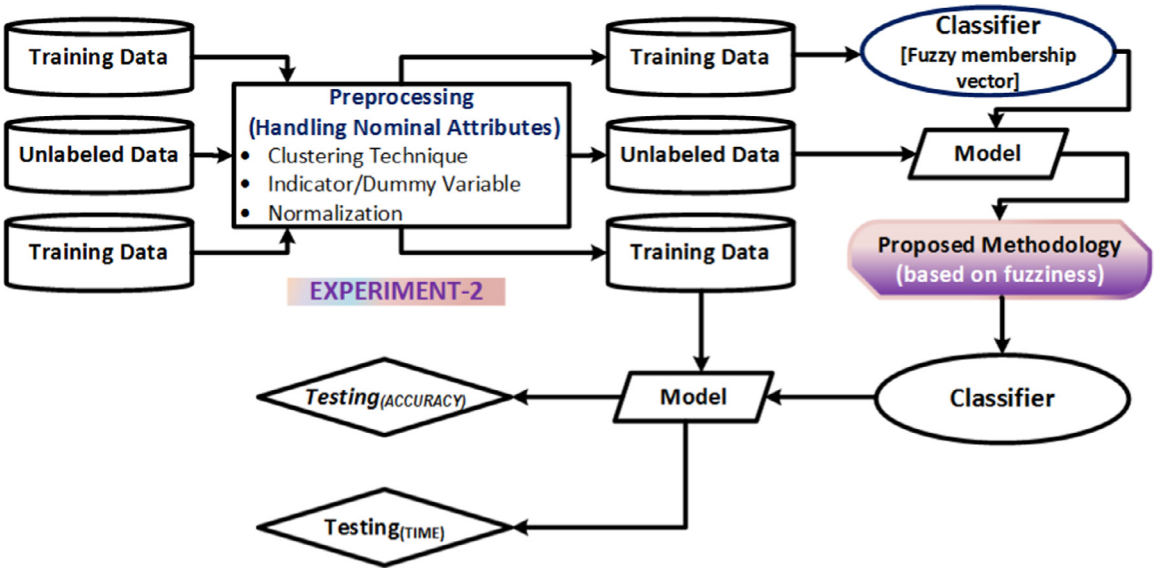


Fig. 4. Flow chart of Experiment-2 by using proposed methodology.

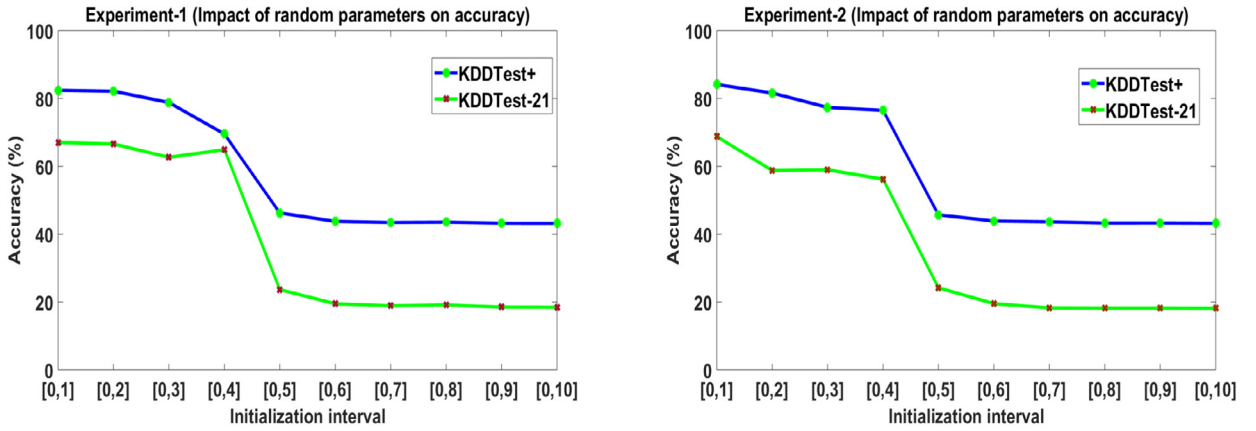
Table 11
Testing accuracy on $KDDTest^{+}$ and $KDDTest^{-21}$.

