Knowledge-Based Systems 78 (2015) 13–21

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

CANN: An intrusion detection system based on combining cluster centers and nearest neighbors

Wei-Chao Lin [a], Shih-Wen Ke [b], Chih-Fong Tsai [c,*]

[a] Department of Computer Science and Information Engineering, Hwa Hsia University of Technology, Taiwan
[b] Department of Information and Computer Engineering, Chung Yuan Christian University, Taiwan
[c] Department of Information Management, National Central University, Taiwan

ARTICLE INFO

Article history:
Received 4 April 2014
Received in revised form 17 December 2014
Accepted 17 January 2015
Available online 23 January 2015

Keywords:
Intrusion detection
Anomaly detection
Feature representation
Cluster center
Nearest neighbor

ABSTRACT

The aim of an intrusion detection systems (IDS) is to detect various types of malicious network traffic and computer usage, which cannot be detected by a conventional firewall. Many IDS have been developed based on machine learning techniques. Specifically, advanced detection approaches created by combining or integrating multiple learning techniques have shown better detection performance than general single learning techniques. The feature representation method is an important pattern classifier that facilitates correct classifications, however, there have been very few related studies focusing how to extract more representative features for normal connections and effective detection of attacks. This paper proposes a novel feature representation approach, namely the cluster center and nearest neighbor (CANN) approach. In this approach, two distances are measured and summed, the first one based on the distance between each data sample and its cluster center, and the second distance is between the data and its nearest neighbor in the same cluster. Then, this new and one-dimensional distance based feature is used to represent each data sample for intrusion detection by a *k*-Nearest Neighbor (*k*-NN) classifier. The experimental results based on the KDD-Cup 99 dataset show that the CANN classifier not only performs better than or similar to *k*-NN and support vector machines trained and tested by the original feature representation in terms of classification accuracy, detection rates, and false alarms. I also provides high computational efficiency for the time of classifier training and testing (i.e., detection).

1. Introduction

Advancements in computing and network technology have made the activity of accessing the Internet an important part of our daily life. In addition, the amount of people connected to the Internet is increasing rapidly. However, the high popularity of world-wide connections has led to security problems.

Traditionally, some techniques, such as user authentication, data encryption, and firewalls, are used to protect computer security. Intrusion detection systems (IDS), which use specific analytical technique(s) to detect attacks, identify their sources, and alert network administrators, have recently been developed to monitor attempts to break security [3]. In general, IDS are developed for signature and/or anomaly detection. For signature detection, packets or audit logs are scanned to look for sequences of commands or events which are previously determined as indicative of an attack. On the other hand, for anomaly detection, IDS use behavior patterns which could indicate malicious activities and analyzes past activities to recognize whether the observed behaviors are normal. As early IDS largely used signature detection to detect all the attacks captured in their signature databases, they suffer from high false alarm rates. Recent innovative approaches including behavior-based modeling have been proposed to detect anomalies include data mining, statistical analysis, and artificial intelligence techniques [21,28].

Much related work in the literature focuses on the task of anomaly detection based on various data mining and machine learning techniques. There have been many recent studies, which focus on combining or integrating different techniques in order to improve detection performance, such as accuracy, detection, and/or false alarm rates (see Table 1 in Section 2.4).

However, there are two limitations to existing studies. First, although more advanced and sophisticated detection approaches and/or systems have been developed, very few have focused on feature representation for normal connections and attacks, which is an important issue in enhancing detection performance. There is a huge amount of related studies using either the KDD-Cup 99

* Corresponding author. Tel.: +886 3 422 7151; fax: +886 3 425 4604.
E-mail address: cftsai@mgt.ncu.edu.tw (C.-F. Tsai).

http://dx.doi.org/10.1016/j.knosys.2015.01.009
0950-7051/© 2015 Elsevier B.V. All rights reserved.

or DARPA 1999 dataset for experiments, however there is no an exact answer to the question about which features of these datasets are more representative. Second, the time taken for training the systems and for the detection task to further validate their systems are not considered in many evaluation methods. Recent systems that combine or integrate multiple techniques require much greater computational effort. As a result, this can degrade the efficiency of 'on-line' detection.

Therefore, in this study, we propose a novel feature representation method for effective and efficient intrusion detection that is based on combining cluster centers and nearest neighbors, which we call CANN. Specifically, given a dataset, the *k*-means clustering algorithm is used to extract cluster centers of each pre-defined category. Then, the nearest neighbor of each data sample in the same cluster is identified. Next, the sum of the distance between a specific data sample and the cluster centers and the distance between this data sample and its nearest neighbor is calculated. This results in a new distance based feature that represents the data in the given dataset. Consequently, a new dataset containing only one dimension (i.e., distance = based feature representation) is used for *k*-Nearest Neighbor classification, which allows for effective and efficient intrusion detection.

The idea behind CANN is that the cluster centers or centroids for a given dataset offer discrimination capabilities for recognition both similar and dissimilar classes [9,10,35]. Therefore, the distances between a data sample and these identified cluster centers are likely to provide some further information for recognition. Similarly, the distance between a specific data sample and its nearest data sample in the same class also has some discriminatory power.

The rest of this paper is organized as follows. Section 2 reviews related literature including offering brief descriptions of supervised and unsupervised machine learning techniques. The techniques used in this paper are also described. Moreover, the techniques used, datasets and evaluation strategies considered in related work are compared. The proposed approach for intrusion detection is introduced in Section 3. Section 4 presents the experimental setup and results. Finally, some conclusions are provided in Section 5.

2. Literature review

2.1. Machine learning

Machine learning requires a system capable of the autonomous acquisition and integration of knowledge. This capacity includes learning from experience, analytical observation, and so on, the result being a system that can continuously self-improve and thereby offers increased efficiency and effectiveness. The main goal of the study of machine learning is to design and develop algorithms and techniques that allow computers to learn. In general, there are two types of machine learning techniques, supervised and unsupervised [22,24] which are described in greater detail below.

2.2. Supervised learning

Supervised learning or classification is one common type of machine learning technique for creating a function from a given set of training data. The training data are composed of pairs of input objects and their corresponding outputs. The output of the function can be a continuous value, and can predict a class label of the input object. Particularly, the learning task is to compute a classifier that approximates the mapping between the input–output training examples, which can correctly label the training data with some level of accuracy.

