An Effective Unsupervised Network Anomaly Detection Method

Monowar H. Bhuyan
Dept. of Computer Science and Engineering
Tezpur University
Tezpur, Assam, India
mhb@tezu.ernet.in

D. K. Bhattacharyya
Dept. of Computer Science and Engineering
Tezpur University
Tezpur, Assam, India
dkb@tezu.ernet.in

J. K. Kalita
Dept. of Computer Science
University of Colorado at Colorado Springs
CO 80918, USA
jkalita@uccs.edu

ABSTRACT

In this paper, we present an effective tree based subspace clustering technique (TreeCLUS) for finding clusters in network intrusion data and for detecting unknown attacks without using any labelled traffic or signatures or training. To establish its effectiveness in finding all possible clusters, we perform a cluster stability analysis. We also introduce an effective cluster labelling technique (CLUSLab) to generate labelled dataset based on the stable cluster set generated by TreeCLUS. CLUSLab is a multi-objective technique that exploits an ensemble approach for stability analysis of the clusters generated by TreeCLUS. We evaluate the performance of both TreeCLUS and CLUSLab in terms of several real world intrusion datasets to identify unknown attacks and find that both outperform the competing algorithms.

Categories and Subject Descriptors
K.6.5 [Management of Computing and Information Systems]: Security and Protection—Authentication, Invasive software and unauthorized access

General Terms
Algorithm, Cluster, Anomaly

Keywords
Cluster, unsupervised, intrusion, cluster stability, ensemble

1. INTRODUCTION

Advances in networking technology have enabled us to connect distant corners of the globe through the Internet for sharing vast amounts of information. However, along with this advancement, the threat from spammers, attackers and criminal enterprises is also growing in multiple speed [1]. As a result, security experts use intrusion detection technology to keep secure large enterprise infrastructures. Intrusion detection systems (IDSs) are divided into two broad categories: misuse detection [2] and anomaly detection [3] systems. Misuse detection can detect only known attacks based on available signatures. Thus, dynamic signature updation is important and new attack definitions are frequently released by the IDS vendors. However, misuse based systems cannot incorporate the rapidly growing number of vulnerabilities and exploits. On the other hand, anomaly based detection systems are designed to capture any deviation from profiles of normal behaviour. They are more suitable than misuse detection for detecting unknown or novel attacks without any prior knowledge. But normally they generate large numbers of false alarms.

There are three commonly used approaches for detecting intrusions [4]: (i) supervised (i.e., both normal and attack instances are used for training), (ii) semi-supervised (i.e., only normal instances are used for training) and (iii) unsupervised (i.e., without using any prior knowledge). The first two cases require training on the instances for finding anomalies. But getting large amounts of labelled normal and attack training instances may not be feasible for a particular scenario. In addition, generating a set of true normal instances with all the variations is an extremely difficult task. Hence, unsupervised network anomaly detection which does not require any prior knowledge about the network traffic instances, plays important role in this situation.

We aim to provide an unsupervised solution for identifying network attacks with high detection rate. The main contributions of this paper are: (i) a tree based clustering technique (TreeCLUS) to identify network anomalies in high dimensional datasets, (ii) a cluster stability analysis to obtain a stable set of results generated by TreeCLUS and (iii) a cluster labelling technique (CLUSLab) for labelling the clusters generated by TreeCLUS as normal or attack. It exploits a majority voting based decision fusion technique of the results of various cluster indices.