Datasets	Proposed Algorithm by incorporating $Tr + \text{FG}_{low} + \text{FG}_{high}$	
	Test_{acc} (%)	Test_{time} (s)
$KDDTest^{+}$	84.12	0.010
$KDDTest^{-21}$	68.82	0.312

Experiment-2: In the second experiment, we first apply the clustering technique (already discussed), which is proposed by Hernndez-Pereira et al. [16] to clusters the *flag* and the *service* features, after that these symbolic clusters are converted into dummy variables. The flow chart of Experiment-2 by using the proposed methodology is depicted in Fig. 4.

Again, we test the proposed methodology and obtain the following accuracies on both $KDDTest^{+}$ and $KDDTest^{-21}$ datasets as shown in Table 11. During this experiment, We also tested the impact of different initialization intervals on the overall performance of NNR_{w}. Therefore, a smaller interval, i.e., [0,1], also leads to better accuracy as depicted in Fig. 5(b).

For further verification of the attained results, the accuracies are computed by incorporating other categories with the original training set separately and obtaining the following results as shown in Table 12.



(a) Impact of different initialization intervals in Experiment-1 (b) Impact of different initialization intervals in Experiment-2

Fig. 5. Impact of different initialization intervals on the performance of proposed methodology using $KDDTest^{+}$ and $KDDTest^{-21}$ datasets.

Table 12
Testing accuracy on $KDDTest^{+}$ and $KDDTest^{-21}$ dataset after incorporating other fuzziness groups into the training set.

Fuzziness categories	$Test_{acc}$	
	$KDDTest^{+}$(%)	$KDDTest^{-21}$(%)
FG_{low}	81.01	64.50
FG_{mid}	78.63	56.99
FG_{high}	79.01	62.97
$FG_{low} + FG_{mid}$	81.04	62.94
$FG_{mid} + FG_{high}$	78.97	65.02

Table 13
Performance comparison between different classifiers and proposed algorithm.

Classifiers	$Test_{acc}$	
	$KDDTest^{+}$(%)	$KDDTest^{-21}$(%)
J48	81.05	63.97
Naive Bayes	76.56	55.77
NB tree	82.02	66.16
Random forests	80.67	63.25
Random tree	81.59	58.51
Multi-layer perceptron	77.41	57.34
SVM	69.52	42.29
Proposed algorithm-(Experiment-1)	82.41	67.06
Proposed algorithm-(Experiment-2)	**84.12**	**68.82**

4.4. Comparative analysis

According to the results shown in Table 9, the accuracies obtained by our proposed algorithm on $KDDTest^{+}$ and $KDDTest^{-21}$ dataset are maximum as compared to the accuracies obtained by Tavallaee et al. [48], where they used different classifiers to obtain the accuracy on both testing datasets. Testing accuracies obtained by Tavallaee et al. [48] and our proposed algorithm are depicted in Table 13.

We give some final remarks on both Experiment-1 and Experiment-2. In both experiments, training dataset includes $25,192$ records and both $KDDTest^{+}$ and $KDDTest^{-21}$ contain 22, 544 and 11, 850 records respectively, are used to test the proposed methodology. We extract two subsets from training dataset, one subset having proportion of 10% is used to train the classifier, and the second subset is used as unlabeled dataset having proportion of 90% of original training dataset. In the first experiment we used the indicator variable technique, where the dimensionality of dataset increased from 41 features to 122. While in the second experiment we used clustering methodology proposed by Hernndez-Pereira et al. [16] before applying the indicator or dummy variable technique, which limits the dimensionality to 55 features. Tables 9 and 11 indicate

that the accuracies obtained in both experiments are relatively high as compared to the accuracies on different classifiers as mentioned in Table 13.