The *k*-Nearest Neighbor (*k*-NN) algorithm is a conventional non-parametric classifier used in machine learning [22]. The purpose of this algorithm is to assign an unlabelled data sample to the class of its *k* nearest neighbors (where *k* is an integer). Fig. 1 shows an example for a *k*-NN classifier where $k = 5$. Consider the 5 nearest neighbors around X for the unlabelled data to be classified. There are three 'similar' patterns from class C_2 and two from class C_1. Taking a majority vote enables the assignment of X to the C_2 class.

According to Jain et al. [13], *k*-NN can be conveniently used as a benchmark for all the other classifiers since it is likely to provide a reasonable classification performance in most applications. Other well-known supervised learning techniques used in intrusion detection include support vector machines, artificial neural networks, decision trees, and so on [3,33,37].

2.3. Unsupervised learning

Unsupervised learning or clustering is a method of machine learning where a model is fit to observations. It differs from supervised learning in the absence of prior output. In unsupervised learning, a data set of input objects is gathered first. The input objects are typically treated as a set of random variables. A joint density model is then built for the data set [22].

The machine simply receives the inputs $x_1, x_2, \ldots, x_n$, obtaining neither supervised target outputs, nor rewards from its environment. It may seem somewhat mysterious to imagine what the machine could possibly learn given that it does not get any feedback from its environment. However, it is possible to develop a formal framework for unsupervised learning based on the notion that the machine's goal is to build representations of the input that can be used for decision making, predicting future inputs, efficiently communicating the inputs to another machine, etc.

The *k-means* clustering algorithm is the simplest and most commonly used unsupervised machine learning technique [14] being a simple and easy way to classify a given dataset through a certain number of clusters. The goal of the *k-means* algorithm is to find *k* points of a dataset, which can best represent this dataset in a certain number of groups. The point, *k*, is the cluster center or centroid of each cluster.

In particular, *k-means* is used to cluster or group N data points into K disjoint subsets S_j containing N_j data points so as to minimize the sum-of-squares criterion,

$$J = \sum_{j=1}^{k} \sum_{n \in S_j} |x_n - \mu_j|^2 \tag{1}$$

where x_n is a feature vector representing the *n*-th data point and μ_j is the geometric centroid of the data points in S_j.

In the literature, it can be seen that some clustering techniques are combined with specific supervised learning techniques for intrusion detection. For example, Khan et al. [16] combined self-organizing maps (SOM) and support vector machines, Xiang et al. [37] combined Bayesian clustering and decision trees, and *C*-means clustering and artificial neural networks are combined in Zhang et al. [38].

2.4. Comparison of related work

A number of related intrusion detection systems are compared and the results shown in Table 1. In particular, we compare the machine learning techniques used for developing the detection systems, datasets used for experiments, evaluation methods considered, baseline classifiers for comparisons, etc. in relevant studies. For a detailed review, readers can refer to García-Teodoro et al. [7] and Tsai et al. [34].

Table 1
Comparisons of related work.

Work	Technique	Dataset	Problem domain	Evaluation method	Baseline	Feature selection
Eesa et al. [6]	DT[a]	KDD-Cup 99	Anomaly detection	DR[b], FP[c], accuracy, ROC curve[d]	N/A	Yes
de la Hoz et al. [4]	GHSOM[e]	KDD-Cup 99 + DARPA 1999	Anomaly detection	DR, ROC curve	NB[f], RF[g], DT, AdaBoost	Yes
Feng et al. [5]	SVM[h]+ant colony network	KDD-Cup 99	Anomaly detection	DR, FP, FN[i]	SVM	No
Kim et al. [17]	DT + SVM	A modified version of KDD-Cup 99	Anomaly and misuse detection	DR, ROC curve	DT, SVM	No
Baig et al. [2]	GMDH[j]	KDD-Cup 99	Anomaly detection	Accuracy, Recall, Precision, FP, FN, ROC	NB, ANN[k]	Yes
Shin et al. [29]	Markov chain + probabilistic modeling of network events	DARPA 2000	Anomaly detection	DR, FP, ROC	Markov chain model	Yes
Lin et al. [19]	SA[l]+DT, SVM	KDD-Cup 99	Anomaly detection	DR	DT, SVM	Yes
Sangkatsanee et al. [27]	DT, ANN, Ripper rule	Reliability Lab Data 2009/KDD-Cup 99	Anomaly detection	DR	N/A	Yes
Wang et al. [36]	Fuzzy clustering + ANN	KDD-Cup 99	Anomaly detection	Precision, Recall, F-measure	DT/NB[m], ANN	No
Tajbakhsh et al. [32]	FL[n]+AR[o]	KDD-Cup 99	Anomaly detection	DR, FP	SVM, *k*-NN	No
Tong et al. [33]	RBF, Elman neural networks	DARPA 1999	Anomaly detection	DR, FP	ANN, SOM[p]	No
Giacinto et al. *[8]*	*k*-means, SVM ensembles	KDD-Cup 99	Anomaly detection	DR	N/A	No
Hu et al. [12]	AdaBoost DT	KDD-Cup 99	Anomaly detection	DR, FA[q], Run time	N/A	No
Xiang et al. [37]	Bayes clustering + DT	KDD-Cup 99	Anomaly detection	DR, Run time	DT	Yes
Abadeh et al. [1]	GA[r]+FL	DARPA 1998	Anomaly detection	DR, FA	FL	No
Chen et al. [3]	GA + ANN	DARPA 1998	Anomaly detection	FP, FN	ANN, 2 layer ANN	Yes
Hansen et al. [11]	GA	KDD-Cup 99	Anomaly detection	DR	N/A	No
Khan et al. [16]	SOM + SVM	DARPA 1998	Anomaly detection	FP, FN, accuracy	SVM	No
Li and Guo [18]	TCM *k*-NN	KDD-Cup 99	Anomaly detection	TP[s], FP	SVM, ANN, *k*-NN	No
Liu et al. [20]	SOM + ANN	DARPA 1998	Anomaly and misuse detection	DR, FA, FP	SVM, DT, SOM	Yes
Ozyer et al. [25]	GA + FL	KDD-Cup 99'	Anomaly and misuse detection	DR	GA	No
Peddabachigari et al. [26]	DT + SVM	KDD-Cup 99'	Anomaly and misuse detection	Accuracy	SVM, DT	No
Shon and Moon [30]	GA + SVM	DARPA 1999	Anomaly detection	DR, FP, FN	SVM	Yes
Shon et al. [31]	GA + ANN/*k*-NN/SVM	DARPA 1998	Anomaly detection	DR, FP, FN	ANN, *k*-NN, SVM	Yes
Sarasamma et al. [28]	Hierarchical SOM	KDD-Cup 99	Anomaly detection	DR. FP	SOM	Yes
Zhang et al. [38]	C-means clustering + ANN	KDD-Cup 99'	Anomaly and misuse detection	DR, FP	ANN	No

