Paper

L-SCANN: Logarithmic Subcentroid and Nearest Neighbor

Tohari Ahmad and Kharisma Muchammad

Department of Informatics, Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia

Abstract—Securing a computer network has become a need in this digital era. One way to ensure the security is by deploying an intrusion detection system (IDS), which some of them employs machine learning methods, such as *k*-nearest neighbor. Despite its strength for detecting intrusion, there are some factors, which should be improved. In IDS, some research has been done in terms of feature generation or feature selection. However, its performance may not be good enough. In this paper, a method to increase the quality of the generated features while maintaining its high accuracy and low computational time is proposed. This is done by reducing the search space in training data. In this case, the authors use distance between the evaluated point and the centroid of the other clusters, as well as the logarithmic distance between the evaluated point and the subcentroid of the respective cluster. Besides the performance, the effect of homogeneity in extracting centroid and subcentroid on the accuracy of the detection model is also evaluated. Based on conducted experiment, authors find that the proposed method is able to decrease processing time and increase the performance. In more details, by using NSL-KDD 20% dataset, there is an increase of 4%, 2%, and 6% from those of TANN in terms of accuracy, sensitivity and specificity, respectively. Similarly, by using Kyoto 2006 dataset, proposed method rises 1%, 3%, and 2% than those of TANN.

Keywords*—*clustering, feature transformation, information security, network security.

1. Introduction

Transferring data through a computer network has become an essential need in this digital era. This has made it easy for people to communicate each other. Since the data are transmitted in a public network, the security should have more attention. It is because anyone may have access to it. This works especially to sensitive data whose access must be restricted to legitimate users only. Moreover, these private data, such as medical and financial data, may become a target of attacks. Therefore, there must be a mechanism to protect it.

Based on its objective, data protection can be divided into two categories. The first is securing data in a file or transaction. This is done by using either encryption [1], transformation [2] or steganography [3]. The second is protecting data which reside in a network. In this case, the control is usually carried out by an intrusion detection system (IDS). Based on [4], IDS can be grouped into two categories. The first is signature-based IDS, which uses records of known attack. If an activity matches with any of that record, then the IDS will trigger an alarm. The second is anomaly-based IDS, which uses a model representing a normal activity. Any flow that deviates too far from the model will trigger an alarm.

That first type of IDS has lower resource consumption than the second. However, it needs constantly update the signatures, which have been stored in the database. Different from this, the second type of IDS does not need signatures to detect specific packets. So, it is able to detect unknown attacks to some degree. Nevertheless, it has a slightly higher false positive rate than the first [4].

There is some research, which have been introduced to solve those problems. Their results, however, may not be so optimal. In more details, there are at least two factors that the anomaly-based IDS should be improved, those are [5]:

- outlier detection – in common anomaly-based IDS, the detection model only uses normal data. It is assumed that data which are outside of the detection model is anomaly, as specified in [6]. In real life, however, this assumption may not be realistic;

- cost of false alarm – unlike other applications of machine learning, such as hand written recognition or product recommendation, in this case false alarm requires much time and energy to verify. Ignoring an alarm is not a good idea since a true alarm can cause the whole system down.

Some research, such as [7] and [8], has proposed a feature transformation method to reduce the false alarm rate and at the same time maintain high detection rate. This method needs to use whole training data for the final classification step. If the training data is too large, then the time needed to classify an activity rises greatly. In authors' previous research [9], a method to reduce the size of training data and to transform them for the classification purpose have been proposed. This is done by measuring the distance between the respective point and centroids, as well as and logarithmic distance between the respective point and subcentroid. In this paper, we expands and further validate this method.

The rest of this paper is organized as follows. Section 2 contains previous works, which relate to this research. Section 3 explains the proposed method. The Sections 4 and 5 describe the experimental design and result, respectively. Finally, the conclusions in the Section 6 are drawn.

2. Related Works

Some research has been introduced to classify data by using distance to the centroid. This includes [7], [8], [10]–[14], which have used that distance to generate features. The basic idea of those methods is similar, that is, the distance between a point and the respective centroid is used to differentiate data within dataset. The differences between that researches are how the centroid is extracted and how that distances is used. Some methods use *k*-means by using the previously defined number of clusters [7], [8], [14] and by partitioning the dataset based on their label and by extracting the centroids according to their average features in each partition [10], [11], [13].
The research [7] and [8] are proposed in order to address problems in [5]. Both methods have achieved a relatively low false positive rate while maintaining its high detection rate. Here, they employ feature transformation to extract new features and to classify the data by using *k*-Nearest Neighbor (*k*NN). For handling the dynamic nature of the data stream, some research such as [15], [16] propose to use the affinity propagation [17] for selecting exemplars. This is to be applied to the on line clustering algorithms. Some methods, which are often used for classification, are described as follows.

2.1. TANN

Triangle Area and Nearest Neighbor (TANN) is proposed in [8]. It works by assuming that the centroid of a dataset can be used to differentiate data, and the distance of unknown data to centroid of other clusters can be used for classification. This method generates features by using data, which have been extracted, and by utilizing any 2 out of 5 centroids. This method requires a parameter *k*, similar to that used in *k*NN. The overall framework of this method can be observed in Fig. 1. Here, TANN works as follows:

1. **Extracting centroids** – by using *k*-means, this method extract 5 centroids. This number is chosen by assuming that there are 5 classes: 1 normal and 4 type of attacks.
2. **Generating new features** – the transformed data are generated by summing the area of all possible triangles formed by using the old data and any two of five centroids, which have been extracted in step 1. Possible triangles, which are formed are depicted in Fig. 2.
3. **Training and testing *k*NN** – the new data are divided into training and testing sets. Next, the testing set is classified by using *k*NN.