The essential part of this algorithm is to add the samples belonging to both low and high fuzziness groups with their predicted labels into original training set, which enlarges the training set. It is also possible that the new training set may include some misclassified samples. To make clear which fuzziness group plays an important role to the improvement in classification accuracy, we conducted the experiment by adding each group separately. Hence, we conclude that the samples belong to low and high fuzziness groups together with original training set effectively increase the classification accuracy. Tables 10 and 12 list the classification accuracies by adding other fuzziness groups separately with the original training set. The main contribution of this research study is to improve the classification accuracy by finding the relationship between classifier's fuzziness and its misclassification on unlabeled samples. Hence, it is verified that the samples belonging to low and high fuzziness groups play an important role for improving the classifier performance and the samples with mid fuzziness exhibit higher risk of misclassification for ID dataset.

5. Conclusion

In this paper we have designed a new SSL algorithm for improving the classifier performance on ID datasets by investigating a divide-and-conquer strategy in which unlabeled samples with their predicted labels are categorized according to the magnitude of fuzziness. We used the neural network with random weights (NNR_w) as a base classifier because it is computationally efficient and has an excellent learning performance. The hidden-node parameters (i.e. weights and biases) in NNR_w are selected randomly and independently. The study mentioned in this paper is limited to achieve the better classification accuracy after finding the relationship between the fuzziness outputted by the classifier on a group of samples and their misclassification rate. It is experimentally observed that this methodology is an effective way to improve the classification accuracy when we train the NNR_w to get the fuzzy vector output and perform the sample categorization on unlabeled samples according to their fuzziness quantity. The classifier is retrained after incorporating the unlabeled samples with their predicted labels (that belong to the low and high fuzziness groups) into the original training set. This study also verifies that samples belonging to the mid fuzziness group have a higher risk of misclassification for IDSs. In this paper, we reported two-class problem, i.e., normal and anomaly. Our future research will be directed towards applying this strategy to improve the effectiveness of IDSs for detecting multiple types of attacks.

Acknowledgments

The authors would like to extend their sincere appreciation to the Deanship of Scientific Research at King Saud University for its funding of this research through the Research Group Project no. RG-1435-048. This research is also supported by China Postdoctoral Science Foundation (2015M572361), Basic Research Project of Knowledge Innovation Program in Shenzhen (JCYJ20150324140036825), and National Natural Science Foundations of China (61503252 and 71371063).