[a] DT: Decision Tree References.
[b] DR: detection rate.
[c] FP: false positive.
[d] ROC curve: Receiver Operating Characteristic curve.
[e] GHSOM: Growing Hierarchical SOM.
[f] NB: Naïve Bayes.
[g] RF: Random Forest.
[h] SVM: support vector machine.
[i] FN: false negative.
[j] GMDH: Group Method for Data Handling.
[k] ANN: Artificial Neural Networks.
[l] SA: Simulated Annealing.
[m] NB: Naïve Bayes.
[n] FL: Fuzzy Logic.
[o] AR: Association Rules.
[p] SOM: self-organizing maps.
[q] FA: false alarm.
[r] GA: Genetic Algorithm.
[s] TP: true positive.

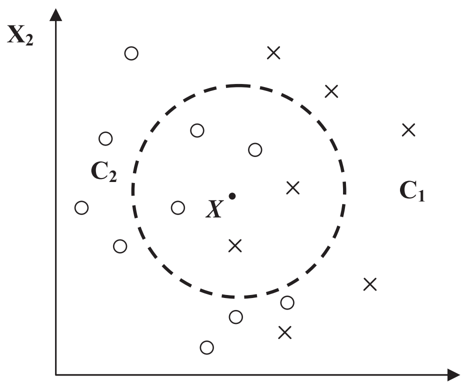

Fig. 1. A k-Nearest Neighbor rule for $k = 5$.

As we can see from Table 1, DARPA1998 and KDD-Cup99 are the most commonly used datasets for simulating intrusion detection while SVM and k-NN classifiers are popular baseline techniques used in related work proposing novel techniques. In addition, it is the trend that related studies consider combining or integrating two different techniques in order to improve the intrusion detection performance.

Related studies where the main focus is on developing more advanced techniques to improve the intrusion detection performance, tend to rely only on some feature selection methods, such as principal component analysis (PCA), to filter out unrepresentative features in the DARPA1998 and/or KDD-Cup99 dataset (e.g., [20,25,39]. However, according to Table 1, very few studies combine feature selection and classification techniques to examine the effect of performing feature selection on the intrusion detection performance.

However, each study uses different features over the same datasets. Therefore, currently it is not known what features are more representative in the two datasets.

The detection rate (DR), false positive (FP), false negative (FN), true positive (TP), false alarm (FA), and the accuracy rate are most often examined for evaluation measurements. Only Hu et al. [12] and Xiang et al. [37] considered the run time during intrusion detection as another performance indicator. It is known that for intrusion detection systems the computational effort, i.e., run time, for online detection should be as short as possible. Although this is a very critical issue, very few have considered the detection time of their systems.

The above discussion leads us to propose the method described below, which can not only extract representative features for improving detection performances, but also provide computational efficiency.

3. CANN: the proposed approach

3.1. The CANN process

The proposed approach is based on two distances which are used to determine the new features, between a specific data point and its cluster center and nearest neighbor respectively. CANN is comprised of three steps as shown in Fig. 2.

Given a training dataset T, the first step is to use a clustering technique to extract cluster centers. The number of clusters is based on the number of classes to be classified. Since intrusion detection is one classification problem, the chosen dataset has already defined the number of classes to be classified. Therefore, for example, if the given dataset is a three-class problem, then the number of clusters is defined as three. Besides extracting

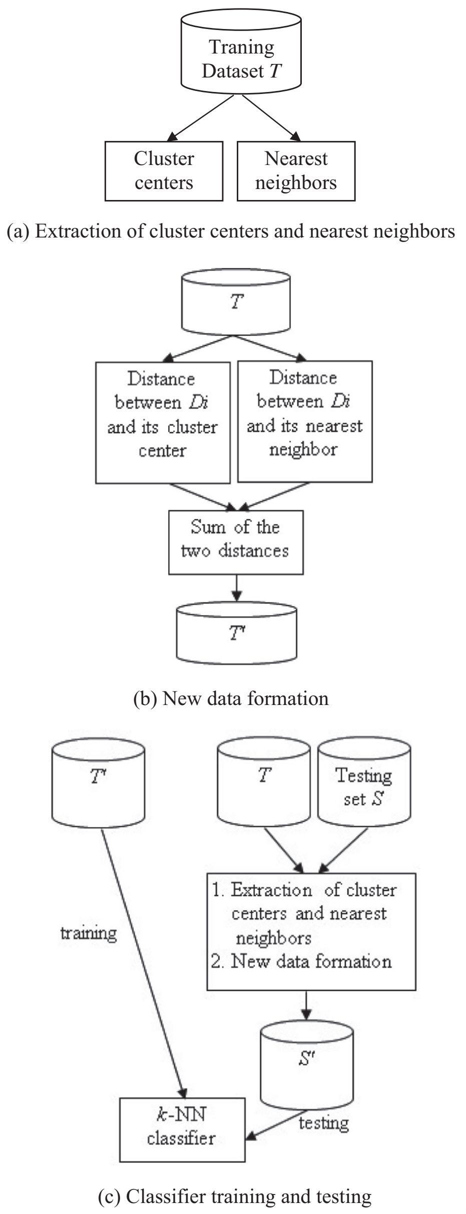
Traning Dataset T
Cluster centers
Nearest neighbors
(a) Extraction of cluster centers and nearest neighbors
T
Distance between Di and its cluster center
Distance between Di and its nearest neighbor
Sum of the two distances
T'
(b) New data formation
T'
T
Testing set S
1. Extraction of cluster centers and nearest neighbors
2. New data formation
training
S'
k-NN classifier
testing
(c) Classifier training and testing

Fig. 2. The CANN process.

cluster centers, each data point of the given dataset and its nearest neighbor in the same cluster is identified. This can be done by calculating the distances between one specific data point (Di) and all of the other data in the same cluster. Then, the shortest distance between two data examples representing Di and its nearest neighbor can be found.