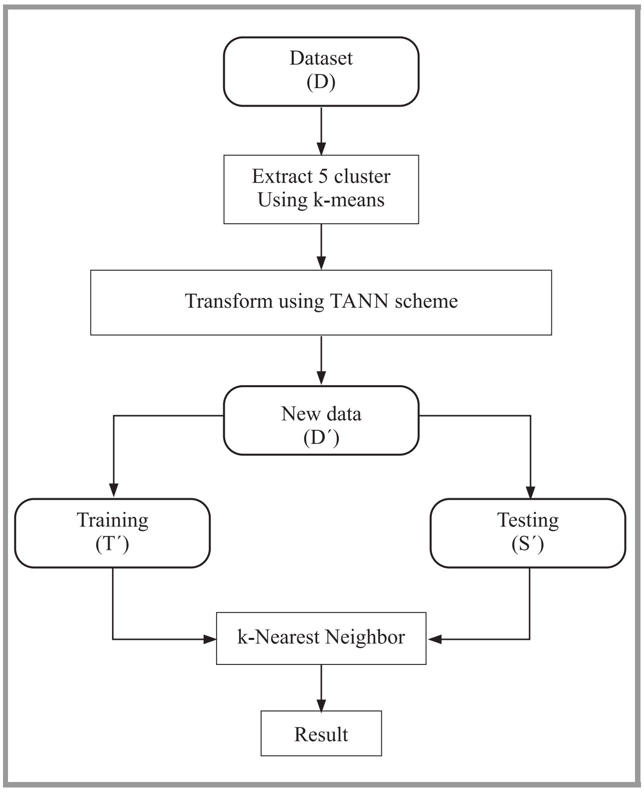

Fig. 1. TANN workflow.

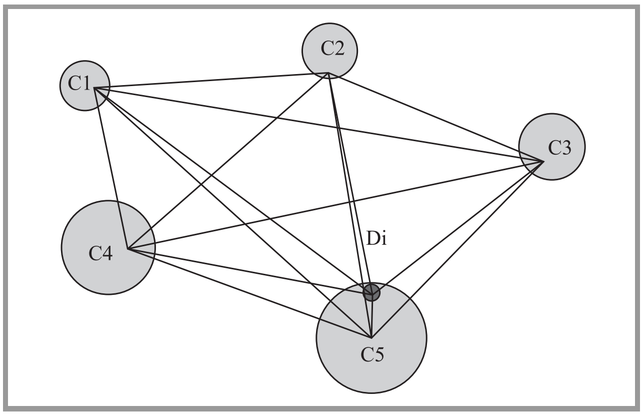

Fig. 2. TANN scheme.

A common problem of TANN is that it needs to use all training data in order to classify the testing or other data. Consequently, if the size of the training data is large, the time needed to process rises significantly. To overcome this problem, some applications use parallel searching by using Graphic Processing Unit (GPU) such as [18] and [19]. However, not all computers have GPU, which is able to do it. Therefore, reducing the search space is one of the feasible solution for this problem.

2.2. Bisecting k-Means

This clustering method is proposed by [20] to classify documents. It works as follows:

1. Pick a cluster to split. In this case, they use the largest cluster.
2. Split the selected cluster into two clusters by using ordinary *k*-means.

3. Repeat step 1 and 2 until the desired number of clusters has been reached.

In [20], authors also propose to use the centroids, which have been extracted from those steps as initial centroids for basic *k*-means. In addition, they believe that their method works better than simple agglomerative clustering. This is because the agglomerative clustering sometimes merges data from different classes, which may happen in the early clustering step. If so, then this mistake cannot be fixed. Divisive clustering on the other hand, may split clusters whose members actually have same labels. Fortunately, this mistake can be fixed in later phases.

A weakness of this method is, similar to that of *k*-means, we need to know the number of required clusters. Unfortunately, we may not have this information every time.

3. Proposed Method

The proposed method is inspired by TANN [8], CANN [7], and comparison of clustering techniques [20]. Similar to [8] and [7], the distance between the evaluated point and the respective centroid is used to transform data while the difference is on how the extraction of centroid is performed. In more details, the proposed method differs in the extraction process and in treating the subcentroid to add differentiating power to the classification method. The overall proposed method is depicted in Fig. 3, where T is training dataset, S is testing dataset and $D = T \cup S$.

The proposed method is developed based on authors' previous research [9]. In this paper, the scope is extended as follows.

1. The dataset used here is bigger than that in [9].
2. The use of entropy is explored as well as Gini impurity index. Additionally, their combination is also investigated.
3. The use of bisecting *k*-means and logarithmic distance in the transformation process is investigated. This is intended to evaluate their effect on the performance.

As shown in Fig. 3, the first step is to divide the dataset into testing and training. In the experiment, authors further divide the data into 10 partitions. This is to carry out cross validation with each partition whose composition is proportional.

Next is to cluster the training data by using a modified variant of bisecting *k*-means whose diagram is presented in Fig. 4. This method works as follows:

1. Split the dataset to two clusters by using *k*-means.
2. If the impurity of any resulting cluster is higher than the user defined threshold U, then split it by using ordinary *k*-means into two clusters.

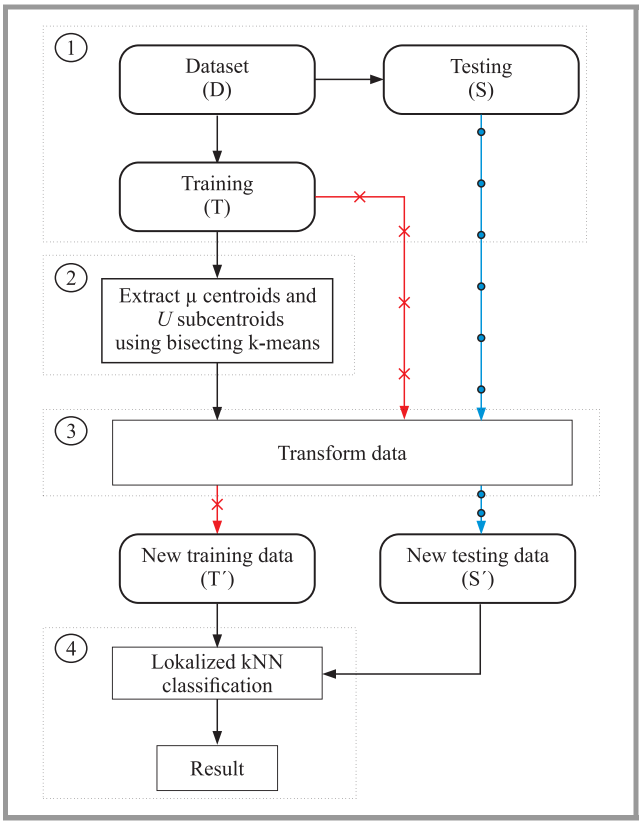

Fig. 3. L-SCANN workflow.