References

[1] D. Aha, D. Kibler, M. Albert, Instance-based learning algorithms, Mach. Learn. 6 (1) (1991) 37–66.
[2] M. Alhamdoosh, D.H. Wang, Fast decorrelated neural network ensembles with random weights, Inf. Sci. 264 (2014) 104–117.
[3] S. Baluja, Using labeled and unlabeled data for probabilistic modeling of face orientation, Int. J. Pattern Recognit. Artif. Intell. 14 (08) (2000) 1097–1107.
[4] A. Blum, S. Chawla, Learning from labeled and unlabeled data using graph mincuts, in: Proceedings of the Eighteenth International Conference on Machine Learning, 2001, pp. 19–26.
[5] A. Blum, T. Mitchell, Combining labeled and unlabeled data with co-training, in: Proceedings of the Eleventh Annual Conference on Computational Learning Theory - COLT' 98, 1998, pp. 92–100.
[6] S. Bosworth, M. Kabay, Computer Security Handbook, John Wiley & Sons, New York, 2002.
[7] F.L. Cao, H.L. Ye, D.H. Wang, A probabilistic learning algorithm for robust modeling using neural networks with random weights, Inf. Sci. 313 (2015) 62–78.
[8] C. Chen, Y. Gong, Y. Tian, Semi-supervised learning methods for network intrusion detection, in: Proceedings of the 2008 IEEE International Conference on Systems, Man and Cybernetics, 2008, pp. 2603–2608.
[9] W. Chen, Y. Shao, N. Hong, Laplacian smooth twin support vector machine for semi-supervised classification, Int. J. Mach. Learn. Cybern. 5 (3) (2013) 459–468.
[10] J. Chen, Y. Wang, X. Wang, On-demand security architecture for cloud computing, Computer 45 (7) (2012) 73–78.
[11] A. De Luca, S. Termini, A definition of a nonprobabilistic entropy in the setting of fuzzy sets theory, Inf. Control 20 (4) (1972) 301–312.
[12] D. Denning, An intrusion-detection model, IEEE Trans. Softw. Eng. 13 (2) (1987) 222–232.
[13] A. Fujino, N. Ueda, K. Saito, A hybrid generative/discriminative classifier design for semi-supervised learning, Trans. Jpn. Soc. Artif. Intell. 21 (2006) 301–309.
[14] Q. Gao, Y. Huang, X. Gao, W. Shen, H. Zhang, A novel semi-supervised learning for face recognition, Neurocomputing 152 (2015) 69–76.
[15] Y. He, X.Z. Wang, J.Z.X. Huang, Fuzzy nonlinear regression analysis using a random weight network, Inf. Sci. (2016) In press, doi:10.1016/j.ins.2016.01.037.
[16] E. Hernndez-Pereira, J. Surez-Romero, O. Fontenla-Romero, A. Alonso-Betanzos, Conversion methods for symbolic features: a comparison applied to an intrusion detection problem, Expert Syst. Appl. 36 (7) (2009) 10612–10617.
[17] B. Igelnik, Y.-H. Pao, Stochastic choice of basis functions in adaptive function approximation and the functional-link net, IEEE Trans. Neural Netw. 6 (6) (1995) 1320–1329.
[18] KDDCup 1999 Data, 2015. [Online]. Available: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html. [Accessed: 23-06-15].
[19] J. Keller, M. Gray, J. Givens, A fuzzy k-nearest neighbor algorithm, IEEE Trans. Syst. Man Cybern. 15 (4) (1985) 580–585.
[20] C. Kruegel, F. Valeur, G. Vigna, Intrusion Detection and Correlation, Springer, New York, 2005.