The second step is to measure and sum the distance ($dis1$) between all data of the given dataset and the cluster centers and the distance ($dis2$) between each data point and its nearest neighbor in the same cluster. This leads to a new distance based feature value to represent each data point of the given dataset, which is T'. That is, the original features (i.e., the number of dimensions is usually larger than one) are replaced by one new dimension feature.

To test the new unknown data for intrusion detection, the testing set S is combined with the original training set T. Then, the processes of extracting cluster centers and nearest neighbors (Fig. 2(a)) and new data formation (Fig. 2(b)) are executed. During these processes, only the data samples in S are considered. As a result, the new distance based feature dataset S' is obtained.

Therefore, T' and S' are used to train and test the k-NN classifier for intrusion detection.

3.2. Extraction of cluster centers and nearest neighbors

To extract cluster centers, a clustering technique can be applied in this stage. In this study, the k-means clustering algorithm is used. Fig. 3 shows an example where the chosen dataset consisting of 12 data samples (N_1 to N_{12}) is a five-class classification problem. Thus, the number of clusters is defined as five (i.e., $k = 5$) for the k-means clustering algorithm. As a result, there are five clusters, which each cluster containing a cluster center (i.e., C_1, C_2, C_3, C_4, and C_5).

On the other hand, to identify the nearest neighbor of a data point, D_i for example, the k-NN approach is used where the distance between D_i and each of the other data points in the same cluster can be obtained. That is, the nearest neighbor of D_i is based on the shortest distance identified by k-NN. Therefore, for Fig. 3, N_1 is the nearest neighbor of D_i.

3.3. New data formation

After the cluster center and nearest neighbor for every data point of the chosen dataset are extracted and identified, two types of distances are calculated and then summed. The first type is based on the distance from each data point to the cluster centers. That is, if there are three cluster centers, then there are three distances between a data point to the three cluster centers, respectively. The second type is based on the distance from each data point to its nearest neighbor. Fig. 4 shows an example of five clusters, in which the two types of distances for the data point D_i are obtained by

$$D_i = \overline{D_iC_1} + \overline{D_iC_2} + \overline{D_iC_3} + \overline{D_iC_4} + \overline{D_iC_5} \quad (2)$$

Specifically, the distance between two data points is based on the Euclidean distance. For example, given that data A and B contain n-dimensional features, their Euclidean distance is based on

$$dis\ AB = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \ldots + (a_n - b_n)^2} = \sqrt{\sum_{i=1}^{n}(a_i - b_i)^2} \quad (3)$$

Following the example shown in Fig. 3, the CANN approach transforms the original n-dimensional features to the one-dimensional distance feature by

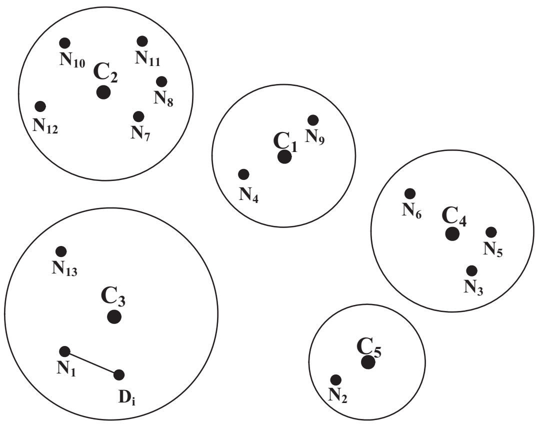

Fig. 3. An example of extracting cluster centers and nearest neighbors.

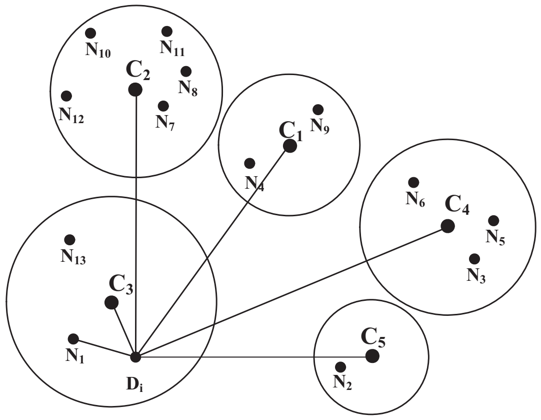

Fig. 4. An example of new data formation.

$$Dis\,(D_i) = \sum_{j=1}^{5} dis(D_i, C_j) + \sum_{k=1} dis(D_i, N_k) \quad (4)$$

,where D_i is the i-th data point of a given dataset, C_j is the j-th cluster center identified by k-means, and N_k is the k-th data point which is the nearest neighbor of D_i. These distance values are summed to represent the feature of D_i. As a result, each data point of the given dataset has its one dimensional distance feature. Finally, for the dataset containing m data samples, in which each sample originally has n dimensions, the CANN approach produces a new dataset containing m data samples, in which each sample has one dimension, i.e., the distance feature.

For classifier construction, the final step of CANN, the new dataset is divided into the training and testing datasets to train and test a specific classifier. In this study, we consider the k-NN classifier since it is easy to implement and widely used as a baseline classifier in many applications.

It should be noted that the CANN process can be applied to any dataset with and without feature selection. For instance, the collected dataset contains D dimensional features. When dimensionality reduction is required, a chosen feature selection algorithm is used to select some representative features from the training set resulting in D' features where $D' < D$. Next, the reduced training dataset is used for the CANN process, which can be regarded as T shown in Fig. 2(a) and (b). For the testing set, its features are the same as the ones identified in the training set, i.e. D'. Then, the reduced testing set containing D' dimensional features, which can be regarded as S shown in Fig. 2(c), is used for intrusion detection.