3. Repeat step 2 until there is no cluster whose impurity is higher than U, or no more impurity can be reduced. At this stage, the number of clusters n has been obtained.

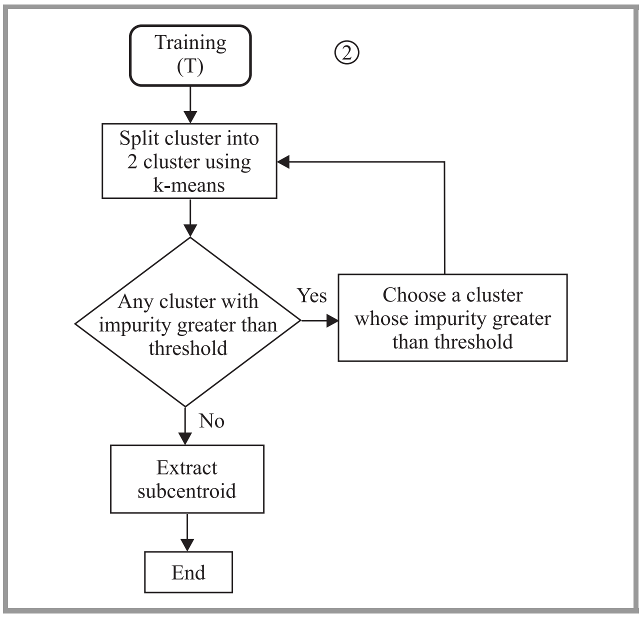
Training (T)
2
Split cluster into 2 cluster using k-means
Any cluster with impurity greater than threshold
Yes
Choose a cluster whose impurity greater than threshold
No
Extract subcentroid
End

Fig. 4. Variant of bisecting *ki*-means used.

This step is intended to solve the problems experienced by previous research, such as [7] and [8]. That is, the number of needed clusters to give decent performance is not always available. In addition, data with a same label is not always grouped into one cluster.

Different from other research, such as [21], which uses a genetic algorithm to iteratively search the best centroid in partially labeled data, a variant of bisecting k-means is employed. Furthermore, they make use of Gini impurity index, mean square error and combination of both for measuring the quality of the generated clusters. Here, authors take Gini impurity index, entropy and combination of them as stopping criteria as previously described.

Let O and be m be the number of specified subcentroids in each cluster whose value is obtained from an experiment, and the number of members in a cluster, respectively. There are three possibilities of subcentroids according to those O and m values:

- $m > O$ – subcentroids are extracted by using k-means whose k is equal to O,
- $m = O$ – each member of the clusters is treated as subcentroid,
- $m = O$ – in this paper, we assumed that the cluster is homogeneous. So, there is no subcentroid, which can be extracted.

The following step is transforming the data such that the distance of the evaluated data can be collected. In authors' previous works [9], the transformation of the data is done by summing two types of distance: between the point and centroids, and between the point and the subcentroids in the same cluster. Let D_i be the i-th data being evaluated, C_j be the j-th centroid of each cluster, L_k be the k-th subcentroid of the respective cluster. It is shown in Fig. 5 that D_i is closer to C_5 than to other centroids. Therefore, C_5 is firstly assigned as its centroid.

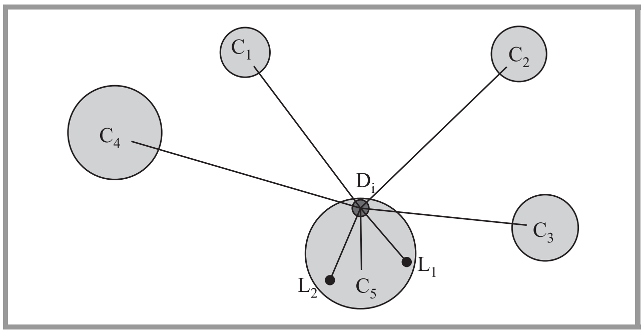

Fig. 5. L-SCANN scheme.

Let $Dist1_i$ be the distance between the i-th data and all centroids in each cluster, n be the number of clusters extracted from the clustering step, and $||C_j - D_i||$ be the Euclidean distance between D_i and C_j. This distance can be depicted in Eq. (1). Next, the distance between the evaluated data and its subcentroids, $Dist2_i$, is presented in Eq. (2), where $||D_i - L_k||$ is the Euclidean distance between D_i and L_k, and O is the number of subcentroids in the respective cluster. If the distance between D_i and L_k is 0 (i.e. overlapping), then the distance is reset. The new transformed data, D'_i is then obtained by summing $Dist1_i$ and $Dist2_i$, as provided in Eq. (3). This transformation step is repeated until all D_i have been processed. That is, each data in training set T will be transformed into new training data T' and each data in testing set S will also be transformed into new testing data S'. This is noted by red (×) and blue (•) arrows, respectively in Fig. 3. Each new training data T' is then assigned to its respective cluster.

As an alternative, it is also possible to remove the logarithmic calculation, as shown in Eq. (4) where D''_i is the transformation results. This is because logarithmic calculation may take a little bit longer.

The final step of proposed method is classifying the data in testing set S. For this purpose, we classify the evaluated point D_i as member of the cluster whose centroid is the nearest. Next, the three nearest transformed neighbor points are selected, which D_i is assigned a label similar to the dominant label. That is, if most of those three are normal, than D_i is also normal. If most of those three are attack, than D_i is also attack.