2. CLUSTERING IN UNSUPERVISED NETWORK ANOMALY DETECTION: A REVIEW

The problem of unsupervised detection of network attacks and intrusions has been studied for several years with the goal of identifying unknown attacks in high speed network traffic data. Most network based intrusion detection systems (NIDSs) are misuse or signature based. For example, SNORT [5] and BRO [6] are the two well-known open source misuse based NIDS. To overcome the inability of such systems to detect unknown attacks, several novel anomaly based NIDSs [7] have been introduced in the past decade. A concise review of these unsupervised network anomaly detection techniques is given next.
2.1 Clustering based network anomaly detection

Clustering is an important technique used in unsupervised network intrusion detection. Majority of unsupervised network anomaly detection techniques are based on clustering and outliers detection [8, 9, 10]. Leung and Leckie [9] report a grid based clustering algorithm to achieve reduced computational complexity. An unsupervised intrusion detection method by computing cluster radius threshold (CBUID) is proposed by [11]. The authors claim that CBUID works in linear time with the size of datasets and the number of features. Song et al. [12] report an unsupervised autotuned clustering approach that optimizes parameters and detects changes based unsupervised anomaly detection for identifying unknown attacks. Finally, Casas et al. [13] present a novel unsupervised outlier detection approach based on combining subspace clustering and multiple evidence accumulation to detect various kinds of intrusions. They evaluate the method using KDDcup99 and other two real time datasets.

2.2 Cluster stability analysis

There are several cluster stability analysis techniques available in the literature [14, 15, 16]. We analyze cluster stability for identifying the actual number of clusters generated by our clustering algorithm using stability calculation. Lange et al. [14] introduce a cluster stability measure to validate clustering results. Ben-David et al. [15] provide a formal definition of cluster stability with specific properties. They conclude that stability can be determined based on the behavior of the objective function. If the objective function is unique global optimizer, the algorithm is stable. Das and Sii [16] also present a cluster validation method for stable cluster generation using stability analysis.

2.3 Cluster labelling

Cluster labelling is a challenging issue in unsupervised network anomaly detection. Most common cluster validity measures are summarized in [17, 18]. Validity measures are usually based on internal and external properties of clustering results. Normally, internal validity measures derive the compactness, connectedness and separation of the cluster partitions. The external validity measures assess agreement between a new clustering solution and the reference clusters. Jun [18] presents an ensemble method for cluster analysis. It uses a simple voting mechanism for making decision from the results obtained by using several cluster validity measures. Labelling of a cluster is must in case of cluster based unsupervised network anomaly detection. Our proposed cluster labelling technique works based on the cluster size, compactness and the dominating feature set.

2.4 Discussion

Based on our review, following observations are made.

- Among existing only a few clustering methods have full features of an unsupervised intrusion detection system.
- Existing stability analysis techniques are mostly applied to analyze normal data clusters.
- An appropriate use of indices can help in developing an effective labelling technique, which can support the unsupervised anomaly detection process to a great extent.

3. UNSUPERVISED NETWORK ANOMALY DETECTION : THE FRAMEWORK

The main aim of this work is to detect network anomalies using an unsupervised approach with minimum false alarms. First, we introduce a tree based subspace clustering technique (TreeCLUS) for generating clusters in high dimensional large datasets. TreeCLUS exploits the MMIFS technique [19] for finding a highly relevant feature set. Second, we analyze the stability of the cluster results obtained. Third, we propose a cluster labelling technique (CLUSLab) to label the stable clusters using a multi-objective approach. The framework for unsupervised network anomaly detection is shown in Figure 1, where \( C_1, C_2, \ldots C_k \) represent the clusters obtained from TreeCLUS based on the subset of features selected using the MMIFS technique [19]. Cluster stability analysis is performed based on an ensemble of several index measures such as Dunn index (Dunn) [20], C-index (C) [21], Davies Bouldin index (DB) [22], Silhouette index (S) [23] and Xie-Beni index (XB) [24].

![Figure 1: High level description of unsupervised cluster formation](image)