4. Experiments

4.1. Experimental setup

4.1.1. The dataset

Since there is no standard dataset for intrusion detection, the dataset used in this paper is based on the KDD-Cup 99 dataset[1] containing 494,020 samples, which is the most popular and widely used in related work (c.f., Table 1). Specifically, each data sample represents a network connection represented by a 41-dimensional feature vector, in which 9 features are of the intrinsic types, 13 features are of the content type, and the remaining 19 features are of the traffic type. Each pattern of the dataset is labeled as belonging to one out of five classes, which are *normal* traffic and four different classes of attacks, i.e., *probing*, *denial of service* (DoS), *remote to local*

[1] http://www.sigkdd.org/kddcup/index.php?section=1999&method=data.

(R2L), and *user to root* (U2R). Therefore, this is a five-class classification dataset and the k value of k-means to extract cluster centers is set to 5.

The dataset is a large and high dimensional dataset. A dimensional reduction (or feature selection) step based on principal component analysis (PCA) has been considered in related work for example, to filter out unrepresentative features (e.g., [20,25]. However, as there is no standard answer about which features are well representative for intrusion detection, this study considers two different numbers of features, 6 and 19, in order to fully assess the performance of CANN. The 6-dimension KDD dataset contains 'land', 'urgent', 'num_failed_logins', 'num_shells', 'is_host_login', and 'num_outbound_cmds' [35]. On the other hand, the KDD dataset containing 19 features is based on the work of Zhang et al. [39].

4.1.2. Classifier design

The final step of CANN for classifier construction is based on the k-NN algorithm. One of the baseline classifiers to compare with CANN is the k-NN classifier. That is, the baseline k-NN classifier is trained and tested over the original (6 and 19 dimensional) KDD datasets without the CANN process. In particular, for the k value of the k-NN algorithm used in CANN and the baseline, we examine $k = 1,3,5,\ldots,25$ in order to obtain the best k-NN classifier for comparison.

Another baseline classifier based on the support vector machine (SVM), a popularly used baseline classifier in the literature, is considered. The polynomial kernel function is used to construct the SVM, in which the degree of the polynomial is set from 1 to 5 to obtain the SVM classifier providing the best performance for comparison.

The 10-fold cross validation method is used to train and test these classifiers, where the dataset is divided into 10 un-duplicated subsets, and any nine of the ten subsets are used for training and the remaining one for testing. Thus, the classifier will be trained and tested10 times.

4.1.3. Evaluation methods

In this study, we consider the rates of accuracy, detection and false alarms, which are widely used in literature, to evaluate the performance of intrusion detection (c.f., Table 1). They can be calculated by a confusion matrix as shown in Table 2.

Then, the rates of accuracy, detection and false alarm can be obtained by:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (5)$$

$$Detection\ Rate = \frac{TP}{TP + FP} \quad (6)$$

$$False\ Alarm = \frac{FP}{FP + TN} \quad (7)$$

Table 2
Confusion matrix.

↓ *Actual* \ *predicted* →	Normal	Intrusions (Attacks)
Normal	TN	FP
Intrusions (Attacks)	FN	TP

True Positives (TP): the number of malicious executables correctly classified as malicious.
True Negatives (TN): the number of benign programs correctly classified as benign.
False Positives (FP): the number of benign programs falsely classified as malicious.
False Negative (FN): the number of malicious executables falsely classified as benign.

Moreover, in order to understand the efficiency of CANN in terms of training and testing efforts, the time for executing all three steps will be examined and compared with the baseline classifiers.

4.2. Study I: the 6-dimensional dataset

Study I is based on the 6-dimensional KDD dataset. Table 3 shows the performance of CANN, TANN [35], and k-NN. The results show that CANN provides the highest accuracy rate of 99.761%, significantly outperforming TANN and k-NN. In addition, CANN can produce the highest detection rate and lowest false alarm rate.

Tables 4 and 5 show the confusion matrices of k-NN and CANN, respectively, for the 6-dimensional dataset. It is interesting that with the k-NN classifier, most of the normal accesses are recognized for probing U2R and R2L as DoS attacks, which results in a very high false alarm rate. However, almost all DoS attacks are classified correctly. This suggests that these 6-dimensonal features are not representative enough to allow k-NN to distinguish between these five different classes. In other words, the results obtained using k-NN for probing U2R and R2L attacks are very similar to the DoS attacks in the 6-dimensional feature space.

Although CANN performs the best, one weakness is that it totally misclassifies U2R and R2L attacks into the normal traffic class (see Table 5). This may be because these two attack categories are similar to the pattern of normal traffic after performing the CANN process. The proposed feature representation extracted from the 6-dimensonal dataset allows the classifier to distinguish between the normal traffic, probing, and DoS classes quite well.

4.3. Study II: the 19-dimensional dataset

The second study is based on a 19-dimensional KDD dataset. Table 6 shows the performance of CANN, k-NN, and SVM. As can be seen in this table, the k-NN classifier performs the best in terms

Table 3
Performances of CANN, k-NN, and SVM over the 6-dimensonal dataset.

	Accuracy	Detection Rate	False Alarm
CANN ($k = 1$)	99.76% (1)	99.99% (1)	0.003% (1)
TANN ($k = 21$)	93.87% (2)	93.39% (2)	28.69% (2)
k-NN ($k = 5$)	80.65% (3)	80.32% (3)	99.92% (3)

Table 4
Confusion matrix of k-NN over the 6-dimensonal dataset.

	Predicted					Accuracy (%)
	Normal	Probe	DoS	U2R	R2L	
Actual						
Normal	73		97,204	0	0	0.075
Probe	0	0	4107	0	0	100
Dos	12	0	391,446	0	0	99.99
U2R	0	0	52	0	0	0
R2L	0	0	1126	0	0	0

Table 5
Confusion matrix of CANN over the 6-dimensonal dataset.

	Predicted					Accuracy (%)
	Normal	Probe	DoS	U2R	R2L	
Actual						
Normal	97,275	0	2	0	0	99.99
Probe	0	4106	1	0	0	99.98
Dos	2	0	391,456	0	0	99.99
U2R	52	0	0	0	0	0
R2L	1126	0	0	0	0	0

Table 6
Performances of CANN, *k*-NN, and SVM over the 19-dimensonal dataset.