An example of this classification process is presented in Fig. 6, where new data D_i is being classified. In Fig. 6a, the centroid C_j with the smallest distance to D_i is looked for. In this case, C_5 is found in a cluster, whose subcentroids are L_1, L_2 and L_3. The respective cluster has T_1, T_2, T_3, T_4 and T_5 as the members. By using Eq. 3, its transformation value is calculated as depicted in Fig. 6b. Finally, the three closest transformed members are selected, which are: T'_3,

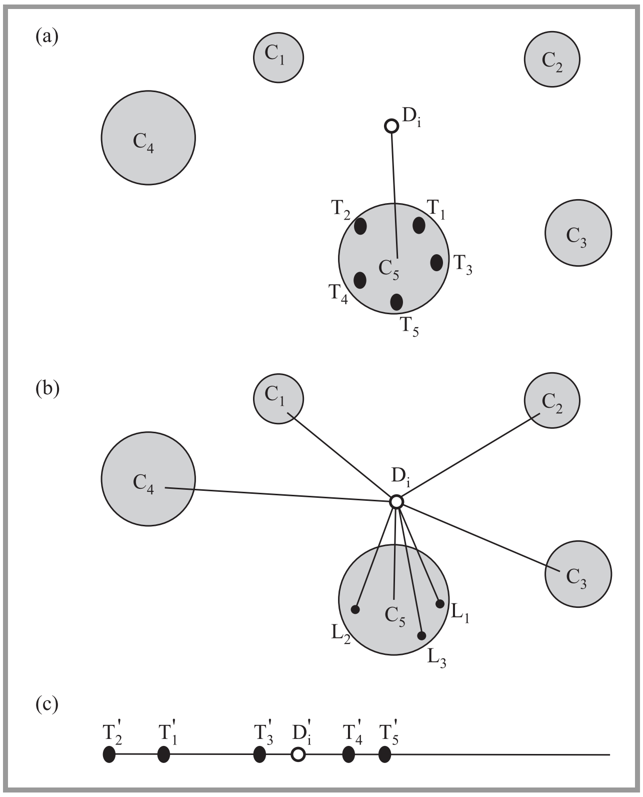

Fig. 6. Example of classification process.

T_4' and T_5', see Fig. 6c. The majority labels of them are then assigned to D_i:

$$Dist1_i = \sum_{j=1}^{n} ||C_j - D_i||\,, \tag{1}$$

$$Dist2_i = \sum_{L=1}^{O} \begin{cases} \ln(||D_i - L_k||), & \text{if } D_i \neq L_k \\ 0, & \text{otherwise} \end{cases}, \tag{2}$$

$$D_i' = Dist1_i + Dist2_i\,. \tag{3}$$

$$D_i'' = \sum_{j=1}^{n} ||C_j - D_i|| + \sum_{L=1}^{O} ||D_i + L_k||\,. \tag{4}$$

4. Experimental Design

The experiment is carried out in Python 2.6 with scipy and numpy package in 64-bit Ubuntu Linux using Intel Core i5-2410M processor with 4 GB of RAM.

In the modified bisecting k-means, we use an impurity index as the stopping criteria in finding clusters. That is, Gini impurity index, entropy, and their combination as provided in Eqs. (5)–(7) respectively, where m is the number of class labels and f_i is the frequency of i-th label. Those first two are common index while the third is our proposed combination method. The Eq. (7) is designed based on the facts that gini index has a range between 0 to 0.5 in two label cases, and the entropy index has a range between 0 to 1 in any case. In addition, we use two labels, normal and attack. For a deeper analysis, the attack is further divided into subclasses (sublables) for NSL-KDD.

$$I_G = 1 - \sum_{i=1}^{m} f_i^2\,, \tag{5}$$

$$H = -\sum_{i=1}^{m} f_i log_2 f_i\,, \tag{6}$$

$$Combined = \frac{H + 2I_G}{2}\,. \tag{7}$$

4.1. Dataset

In the experiment, we use NSL-KDD 20% [22] and a subset of Kyoto 2006 [23] datasets. In NSL–KDD, we remove protocol_type, service and flag because those three are categorical types. In Kyoto 2006, we use a subset which was taken in 20 July 2009 whose unused attributes are removed, i.e. flag, start_time, and duration. Additionally, redundant records from Kyoto 2006 are then removed. At the end, we have 25192 records in NSL-KDD and 107213 records in Kyoto 2006. The composition of NSL-KDD and Kyoto 2006 is available in Table 1.

Table 1
Dataset composition

NSL-KDD			
Label	Sublabel	Number	
Attack	Dos	9234	11743
	Probe	2289	
	U2R	11	
	R2L	209	
Normal	Normal	13449	
Kyoto 2006			
Label		Number	
Attack		52554	
Normal		64659	

4.2. Parameter Tested

The proposed method requires two parameters to work. Those are the impurity index U, and the assumption number of subcentroids O. We define $U \in \{0.1, 0.2, 0.3\}$ and $O \in \{4, 5, 6\}$. In the experiment, we combine all possibilities of those values, such that, we have 9 combinations. The number of subclusters is not more than 6 in order to ensure that the cluster with lower number members is homogeneous. The value of impurity index is not larger than 0.3 because it is over half of Gini impurity index, whose range is between 0 and 0.5, for 2 labels. Here, three types of impurity index as previously described are used.

4.3. Evaluation Criteria

The proposed method is evaluated in terms of accuracy, sensitivity, and specificity as specified in Eqs. (8), (9), and (10), respectively. In this case, TP is true positive (correctly classified attack), TN is true negative (correctly classified normal), FP is false positive (misclassified normal), and FN is false negative (misclassified attack). Another evaluation criterion is running time, to see how search-space reduction affect the speed of the algorithm.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}\,, \tag{8}$$

$$\text{Sensivity} = \frac{TP}{TP + FN}\,, \tag{9}$$

$$\text{Specificity} = \frac{TN}{TN + FP}\,. \tag{10}$$

5. Experimental Results

The experiment results are provided in this section. These comprise that of proposed method as well as TANN for the comparison purpose. In this case, we firstly present the results of TANN in order to make it easier to compare.

5.1. TANN

Accuracy, sensitivity, and specificity of TANN on NSL-KDD and Kyoto 2006 can be observed in Table 2, where Acc, Sen and Spe represent accuracy, sensitivity and specificity respectively (in percent) and time depicts processing time (in hour).

In NSL-KDD dataset, increasing the value k of kNN leads to raising the value of accuracy and sensitivity but reducing specificity. The table also shows that the processing time is not affected by k. When it goes up from 3 to 5, the processing time drops. However, when it then grows to 7, the time also increases.

It is depicted that the best results is achieved by using $k = 5$ where accuracy, sensitivity and specificity are 96.80%, 95.92% and 97.64%, respectively with processing time 9.15 hours. It is worth noting that the processing time raises about 16 times when the dataset is 4 times larger than the size of NSL-KDD.