The symbols used to describe the unsupervised network anomaly detection method are given in Table 1.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>( D )</td>
<td>dataset</td>
</tr>
<tr>
<td>( D_s )</td>
<td>sample dataset</td>
</tr>
<tr>
<td>( n )</td>
<td>number of data objects in ( x )</td>
</tr>
<tr>
<td>( C )</td>
<td>set of clusters</td>
</tr>
<tr>
<td>( f_s )</td>
<td>sample feature set</td>
</tr>
<tr>
<td>( sim )</td>
<td>proximity measure between two objects ( O_i ) and ( O_j )</td>
</tr>
<tr>
<td>( \alpha )</td>
<td>threshold for ( L_1 ) cluster</td>
</tr>
<tr>
<td>( \beta )</td>
<td>threshold for ( L_2 ) cluster</td>
</tr>
<tr>
<td>( \xi )</td>
<td>a factor for step down ratio</td>
</tr>
<tr>
<td>( k )</td>
<td>number of clusters</td>
</tr>
<tr>
<td>( l )</td>
<td>level</td>
</tr>
<tr>
<td>( x_i )</td>
<td>( i^{th} ) data object</td>
</tr>
<tr>
<td>( \theta )</td>
<td>height of the tree</td>
</tr>
<tr>
<td>( n_f )</td>
<td>total number relevant features</td>
</tr>
<tr>
<td>( minRank_f )</td>
<td>minimum rank value found w.r.t. MMIFS algorithm [19]</td>
</tr>
<tr>
<td>( N_i )</td>
<td>( i^{th} ) node in tree</td>
</tr>
<tr>
<td>( CL )</td>
<td>class label</td>
</tr>
</tbody>
</table>
3.1 TreeCLUS: the clustering technique

TreeCLUS is a tree based subspace clustering technique for high dimensional data. It is especially tuned for unsupervised network anomaly detection. It exploits the MMIFS technique [19] to identify a subset of relevant features. TreeCLUS depends on two parameters, viz., initial node formation threshold (α) and a factor for step down ratio (ε) to extend the initial node depth-wise. Both these parameters are computed using a heuristic approach. Next we present definitions and lemmas which help to describe the TreeCLUS algorithm.

Definition 1. Data Stream: A data stream D is denoted as \{O_1,O_2,\ldots,O_n\} with n objects, where O_i is the i\textsuperscript{th} object with d-dimensional feature subset, i.e., \(O_i = \{x_{i1},x_{i2},x_{i3}\ldots x_{id}\}\).

Definition 2. Neighbour of an object: An object \(O_i\) is a neighbor of \(O_j\), w.r.t. a threshold \(\alpha\), if \(\text{sim}(O_i,O_j) \leq \alpha\), where \(\text{sim}\) is a proximity measure and \(\beta\) is a subset of relevant features.

Definition 3. Connected objects: If object \(O_i\) is a neighbor of \(O_j\) and \(O_j\) is a neighbor of \(O_k\), w.r.t. \(\alpha\), then \(O_i, O_j, O_k\) are connected.

Definition 4. Node: A node \(N_i\) in the i\textsuperscript{th} level of a tree is a non-empty subset of objects \(x^i\), where for any object \(O_i \in N_i\), there must be another object \(O_j \in x^i\), which is a neighbor of \(O_i\) and \(O_j\) is either (a) itself an initiator object or (b) is within the neighborhood of an initiator object \(O_j \in N_i\).

Definition 5. Degree of a node: Degree of a node \(N_j\) w.r.t. \(\alpha\) is defined as the number of objects in \(N_j\) that are within \(\alpha\)-neighborhood of any object \(O_j \in N_j\).

Definition 6. \(L_{1,\alpha}^i\) cluster: It is a set of connected objects \(C_i\) at level 1 w.r.t. \(\alpha\), where for any two objects \(O_i, O_j \in C_i\), the neighborhood condition (Definition 2) is true with reference to \(f_i\).

Definition 7. \(L_{2,\beta}^i\) cluster: It is a set of connected objects \(C_j\) at level 2 w.r.t. \(\beta\), where for any two objects \(O_i, O_j \in C_j\), the neighborhood condition (Definition 2) is true with reference to \(f_i\) and \(\beta \leq (\frac{\alpha}{2} + \varepsilon)\). Also, \(L_{2,\beta}^i \subseteq L_{1,\alpha}^i\).