	Accuracy	Detection rate	False alarm
CANN (k = 1)	99.46% (2)	99.28% (2)	2.95% (2)
SVM (degree = 2)	95.37% (3)	98.97% (3)	4% (3)
k-NN (k = 1)	99.89% (1)	99.92% (1)	0.32% (1)

Table 7
Confusion matrix of *k*-NN over the 19-dimensonal dataset.

	Predicted					Accuracy (%)
	Normal	Probe	DoS	U2R	R2L	
Actual						
Normal	96,964	42	118	17	126	99.68
Probe	48	4045	11	0	3	98.49
Dos	52	6	391,394	3	3	99.98
U2R	29	1	3	9	10	17.31
R2L	80	6	4	3	1033	91.74

of accuracy and detection rates. The SVM classifier performs the worst. Although CANN does not outperform *k*-NN, the accuracy and detection rates of *k*-NN and CANN are very similar, being less than 1%. This indicates that there is no significant difference in their performances.

Tables 7–9 show the confusion matrices of *k*-NN, CANN, and SVM respectively, for the 19-dimensional dataset. It is worth noting here that *k*-NN can accurately recognize most normal accesses over the 19-dimensonal dataset, which is different from the result obtained using the 6-dimensonal dataset. However, these 19-dimensional features are not discriminative enough for *k*-NN to detect the U2R attacks.

On the other hand, although CANN can correctly classify some U2R and R2L attacks into the right attack groups, the accuracy rates are not satisfactory, i.e., 3.846% and 57.016%, respectively. This means that the distance based feature of CANN extracted from the 19-dimensional features only allows the classifier to better distinguish between the normal accesses, probing, and DoS classes. This result is similar to the distance based feature extracted from the 6-dimensional features which shows that CANN is not good at detecting U2R and R2L attacks. In summary, CANN is good at detecting the normal traffic, probing, and DoS classes, whereas SVM can be used for detecting the U2R and R2L classes.

These results indicate that the 19-dimensional features are more representative for intrusion detection, making SVM and *k*-NN perform very well. However, it is important to examine the run time of these classifiers. CANN only uses the one-dimensional feature for intrusion detection, but SVM and *k*-NN are trained and tested by the original features, which can result in high computational effort.

Table 8
Confusion matrix of CANN over the 19-dimensonal dataset.

	Predicted					Accuracy (%)
	Normal	Probe	DoS	U2R	R2L	
Actual						
Normal	94,398	221	2130	35	493	97.04
Probe	201	3598	306	1	1	87.61
Dos	1076	177	390,190	8	7	99.68
U2R	36	1	11	2	2	3.85
R2L	471	1	10	2	642	57.02

Table 9
Confusion matrix of SVM over the 19-dimensonal dataset.

	Predicted					Accuracy (%)
	Normal	Probe	DoS	U2R	R2L	
Actual						
Normal	93,367	3780	130	0	0	95.98
Probe	120	3967	20	0	0	96.59
Dos	20,760	44,639	324,329	1390	340	82.85
U2R	0	0	20	32	0	61.54
R2L	0	69	9	159	889	78.95

4.4. Efficiency evaluation

Table 10 shows the results obtained by comparing the run time of these classifiers.[2] Note that the data preparation time for *k*-NN includes data pre-processing and loading. For CANN, it includes the time for the processes of extracting cluster centers and nearest neighbors and new data formation. This comparison does not consider SVM because so much time is needed for training, e.g., over 100 h for the 19-dimensonal dataset.

As we can see, a much longer run time is needed for the dataset containing higher numbers of dimensions needs. Thus *k*-NN needs about 25 more hours to deal with the 19-dimensional dataset compared with the 6-dimensional one. However, with the CANN approach, increasing the number of dimensions for the dataset does not greatly affect the run time. In this case, we only need an additional 3 h.

If we consider the results from Study II, CANN does not outperform *k*-NN. However, CANN is still a good candidate for intrusion detection since it saves over two times the run time over the 6- and 19-dimensional dataset compared with *k*-NN, while still providing very similar performance to *k*-NN and SVM. In addition, with the 6-dimensional dataset, CANN provides an accuracy, detection rate, and false alarm of 99.76%, 99.99%, and 0.003% respectively, results which are better or similar to the best classifier for the 19-dimensional dataset, which for 99.89%, 99.92%, and 0.0289%, respectively.

Compare the run time with the most recent related works (i.e., testing times). Kim et al. [17] and Nadiammai and Hemalatha [23] obtained times of 11.2 and 8 s, respectively, whereas CANN requires about 13 s. However, it should be noted that it is very difficult to make a direct comparison between these works since the computing environments and relevant settings are different. The run time could certainly be enhanced by using more efficient computing equipment, but not the detection performance. Thus, although the methods discussed in these two works may be more efficient than CANN, they only provide about 99% accuracy and 0.15% false alarm rates, which are all lower than the ones produced by CANN. Moreover, CANN performs better than the winner of the KDD'99 contest, which provides 97.12% accuracy and 0.03 false alarm rates.

4.5. Discussion

Regarding the previous results, if we consider the average accuracy, detection rate, and false alarm rate, for the 6-dimensional dataset, CANN performs the best in terms of the detection rate and false alarm rate, whereas for the 19-dimensional dataset *k*-NN performs the best in terms of accuracy.

We further examine these two approaches over the five classes (including the four attack classes) in advance. Fig. 5 shows the difference in performance. To detect the normal, probe, and DoS classes, the CANN approach performs slightly better than *k*-NN.

[2] The software is based on Matlab 7 and carried out on an Intel Pentium 4, with 3.4 GHz CPU, and 1.5 GB RAM.

Table 10
Run time of CANN and *k*-NN.