Table 2
Performance of TANN on different dataset

k	Acc [%]	Sen [%]	Spe [%]	Time (hour)
NSL-KDD				
3	93.08	94.85	91.53	0.47
5	93.36	95.86	91.17	0.39
7	93.51	96.32	91.05	0.56
9	93.56	96.48	91.02	0.42
11	93.60	96.60	90.99	0.50
13	93.61	96.66	90.95	0.53
15	93.54	96.62	90.86	0.53
Kyoto 2006				
3	96.73	95.79	97.63	9.12
5	96.80	95.92	97.64	9.15
7	96.69	95.84	97.51	9.14

5.2. Proposed Method with Gini Impurity Index

The experimental results of the proposed method which uses the gini impurity index on NSL-KDD can be observed in Table 3, where Clu represents the average number of generated clusters. Based on this result, we find that the highest accuracy, sensitivity and specificity level is achieved when the impurity threshold and the number of subcentroids are 0.1 and 4, respectively. It is also shown that increasing the impurity value results to decreasing those performances, and raising the number of subcentroids may cause the performances go up to some extent. It can be inferred that smaller number of both impurity index and subcentroids holds the best performance.

Compared to the performance of TANN, proposed method produces significantly higher result in terms of accuracy and specificity, but slightly lower in sensitivity. At the same time, the time needed to process the whole dataset goes down.

The experimental results which are obtain on Kyoto 2006 are depicted in Table 3. It shows that increasing the impurity index leads to decreasing the performance. In addition, higher number of clusters results to a little higher level of accuracy and sensitivity. However, this causes the specificity slightly lower. As specified in Table 3, it also found that proposed method is able to decrease the processing time from 9 to 2 hours for 100,000 records.

This result represents the characteristics of dataset we used. In this case, the average number of generated clusters in NSL-KDD is much higher than that of Kyoto 2006. It is very likely that NSL-KDD comprises heterogeneous data, which are more sparsely distributed across the initial clusters than that of Kyoto 2006. Furthermore, it is also found that in NSL-KDD, most clusters ($> 50\%$) are small with 3 or lower members, while those in Kyoto 2006 mostly ($> 50\%$) have 6 or more members.

5.3. Proposed Method with Entropy Index

The experimental result of the proposed method by using entropy index on NSL-KDD can be observed in Table 4. It depicts that an increase of impurity threshold results to a decrease of accuracy, sensitivity, and specificity; and raising the number of subcentroids, on the other hand, slightly change the accuracy, sensitivity, and specificity.

Similar to the experiment on NSL-KDD, the use of entropy on Kyoto 2006 dataset presents better results than that of gini index, as shown in Table 4. Furthermore, it also gives similar trends, that higher entropy values result in slightly lower performance. In addition, the number of subcentroids does not much affect the performance.

Based on this experimental result, it can be inferred that in general, the use of entropy in the proposed method is more appropriate to the performance than Gini impurity index. This is because by using entropy, the number subcentroids resulted from the clustering process is much higher than that of Gini index. Furthermore, the overall performance is also better than that of TANN.

5.4. Proposed Method with Combined Index

The experimental results with combination of Gini and entropy indexes are in Table 5. Similar to the results with either Gini or entropy indexes, that with their combination is, in general, down due to rising this index. Also, an increase of the number of subcentroids leads to slightly changes in the performances.

The overall performance resulted from the use of this combination index in both NSL-KDD and Kyoto 2006 datasets is lower than that of entropy, depending on the environment, which is represented by the parameter values. It is possible that this performance is affected by the resulted cluster numbers. That is, the accuracy, sensitivity and specificity are more or less proportional to the the number of generated clusters.

Table 3
Results of proposed method using Gini impurity index on different dataset

NSL-KDD															
Subcentroid	Gini impurity index														
	0.1					0.2					0.3				
	Acc [%]	Sen [%]	Spe [%]	Clu	Time [h]	Acc [%]	Sen [%]	Spe [%]	Clu	Time [h]	Acc [%]	Sen [%]	Spe [%]	Clu	Time [h]
4	94.95	91.71	97.78	541.70	0.32	94.11	90.53	97.23	354.00	0.23	93.13	88.93	96.80	218.40	0.16
5	94.49	90.73	97.78	546.70	0.29	93.33	88.79	97.29	351.60	0.21	93.48	89.54	96.92	217.50	0.15
6	94.50	90.83	97.71	539.00	0.31	93.33	88.66	97.41	347.80	0.27	92.97	88.33	97.03	217.20	0.16
Kyoto 2006															
4	98.40	98.09	98.70	58.40	1.975	98.28	98.03	98.52	42.1	2.031	97.45	97.52	97.38	24.8	2.08
5	98.41	98.13	98.69	58.50	1.956	98.29	98.07	98.07	42	2.036	97.49	97.62	97.36	24.8	2.105
6	98.43	98.16	98.69	58.40	2.001	98.31	98.07	98.53	41.9	2.04	97.51	97.67	97.35	24.8	2.061

Table 4
Results of proposed method using entropy index on different datasets

NSL-KDD															
Subcentroid	Entropy														
	0.1					0.2					0.3				
	Acc [%]	Sen [%]	Spe [%]	Clu	Time [h]	Acc [%]	Sen [%]	Spe [%]	Clu	Time [h]	Acc [%]	Sen [%]	Spe [%]	Clu	Time [h]
4	98.28	98.68	97.93	716.4	0.343	95.28	92.64	97.58	593.4	0.353	94.71	91.20	97.78	540.6	0.336
5	98.28	98.65	97.95	715.9	0.348	95.70	93.41	97.70	596	0.348	94.96	91.74	97.78	541.3	0.341
6	98.26	98.63	97.94	716	0.386	95.02	91.71	97.91	596.7	0.415	94.47	90.70	97.76	542.4	0.356
Kyoto 2006															
4	98.93	98.64	99.21	199.7	1.525	98.65	98.50	98.80	68.5	1.7525	98.38	98.06	98.69	58.5	1.72
5	98.93	98.64	99.21	197.3	1.58	98.60	98.50	98.70	69.5	1.821	98.39	98.08	98.69	58.5	1.863
6	98.94	98.65	99.22	199.4	1.58	98.64	98.48	98.79	68.4	1.848	98.41	98.10	98.70	58.5	1.83