Definition 8. Outlier: An object \(O_i \in D\) is an outlier if \(O_i\) is not connected with any other object \(O_j \in D\), where \(O_j \in L_{1,\alpha}^i\). In other words, \(O_i\) is an outlier if there is no \(O_j \in D\), so that \(O_i\) and \(O_j\) are neighbors (as per Definition 2).

Lemma 1. Two objects \(O_i\) and \(O_j\) belonging to two different nodes are not similar.

Proof. Let \(O_i \in N_i, O_j \in N_j\) and \(O_i\) is a neighbor of \(O_j\). According to Definition 2 and Definition 4, \(O_i\) and \(O_j\) should belong to same node. Therefore, we come to a contradiction and hence the proof.

The algorithm is illustrated using an example. Suppose \(D\) is a dataset of \(d\) dimensions and \(D_s\) represents a sample dataset taken from \(D\), given in Table 2.

Here, let \(D_s = \{O_1, O_2,\ldots,O_{16}\}\) and \(f_s = \{f_1, f_2,\ldots,f_{10}\}\). The extracted relevant feature set is given in Table 3. The class specific relevant features are identified from \(D_s\) w.r.t. a threshold \(\gamma\). We achieved best results while \(\gamma \geq 1\) for class \(C_1\), \(\gamma \geq 0.918\) for class \(C_2\) and \(\gamma \geq 0.917\) for class \(C_3\) as shown in Table 3. TreeCLUS starts by creating a tree structure in a depth first manner with an empty root node.

The root is at level 0 and is connected to all the nodes in level 1. The nodes in level 1 are created based on maximal subset of relevant features by computing proximity within a neighborhood w.r.t. an initial cluster formation threshold \(\alpha\). A tree is extended in depth first manner by forming lower level nodes w.r.t. \((\frac{\alpha}{2} + \varepsilon)\), where \(\varepsilon\) is a controlling parameter of the step down factor i.e., \(\frac{\alpha}{2}\). \(\alpha\) and \(\varepsilon\) are computed using a heuristic approach. A proximity measure, \(\text{sim}\) is used in TreeCLUS during cluster formation. \(\text{sim}\) is free from the restriction of using a specific proximity measure. The tree generated from \(D_s\) is given in Figure 2 based on Euclidean distance as the proximity measure.

![Figure 2: Tree obtained from \(D_s\)](image)

3.2 Cluster Stability Analysis

We analyze the stability of clusters obtained from TreeCLUS and other clustering algorithms such as k-means, fuzzy c-means, and hierarchical clustering. An ensemble based cluster stability analysis technique is proposed based on Dunn index [20], C-index [21], Davies Bouldin index (DB) [22], Silhouette index (S) [23] and Xie-Beni index (XB) [24] (shown in Figure 1). We discuss briefly each of them.

(a) Dunn index [20] is computed for finding compact and well separated clusters. It depends on \(d_{min}\) and \(d_{max}\), where \(d_{min}\) denotes the smallest distance between two objects from different clusters and \(d_{max}\) is the largest distance between two elements within the same cluster. The Dunn index is \(\frac{d_{min}}{d_{max}}\). Clearly, Dunn \(\in [0,\infty]\). Larger values of Dunn index indicates better clustering.

(b) The C-index [21] is used to find cluster quality when the clusters are of similar sizes. The C-index is defined as \((\frac{S-S_{min}}{S_{max}-S_{min}})\), where \(S\) is the sum of distances over all pairs of objects form the same cluster, \(n\) is the number of such pairs, \(S_{min}\) and \(S_{max}\) are the sum of \(n\) smallest distances and \(n\) largest distances, respectively. \(C \in [0,1]\). Smaller values of \(C\) state better clusters.

(c) The Davies Bouldin index [22] is a ratio of the sum of within-cluster to between-cluster separation. The Davies Bouldin index is \((\frac{1}{c} \sum_{i=1}^{n} \max \{\frac{d(c_i,c_j)}{{\sigma_i} + {\sigma_j}}\})\), where \(n\) is the number of clusters; \(\sigma_i\) is the average distance of all patterns in cluster \(i\) to their cluster center, \(c_i\); \(\sigma_j\) is the average distance of all patterns in cluster \(j\) to their cluster center, \(c_j\); and \(d(c_i,c_j)\) represents the proximity between the cluster centers \(c_i\) and \(c_j\). Lower value of DB indicates better clusters.