		Data preparation	Training and testing	Total
6-dimension dataset	CANN	40 min (0.7 h)	1570 min (26 h)	1648 min (27 h)
	k-NN	20 min (0.3 h)	2765 min (46 h)	2785 min (46 h)
19-dimension dataset	CANN	180 min (3 h)	1608 min (27 h)	1750 min (30 h)
	k-NN	20 min (0.3 h)	4210 min (70 h)	4230 min (71 h)

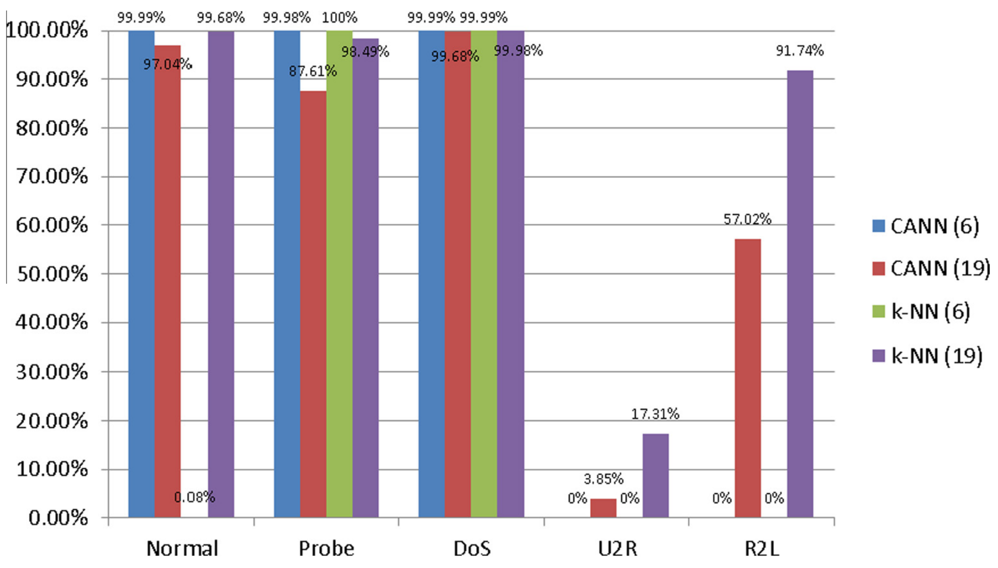

Fig. 5. The performances of CANN and *k*-NN over 5 classes.

However, *k*-NN can correctly detect some U2R and R2L cases, but CANN cannot.

These results show that with the 19-dimensional dataset U2R and R2L attacks are very hard to detect by all methods, except for *k*-NN. According to the definition of KDD-Cup 99, U2R is either unauthorized access to a local super user or administrator privileges by a local unprivileged user. These attacks are opportunities for exploitation where the hacker starts off with a normal user account but attempts to abuse vulnerabilities in the system in order to gain super user privileges e.g., perl, xterm. On the other hand, R2L refers to a remote user obtaining unauthorized user privileges on a local host. In the attack the user sends packets to a machine over the internet, to which she/he does not have access to in order to expose the machines vulnerabilities and exploit privileges which a local user would have, e.g., xlock, guest, xnsnoop, phf, sendmail dictionary, etc.

These results indicate that with 6 and 19 features the classifier cannot detect representative U2R attacks whereas with 19 features it can somewhat detect representative R2L attacks. Similarly, transforming the 6 and 19 features by CANN still cannot lead to effective detection of U2R and R2L attacks. According to Jeya et al. [15], the four attack classes have different representative features among the 41. Therefore, CANN is generally suitable for detection of normal or attack cases in the binary classification problem, with a false alarm rate of 0.003%. However, for U2R and R2L attacks, for which there are fewer cases among all accesses, existing approaches still have room for improvement.

5. Conclusion

This paper presents a novel feature representation approach that combines cluster centers and nearest neighbors for effective and efficient intrusion detection, namely CANN. The CANN approach first transforms the original feature representation of a given dataset into a one-dimensional distance based feature. Then, this new dataset is used to train and test a *k*-NN classifier for classification.

The experimental results show that CANN performs better than the *k*-NN and SVM classifiers over the original 6-dimension dataset, providing higher accuracy and detection rates and a lower false alarm rate. On the other hand, CANN performs similar to the *k*-NN and SVM classifiers over the original 19-dimension dataset. However, the advantage of CANN is that it requires less computational effort than the *k*-NN or SVM classifiers trained and tested by the two original datasets. In other words, although CANN requires additional computation to extract the distance based features, the training and testing (i.e., detection) time is greatly reduced since the new dataset only contains one dimension.

As to the limitations of this research CANN cannot effectively detect U2L and R2L attacks, which means that this one-dimensional distance based feature representation is not able to well represent the pattern of these two types of attacks. This is an issue that future work can look into. One possibility is to consider the weight for the distances between the data to each of the cluster centers and its nearest neighbor. Alternatively, before performing CANN, outlier detection and removal can be employed in order to first filter out noisy or bad data from the given dataset. Finally, as CANN is applicable to the 5-class intrusion detection problem, other domain datasets including different numbers of dimensions and classes can be used to examine its effectiveness.

References

[1] M.S. Abadeh, J. Habibi, Z. Barzegar, M. Sergi, A parallel genetic local search algorithm for intrusion detection in computer networks, Eng. Appl. Artif. Intell. 20 (8) (2007) 1058–1069.
[2] Z.A. Baig, S.M. Sait, A. Shaheen, GMDH-based networks for intelligent intrusion detection, Eng. Appl. Artif. Intell. 26 (7) (2013) 1731–1740.
[3] Y. Chen, A. Abraham, B. Yang, Hybrid flexible neural-tree-based intrusion detection systems, Int. J. Intell. Syst. 22 (2007) 337–352.