Table 5
Results of proposed method using combined index on different dataset

NSL-KDD															
Subcentroid	0.1					0.2					0.3				
	Acc [%]	Sen [%]	Spe [%]	Clu	Time [h]	Acc [%]	Sen [%]	Spe [%]	Clu	Time [h]	Acc [%]	Sen [%]	Spe [%]	Clu	Time [h]
4	98.31	98.57	98.09	705.1	0.361	94.37	90.45	97.80	569	0.348	94.57	91.08	97.62	507.7	0.331
5	98.30	98.69	97.96	704.2	0.36	94.23	90.21	97.75	565.2	0.366	94.08	89.94	97.69	512	0.331
6	98.28	98.63	97.98	703.2	0.293	94.53	90.85	97.74	565.8	0.278	94.84	91.55	97.70	510.8	0.261
Kyoto 2006															
4	98.86	98.57	99.13	152.7	1.598	98.59	98.38	98.78	65	1.726	98.38	98.09	98.66	57.7	1.73
5	98.87	98.60	99.13	153	1.795	98.54	98.31	98.77	64.7	1.968	98.31	97.95	98.66	57.7	1.978
6	98.91	98.69	99.12	152.9	1.806	98.57	98.36	98.78	65.2	2.035	98.41	98.14	98.66	57.7	2.006

5.5. Proposed with k-Means

From the previous experiments we find that the best result is achieved when the number of subclusters is 705 and 200 for NSL-KDD and Kyoto 2006 datasets. Based on these numbers and the previous cluster number of TANN (i.e. 5), we perform experiments by using the common *k*-means method whose result is provided in Table 6 for NSL-KDD and Kyoto 2006, respectively. It is depicted that the overall performance is lower than that of the proposed method with Gini impurity index, entropy index and their combination.

Table 6
Results of proposed method using *k*-means

Cluster	Acc [%]	Sen [%]	Spe [%]	Time [h]
NSL-KDD				
5	94.99	95.89	94.20	0.51
705	97.44	97.54	97.33	0.27
Kyoto 2006				
5	97.70	97.49	97.91	8.84
200	98.39	97.83	98.93	2.01

5.6. Correlation between the Number of Clusters and Performance

Correlation between the number of clusters and performance (accuracy, sensitivity, and specificity) on NSL-KDD and Kyoto 2006 datasets can be observed in Figs. 7–12. In general, it is shown that the number of clusters is proportional to the performance. In more details, NSL-KDD produces an exponential graph for accuracy and sensitivity, and linear for specificity, as depicted in Figs. 7, 8 and 9, respectively. For accuracy and sensitivity, the number of 4 subcentroids is better than others when the number of clusters is less than 600, and 5 subcentroids is for more than 600 clusters. Differently, 6 subcentroids is the best for almost all numbers of clusters.

Slightly different from the experiment on NSL-KDD, that on Kyoto 2006 generates logarithmic graphs, as presented in Figs. 10–12. It also shows that increasing the number of clusters means raising the performance, for the same number of subcentroids.

Nevertheless, there is a drawback of this increasing number of clusters. That is, more numbers of clusters need more time to transform the data. This is because each distance between the data and centroids must be calculated. An advantage of this condition is that the time to classify the data drops since the size of each cluster is very likely to be smaller.

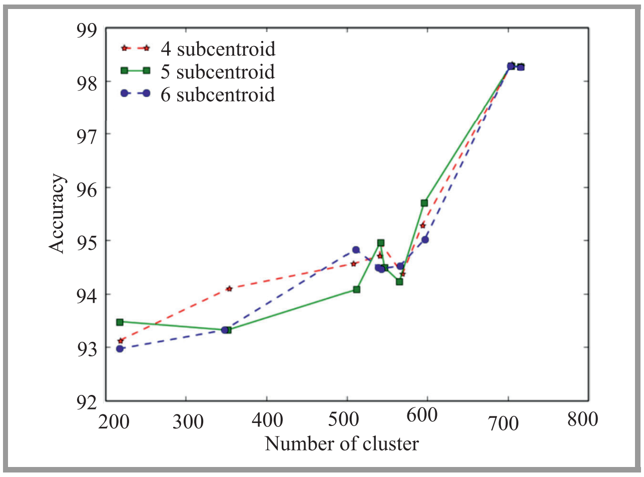

Fig. 7. Correlation between number of cluster and accuracy on NSL-KDD.

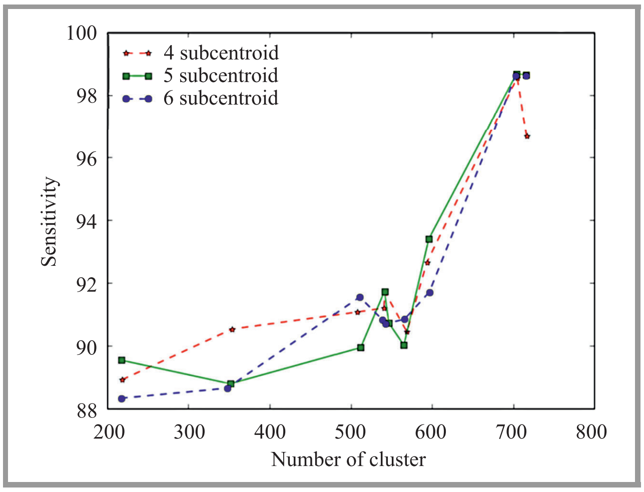

Fig. 8. Correlation between number of cluster and sensitivity on NSL-KDD.

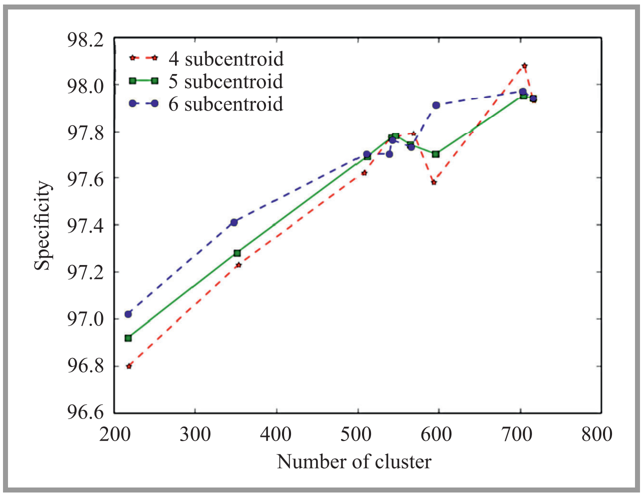

Fig. 9. Correlation between number of cluster and specificity on NSL-KDD.