(d) The Silhouette index [23] is computed for a cluster to identify tightly separated groups. The Silhouette index is defined as \((\frac{b_i-a_i}{\max(a_i,b_i)})\), where \(a_i\) is the average dissimilarity of the \(i\textsuperscript{th}\) object to all other objects in
the same cluster; \( b_i \) is the minimum of average dissimilarity of the \( i^{th} \) object to all objects in other clusters.

(e) The Xie-Beni index [24] is defined as \( \frac{\pi}{n} \), where \( \pi = \frac{1}{n} \) is called compactness of cluster \( i \). Since \( n_i \) is the number of points in cluster \( i \), \( \sigma \) is the average variation in cluster \( i \); \( d_{min} = \min \{ \lvert k_i - k_j \rvert \} \). Smaller values of \( XB \) are expected for compact and well-separated clusters.

We summarize the criteria for cluster stability analysis in Table 4.

Table 4: Criteria of cluster stability analysis

<table>
<thead>
<tr>
<th>Stability measures</th>
<th>Range of value</th>
<th>Criteria for better clusters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dunn index</td>
<td>(0, \infty)</td>
<td>Maximized</td>
</tr>
<tr>
<td>C-index</td>
<td>(0, 1)</td>
<td>Minimized</td>
</tr>
<tr>
<td>Davies Bouldin index</td>
<td>(0, \infty)</td>
<td>Minimized</td>
</tr>
<tr>
<td>Silhouette index</td>
<td>(−1, 1)</td>
<td>Near 1</td>
</tr>
<tr>
<td>Xie Beni index</td>
<td>(0, 1)</td>
<td>Minimized</td>
</tr>
</tbody>
</table>

The criteria for cluster stability analysis used is given in Table 4. We pass each cluster \( C_i \) to a function StableCLUS to identify stability. It computes all the indices for each of the clusters \( C_1, C_2, \ldots, C_k \). If it finds better result for an index then it stores it as 1, otherwise assign 0. The way it computes 1 or 0 for each of the indexes is given below. Here \( \sigma \) and \( \tau \) are threshold parameters.

\[
V_i = \begin{cases} 
1, & I_i \geq \sigma \text{ or } I_i \leq \tau \\
0, & \text{otherwise}
\end{cases}
\]

Finally, we take maximum occurrences of 1 for making decision as stable or not. If a cluster \( C_i \) is not stable, it sends control back to TreeCLUS to regenerate another set with a different number of clusters.

3.3 CLUSLab: cluster labelling technique

CLUSLab is a multi-objective cluster labelling technique for labelling the clusters generated by TreeCLUS. It decides the label of the instances of a cluster based on three measures: (i) cluster size, (ii) compactness and (iii) dominating feature subset. A high level description of this technique is given in Figure 3. The measures are described in brief next.

![Figure 3: High level description of multi-objective cluster labelling](image)

(i) Cluster size: It is the number of instances in a cluster.