[4] E. de la Hoz, E. de la Hoz, A. Ortiz, J. Ortega, A. Martinez-Alvarez, Feature selection by multi-objective optimisation: application to network anomaly detection by hierarchical self-organising maps, Knowl.-Based Syst. 71 (2014) 322–338.
[5] W. Feng, Q. Zhang, G. Hu, J.X. Huang, Mining network data for intrusion detection through combining SVMs with ant colony networks, Future Gener. Comput. Syst. 37 (2014) 127–140.
[6] A.S. Eesa, Z. Orman, A.M.A. Brifcani, A novel feature-selection approach based on the cuttlefish optimization algorithm for intrusion detection systems, Expert Syst. Appl. 42 (5) (2015) 2670–2679.
[7] P. García-Teodoro, J. Dıaz-Verdejo, G. Maciá-Fernández, E. Vázquez, Anomaly-based network intrusion detection: techniques, systems and challenges, Comput. Secur. 28 (2009) 18–28.
[8] G. Giacinto, R. Perdisci, M. Del Rio, F. Roli, Intrusion detection in computer networks by a modular ensemble of one-class classifiers, Inf. Fusion 9 (2008) 69–82.
[9] H. Guan, J. Zhou, M. Guo, A class-feature-centroid classifier for text categorization, in: Proceedings of the International Conference on World Wide Web, 2009, pp. 201–209.
[10] E.-H. Han, G. Karypis, Centroid-based document classification: analysis and experimental results, in: Proceedings of the European Conference on Principles of Data Mining and Knowledge Discovery, 2000, pp. 424–431.
[11] J.V. Hansen, P.B. Lowry, R.D. Meservy, D.M. McDonald, Genetic programming for prevention of cyberterrorism through dynamic and evolving intrusion detection, Decis. Support Syst. 43 (2007) 1362–1374.
[12] W. Hu, W. Hu, S. Maybank, AdaBoost-based algorithm for network intrusion detection, IEEE Trans. Syst. Man Cybernet. – Part B: Cybernet. 38 (2) (2008) 577–583.
[13] A.K. Jain, R.P.W. Duin, J. Mao, Statistical pattern recognition: a review, IEEE Trans. Pattern Anal. Mach. Intell. 22 (1) (2000) 4–37.
[14] A.K. Jain, M.N. Murty, P.J. Flynn, Data clustering: a review, ACM Comput. Survey 31 (3) (1999) 264–323.
[15] P.G. Jeya, M. Ravichandran, C.S. Ravichandran, Efficient classifier for R2L and U2R attacks, Int. J. Comput. Appl. 45 (21) (2012) 28–32.
[16] L. Khan, M. Awad, B. Thuraisingham, A new intrusion detection system using support vector machines and hierarchical clustering, VLDB J. 16 (2007) 507–521.
[17] G. Kim, S. Lee, S. Kim, A novel hybrid intrusion detection method integrating anomaly detection with misuse detection, Expert Syst. Appl. 41 (4) (2014) 1690–1700.
[18] Y. Li, L. Guo, An active learning based TCM-KNN algorithm for supervised network intrusion detection, Comput. Secur. 26 (2007) 459–467.
[19] S.-W. Lin, K.-C. Ying, C.-Y. Lee, Z.-J. Lee, An intelligent algorithm with feature selection and decision rules applied to anomaly instruction detection, Appl. Soft Comput. 12 (12) (2012) 3285–3290.
[20] G. Liu, Z. Yi, S. Yang, A hierarchical intrusion detection model based on the PCA neural networks, Neurocomputing 70 (2007) 1561–1568.
[21] G.A. Marin, Network security basics, IEEE Secur. Privacy 3 (6) (2005) 68–72.
[22] T. Mitchell, Machine Learning, McGraw Hill, New York, 1997.
[23] G.V. Nadiammai, M. Hemalatha, Effective approach toward Intrusion Detection System using data mining techniques, Egypt. Infor. J. 15 (1) (2014) 37–50.
[24] N.J. Nisson, Introduction to Machine Learning, MIT Press, USA, 1996.
[25] T. Ozyer, R. Alhajj, K. Barker, Intrusion detection by integrating boosting genetic fuzzy classifier and data mining criteria for rule pre-screening, J. Network Comput. Appl. 30 (2007) 99–113.
[26] S. Peddabachigari, A. Abraham, C. Grosan, J. Thomas, Modeling intrusion detection system using hybrid intelligent systems, J. Network Comput. Appl. 30 (2007) 114–132.
[27] P. Sangkatsanee, N. Wattanapongsakorn, C. Charnsripinyo, Practical real-time intrusion detection using machine learning approaches, Comput. Commun. 34 (18) (2011) 2227–2235.
[28] S.T. Sarasamma, Q.A. Zhu, J. Huff, Hierarchical Kohonenen net for anomaly detection in network security, IEEE Trans. Syst. Man Cybernet. – Part B: Cybernet. 35 (2) (2005) 302–312.
[29] S. Shin, S. Lee, H. Kim, S. Kim, Advanced probabilistic approach for network intrusion forecasting and detection, Exp. Syst. Appl. 40 (1) (2013) 315–322.
[30] T. Shon, J. Moon, A hybrid machine learning approach to network anomaly detection, Inform. Sci. 177 (2007) 3799–3821.
[31] T. Shon, X. Kovah, J. Moon, Applying genetic algorithm for classifying anomalous TCP/IP packets, Neurocomputing 69 (2006) 2429–2433.
[32] A. Tajbakhsh, M. Rahmati, A. Mirzaei, Intrusion detection using fuzzy association rules, Appl. Soft Comput. 9 (2009) 462–469.
[33] X. Tong, Z. Wang, H. Yu, A research using hybrid RBF/Elman neural networks for intrusion detection system secure model, Comput. Phys. Commun. 180 (10) (2009) 1795–1801.
[34] C.-F. Tsai, Y.-F. Hsu, C.-Y. Lin, W.-Y. Lin, Intrusion detection by machine learning: a review, Expert Syst. Appl. 36 (10) (2009) 11994–12000.
[35] C.-F. Tsai, C.-Y. Lin, A triangle area based nearest neighbor approach to intrusion detection, Pattern Recogn. 43 (1) (2010) 222–229.
[36] G. Wang, J. Hao, J. Ma, L. Huang, A new approach to intrusion detection using artificial neural networks and fuzzy clustering, Exp. Syst. Appl. 37 (9) (2010) 6225–6232.
[37] C. Xiang, P.C. Yong, L.S. Meng, Design of multiple-level hybrid classifier for intrusion detection system using Bayesian clustering and decision trees, Pattern Recogn. Lett. 29 (2008) 918–924.
[38] C. Zhang, J. Jiang, M. Kamel, Intrusion detection using hierarchical neural network, Pattern Recogn. Lett. 26 (2005) 779–791.
[39] X.-Q. Zhang, C.H. Gu, J.J. Lin, Intrusion detection system based on feature selection and support vector machine, in: International Conference on Communications and Networking in China, 2006, pp. 1–5.