[21] T. Lane, A decision-theoretic, semi-supervised model for intrusion detection, part of the series advanced information and knowledge processing, Machine Learning and Data Mining for Computer Security: Methods and Applications, Springer, London, 2006, pp. 157–177, doi:10.1007/1-84628-253-5_10.
[22] P. Laskov, P. Dssel, C. Schfer, K. Rieck, Learning intrusion detection: Supervised or unsupervised? in: Proceedings of the Conference on Image Analysis and Processing ICIAP 2005, 2005, pp. 50–57.
[23] C. Lee, D.A. Landgrebe, Decision boundary feature extraction for neural networks, IEEE Trans. Neural Netw. 8 (1) (1997) 75–83.
[24] Y. Liao, V. Vemuri, Use of k-nearest neighbor classifier for intrusion detection, Comput. Secur. 21 (5) (2002) 439–448.
[25] M. Luo, L. Wang, H. Zhang, J. Chen, A Research on intrusion detectionbased on unsupervised clustering and support vector machine, in: Proceedings of Information and Communications Security: 5th International Conference, ICICS 2003, Huhehaote, China, October 10–13, 2003 (series Lecture Notes in Computer Science), vol. 2836, Springer, Berlin Heidelberg, 2003, pp. 325–336, doi:10.1007/978-3-540-39927-8_30.
[26] A. Mahmood, T. Li, Y. Yang, H. Wang, M. Afzal, Semi-supervised evolutionary ensembles for web video categorization, Knowl. Based Syst. 76 (2015) 53–66.
[27] U. Maulik, D. Chakraborty, A novel semisupervised SVM for pixel classification of remote sensing imagery, Int. J. Mach. Learn. Cybern. 3 (3) (2011) 247–258.
[28] M. Meng, J. Wei, J. Wang, Q. Ma, X. Wang, Adaptive semi-supervised dimensionality reduction based on pairwise constraints weighting and graph optimizing, Int. J. Mach. Learn. Cybern. (2015) in press, doi:10.1007/s13042-015-0380-3 (accessed 19.08.15).
[29] S. Mukkamala, G. Janoski, A. Sung, Intrusion detection using neural networks and support vector machines, in: Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN02 (Cat. No.02CH37290), vol. 2, 2002, p. 17021707.
[30] S. Mukkamala, A. Sung, Detecting denial of service attacks using support vector machines, in: Proceedings of the Twelfth IEEE International Conference on Fuzzy Systems, 2003.
[31] S. Mukkamala, A. Sung, A. Abraham, Intrusion detection using an ensemble of intelligent paradigms, J. Netw. Comput. Appl. 28 (2005) 167. Science Direct
[32] J. Neter, Applied Linear Statistical Models, WCB/MacGraw-Hill, Boston, 1996.
[33] K. Nigam, R. Ghani, Analyzing the effectiveness and applicability of co-training, in: Proceeding of the Ninth International Conference on Information and Knowledge (CIKM-2000), 2000.
[34] K. Nigam, A. Mccallum, S. Thrun, T. Mitchell, Text classification from labeled and unlabeled documents using EM, Mach. Learn. 39 (23) (2000) 103–134.
[35] NSL-KDD Data Set, [Online]. Available: http://nsl.cs.unb.ca/NSL-KDD/. [Accessed: 23-06-15].
[36] B. Pan, J. Lai, L. Shen, Ideal regularization for learning kernels from labels, Neural Netw. 56 (2014) 22–34.
[37] F. Pan, J. Wang, X. Lin, Local margin based semi-supervised discriminant embedding for visual recognition, Neurocomputing 74 (5) (2011) 812–819.
[38] Y. Pao, G. Park, D. Sobajic, Learning and generalization characteristics of the random vector functional-link net, Neurocomputing 6 (2) (1994) 163–180.
[39] Z. Qi, Y. Tian, L. Niu, B. Wang, Semi-supervised classification with privileged information, Int. J. Mach. Learn. Cybern. 6 (4) (2015) 667–676.
[40] M. Qiu, L. Zhang, Z. Ming, Z. Chen, X. Qin, L. Yang, Security-aware optimization for ubiquitous computing systems with SEAT graph approach, J. Comput. Syst. Sci. vol. 79 (5) (2013) 518–529.
[41] E. Riloff, J. Wiebe, T. Wilson, Learning subjective nouns using extraction pattern bootstrapping, in: Proceedings of the Seventh conference on Natural language learning at HLT-NAACL 2003, vol. 4, 2003, pp. 25–32.