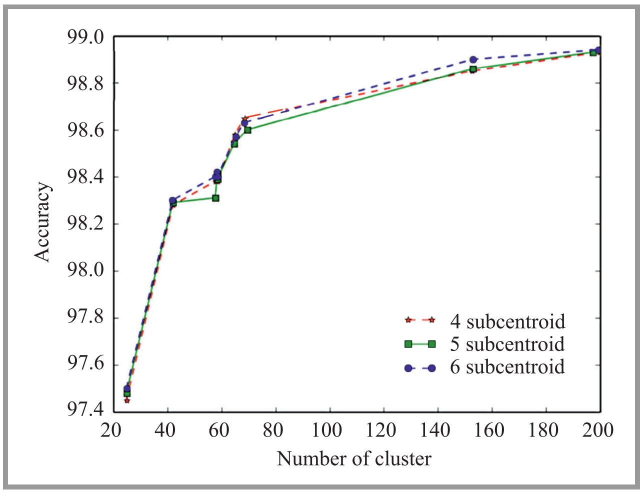

Fig. 10. Correlation between number of cluster and accuracy on Kyoto 2006.

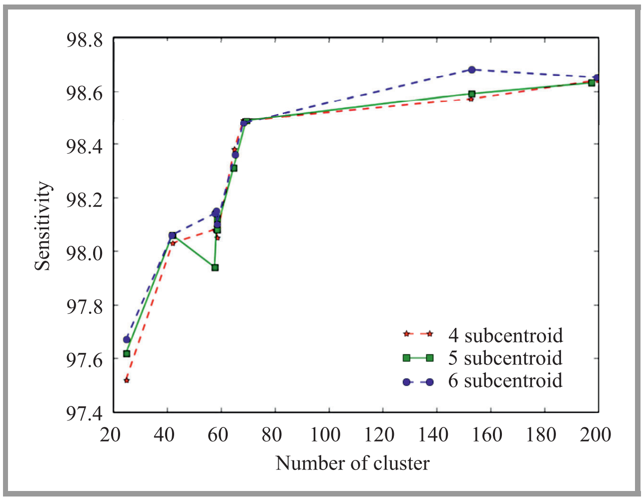

Fig. 11. Correlation between number of cluster and sensitivity on Kyoto 2006.

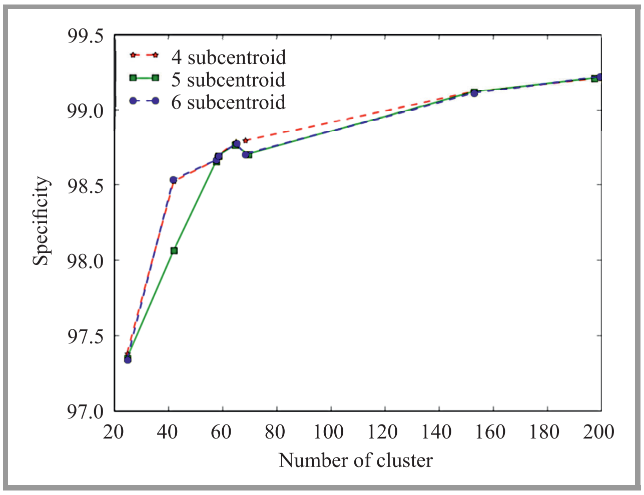

Fig. 12. Correlation between number of cluster and specificity on Kyoto 2006.

5.7. Classification Results

The classification results of our proposed method using combined index = 0.1 and the number of subcentroids = 4 is presented in Table 7. These parameter values are selected because we believe that this combination delivers better results than others, as described in previous tables. In addition, the results of the proposed method with *k*-means whose number of clusters is 5 and 705 as previously specified are in Tables 8 and 9.

Table 7
Classification results of proposed method using combined impurity index 0.1 and subcentroid 4

Actual		Detected				
		Attack				Normal
		Dos	Probe	U2R	R2L	
Attack	Dos	8823	350	0	0	61
	Probe	302	1907	0	1	79
	U2R	0	1	0	2	8
	R2L	0	0	0	189	20
Normal		60	134	1	61	13192

Table 8
Classification results of proposed method with 5 clusters

Actual		Detected				
		Attack				Normal
		Dos	Probe	U2R	R2L	
Attack	Dos	8649	190	0	0	302
	Probe	352	1780	0	0	123
	U2R	0	1	0	0	10
	R2L	3	2	0	155	47
Normal		568	149	0	43	12669

Table 9
Classification results of proposed method
with 705 clusters

Actual		Detected				
		Attack				Normal
		Dos	Probe	U2R	R2L	
Attack	Dos	8933	191	0	0	95
	Probe	305	1799	0	2	159
	U2R	0	0	2	0	9
	R2L	0	1	0	182	23
Normal		120	193	3	19	13091

The result shows that the U2R class can only be classified by using *k*-means with 705 clusters. Here, 2 out of 11 have been correctly classified. The classification result of probe and normal classes, however, drops significantly. This hard-classification of U2R class may happen because their coordinate in the cluster is sparsely distributed. Therefore, a relatively high number of clusters are required in order to be able to classify the data correctly.

6. Conclusion

In this paper we have investigated the classification method of data in an IDS. We propose the use of impurity indexes such as the entropy. In addition, we also examine the effect of bisecting *k*-means and logarithmic distance. Based on the experimental results, the use of those parameters and methods gives better performance in terms of accuracy, sensitivity, and specificity than TANN. Overall, we also find that there is correlation between the number of clusters and the performance. That is, more numbers of clusters may result to higher performance.

In the future, we would like to do further research in order to increase the performance. This may be done by normalizing the data in the preprocessing step. Additionally, the parameter for deciding whether a cluster must be split or not, is to be refined.

References

[1] B. Czaplewski, M. Dzwonkowski, and R. Rykaczewski, "Digital fingerprinting based on quaternion encryption scheme for gray-tone images", *J. Telecommun. & Inform. Technol.*, no. 2, pp. 3–11, 2014.

[2] T. Ahmad and J. Hu, "Generating cancelable biometric templates using a projection line", in *Proc. 11th Int. Conf. on Control, Autom., Robot. & Vision ICARCV 2010*, Singapore, 2010, pp. 7–12.

[3] M. Holil and T. Ahmad, "Secret data hiding by optimizing general smoothness difference expansion-based method", *J. Theor. & Appl. Informa. Technol.*, vol. 72, no. 2, pp. 155–163, 2015.

[4] P. García-Teodoro, J. Díaz-Verdejo, G. Macia-Fernández, and É. Vazquez, "Anomaly-based network intrusion detection: Techniques, systems and challenges", *Comput. Secur.*, vol. 28, no. 1–2, pp. 18–28, 2009.

[5] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection", in *Proc. IEEE Symp. on Secur. & Priv.*, Oakland, CA, USA, 2010, pp. 305–316.

[6] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd ed. San Francisco, CA, USA: Morgan Kaufmann Publ. Inc., 2005.

[7] W.-C. Lin, S.-W. Ke, and C.-F. Tsai, "CANN", *Know.-Based Syst.*, vol. 78, no. C, pp. 13–21, 2015.

[8] C.-F. Tsai and C.-Y. Lin, "A triangle area based nearest neighbors approach to intrusion detection", *Pattern Recogn.*, vol. 43, no. 1, pp. 222–229, 2010.

[9] K. Muchammad and T. Ahmad, "Detecting intrusion using recursive clustering and sum of log distance to subcentroid", *Procedia Comp. Sci.*, vol. 72, no. 1, pp. 446–452, 2015.

[10] C. Guo, Y. Zhou, Y. Ping, Z. Zhang, G. Liu, and Y. Yang, "A distance sum-based hybrid method for intrusion detection", *Appl. Intellig.*, vol. 40, no. 1, pp. 178–188, 2014.

[11] J. Parkinson and M. Blaxter, "Simitri-visualizing similarity relationships for groups of sequences", *Bioinformatics*, vol. 19, no. 3, pp. 390–395, Feb. 2003.

[12] B. Luo and J. Xia, "A novel intrusion detection system based on feature generation with visualization strategy", *Expert Syst. Appl.*, vol. 41, no. 9, pp. 4139–4147, 2014.

[13] E.-H. Han and G. Karypis, "Centroid-based document classification: Analysis and experimental results", in *Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery*. London, UK: Springer, 2000, pp. 424–431.

[14] S. Varuna and P. Natesan, "An integration of *k*-means clustering and nave bayes classifier for intrusion detection", in *Proc. 3rd Int. Conf. on Sig. Process., Commun. & Netw. ICSCN 2015*, Chennai, India, 2015, pp. 1–5.

[15] W. Wang, T. Guyet, R. Quiniou, M.-O. Cordier, F. Masseglia, and X. Zhang, "Autonomic intrusion detection: Adaptively detecting anomalies over unlabeled audit data streams in computer networks", *Knowl.-Based Syst.*, vol. 2014, no. 70, 2014.

[16] X. Zhang, C. Furtlehner, C. Germain-Renaud, and M. Sebag, "Data stream clustering with affinity propagation", *IEEE Trans. on Knowl. & Data Engin.*, vol. 26, no. 7, pp. 1644–1656, 2014.

[17] B. J. Frey and D. Dueck, "Clustering by passing messages between data points", *Science*, vol. 315, no. 5814, pp. 972–976, 2007.

[18] V. Garcia, E. Debreuve, and M. Barlaud,"Fast k-Nearest Neighbor Search using GPU", ArXiv e-prints, Apr. 2008.

[19] J. Heinermann, O. Kramer, K. L. Polsterer, and F. Gieseke, "On GPU based nearest neighbor queries for large-scale photometric catalogs in astronomy", in *KI 2013: Advances in Artificial Intelligence*, I. J. Timm and M. Thimm, Eds. *LNCS*, vol. 8077, pp. 86–97. Springer, 2013.

[20] M. Steinbach, G. Karypis, and V. Kumar, "A comparison of document clustering techniques", in *Proc. KDD Workshop on Text Mining*, Boston, MA, USA, 2000.

[21] A. Demiriz, K. Bennett, and M. J. Embrechts, "Semi-supervised clustering using genetic algorithms", in *Proc. Conf. on Artificial Neural Networks in Engineering ANNIE'99*, St. Louis, MO, USA, 1999, pp. 809–814.

[22] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD cup 99 data set", in *Proc. 2nd IEEE Symp. on Computat. Intellig. for Secur. & Defense Appl. CISDA'09*, Ottawa, Canada, 2009, pp. 53–58.

[23] J. Song, H. Takakura, Y. Okabe, M. Eto, D. Inoue, and K. Nakao, "Statistical analysis of honeypot data and building of Kyoto 2006+ dataset for NIDS evaluation", in *Proc. 1st Worksh. on Build. Analysis Datasets and Gathering Exper. Returns for Secur. BADGERS 2011*, Salzburg, Austria, 2011, pp. 29–36.

Tohari Ahmad is a lecturer at Institut Teknologi Sepuluh Nopember (ITS) Surabaya, Indonesia. He received his Bachelor, Master and Doctorate degrees from ITS, Monash University (Australia) and RMIT University (Australia), respectively. His research interest is in network security, data hiding and pattern recognition.

E-mail: tohari@if.its.ac.id
Department of Informatics
Institut Teknologi Sepuluh Nopember (ITS)
Surabaya, Indonesia

Kharisma Muchammad received the Bachelor and Master of Computer from Institut Teknologi Sepuluh Nopember (ITS), Surabaya, Indonesia in 2013 and 2016, respectively. His research interest is in applied machine learning, computer security and data hiding. Currently, he is also a software developer in an IT industry.

E-mail: kharisma_muchammad@yahoo.co.id
Department of Informatics
Institut Teknologi Sepuluh Nopember (ITS)
Surabaya, Indonesia