[42] C. Rosenberg, M. Hebert, H. Schneiderman, Semi-supervised self-training of object detection models, in: Proceedings of the 2005 Seventh IEEE Workshops on Applications of Computer Vision (WACV/MOTION'05), 1, 2005, pp. 29–36.
[43] D. Sanchez, E. Trillas, Measures of fuzziness under different uses of fuzzy sets, Commun. Comput. Inf. Sci. 298 (2012) 25–34.
[44] S. Scardapane, D.H. Wang, M. Panella, A. Uncini, Distributed learning for random vector functional-link networks, Inf. Sci. 301 (2015) 271–284.
[45] K. Scarfone, P. Mell, 2007, SP 800–94. Guide to Intrusion Detection and Prevention Systems (IDPS). National Institute of Standards & Technology, Gaithersburg, MD, United States.
[46] W. Schmidt, M. Kraaijveld, R. Duin, Feedforward neural networks with random weights, in: Proceedings of the Eleventh IAPR International Conference on Pattern Recognition, Conference B: Pattern recognition Methodology and Systems, 1992, pp. 1–4.
[47] C. Shang, S. Feng, Z. Zhao, J. Fan, Efficiently detecting overlapping communities using seeding and semi-supervised learning, Int. J. Mach. Learn. Cybern. (2015) in press, doi:10.1007/s13042-015-0338-5 (accessed 19.08.15).
[48] M. Tavallaee, E. Bagheri, W. Lu, A. Ghorbani, A detailed analysis of the KDD CUP 99 data set, in: Proceedings of the 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, 2009.
[49] H.T. Braake, G.V. Straten, Random activation weight neural net (RAWN) for east non-iterative training, Eng. Appl. Artif. Intell. 8 (1) (1995) 71–80.
[50] V. Vapnik, Statistical Learning Theory, Wiley, New York, 1998.
[51] X. Wang, R.A.R. Ashfaq, A. Fu, Fuzziness based sample categorization for classifier performance improvement, J. Intell. Fuzzy Syst. 29 (3) (2015) 1185–1196.
[52] H. Wang, R. Nie, X. Liu, T. Li, Constraint projections for semi-supervised affinity propagation, Knowl. Based Syst. 36 (2012) 315–321.
[53] X. Wang, H.J. Xing, Y. Li, Q. Hua, C.R. Dong, W. Pedrycz, A study on relationship between generalization abilities and fuzziness of base classifiers in ensemble learning, IEEE Trans. Fuzzy Syst. 23 (5) (2015) 1638–1654.
[54] J. Xie, K. Hone, W. Xie, X. Gao, Y. Shi, X. Liu, Extending twin support vector machine classifier for multi-category classification problems, Intell. Data Anal. 17 (4) (2013) 649–664.
[55] Y. Yam, T. Chow, C. Leung, A new method in determining initial weights of feedforward neural networks for training enhancement, Neurocomputing 16 (1) (1997) 23–32.
[56] Q. Yan, F. Yu, Distributed denial of service attacks in software-defined networking with cloud computing, IEEE Commun. Mag. 53 (4) (2015) 52–59.
[57] D. Yarowsky, Unsupervised word sense disambiguation rivaling supervised methods, in: Proceedings of the Thirty Third Annual Meeting on Association for Computational Linguistics, 1995, pp. 189–196.
[58] Z. You, J.Z. Yu, L. Zhu, S. Li, Z.K. Wen, A mapreduce based parallel SVM for large-scale predicting proteinprotein interactions, Neurocomputing 145 (2014) 37–43.
[59] Y. Yuan, M.J. Shaw, Induction of fuzzy decision trees, Fuzzy Sets Syst. 69 (1995) 125–139.
[60] L. Zadeh, Probability measures of fuzzy events, J. Math. Anal. Appl. 23 (2) (1968) 421–427.
[61] J.W. Zhao, Z.H. Wang, F.L. Cao, D.H. Wang, A local learning algorithm for random weights networks, Knowl. Based Syst. 74 (2015) 159–166.
[62] M. Zhao, T. Chow, Z. Wu, Z. Zhang, B. Li, Learning from normalized local and global discriminative information for semi-supervised regression and dimensionality reduction, Inf. Sci. 324 (2015) 286–309.
[63] D. Zhou, J. Huang, B. Schlkopf, Learning from labeled and unlabeled data on a directed graph, in: Proceedings of the Twenty Second International Conference on Machine Learning - ICML '05, 2005, pp. 1036–1043.
[64] X. Zhu, Semi-Supervised Learning Literature Survey, Computer Sciences Technical Report 1530, University of WisconsinMadison, 2005.
[65] X. Zhu, A. Goldberg, Introduction to semi-supervised learning, Synth. Lect. Artif. Intell. Mach. Learn. 3 (1) (2009) 1–130.