(ii) Compactness: To find the compactness of a cluster \( C_i \), obtained from TreeCLUS, we use the five very well
Algorithm 1: Part 1 TreeCLUS (D, α, β)
\textbf{Input:} D represents the dataset; α, threshold for \(L_1\) cluster formation; β, threshold for \(L_2\) cluster formation;
\textbf{Output:} generate clusters, \(C_1, C_2, \ldots, C_k\)
1: initialization: node\_id ← 0
2: function BUILDTree(D, node\_id)
3:  for i ← 1 to D do
4:      if \((D_i, \text{classified}) = 1\) and \(\text{check\_ini\_feat(MMIFS(D_i))} == \text{true}\) and \(\text{sim ≤ α}\)
6:         then
7:            CreateNode(D_i, no, p_id, temp, node\_count, node\_id, l)
8:            while \((n_j - (1 - 1)) ≥ \theta\) do
9:               l++
10:          for i ← 1 to D do
11:             if \((D_i, \text{classified}) = 1\) then
12:                p_id = check\_parent(D_i, no, l)
13:           if \((p_id > -1)\) and \(\text{check\_ini\_feat(MMIFS(D_i))} == \text{true}\)
14:              then
15:                 CreateNode(D_i, no, p_id, temp, node\_count, node\_id, l)
16:            end if
17:        end for
18:    end while
19: end for
20: end function
21: function CreateNode(no, p_id, temp, node\_count, id, l)
22:    node\_id = new node()
23:    node\_id.temp = temp
24:    node\_id.node\_count = node\_count
25:    node\_id.p_id = p_id
26:    node\_id.id = id;
27:    node\_id.level = l
28:    ExpandNode(no, id, node\_id.temp, node\_count, l)
29:    temp = NULL;
30:    node\_count = 0;
31:    node\_id++
32: end function
33: function ExpandNode(no, id, node\_id, temp, node\_count, l)
34:    if \(D_{no}.\text{classified} = 1\) then
35:        return
36:    else
37:        \(D_{no}.\text{classified} = 1\);
38:        \(D_{no}.\text{node\_id} = \text{id}\)
39:    for i ← 1 to D do
40:        if \((D_i, \text{classified}) = 1\) then
41:            \(\text{minRank}_j = \text{find\_minRank(MMIFS(D_i))}\)
42:        if \((n_j - \text{minRank}_j) ≥ \theta\)
43:            then
44:                \(\text{minRank}_j++\) until get a specific cluster; otherwise stop.
45:        ExpandNode(D_i, no, id, temp, node\_count, l)
46:    end if
47:    end for
48: end function
49: function STABLECLUS(C_k)
50:  for i ← 1 to \(k\) do
51:      for j ← 1 to 5 do
52:         \(\text{known indices as given in Table 4 and discussed earlier.}\)
53: end function

Algorithm 2: Part 2 TreeCLUS (D, α, β)
53: \(V_L[j] = \text{compute}(\text{I}_c)\)
54: if \((V_L[j] ≥ \sigma \text{ or } V_L[j] ≤ \tau)\) then
55: \(V_L[j] = 1\)
56: else
57: \(V_L[j] = 0\)
58: end if
59: end for
60: if \((C_i = \text{Max}(V_L[i]))\) then
61: \(\text{stable cluster, } C_i\)
62: end else
63: go to step 2
64: end if
65: end for
66: end function

3.4 Complexity analysis
We compute the time complexity of the TreeCLUS clustering technique for unsupervised network anomaly detection. We assume that a total \(k\) number of clusters are obtained from \(n\) number of data objects based on TreeCLUS. Initial cluster formation requires \((n-1)\) comparisons each. Hence, the complexity of TreeCLUS is \(O(kn^2) + O(n)\), where \(k = \log n\). The complexity for stability analysis. For cluster validity measures, it requires \(O(1)\) time for computing cluster size, \(O(n)\) time for each cluster validity measures, i.e., \(n + n + n + n + n = 5n \approx O(n)\) and \(O(f - z)\) time for computing dominating feature subset, where \(f\) is the total number of features and \(z\) is the number of features reduced. Hence, the total computational complexity of CLUSLab is \(O(n) + O(f - z)\).

The time complexity for each stage of our proposed unsupervised network anomaly detection method is linear w.r.t. the size of dataset, number of features, number of clusters and labelling each clusters. This implies good scalability of the proposed method.

4. EXPERIMENTAL ANALYSIS
In this section, we present experimental analysis and results of the proposed unsupervised network anomaly detection method using several real world datasets. We have implemented our TreeCLUS algorithm using Java on an HP xw6600 workstation containing Intel Xeon Processor (3.00 Ghz) with 4GB RAM in Linux (Ubuntu 10.10) platform. The datasets used in this paper to evaluate the proposed method and experimental results are discussed below.

4.1 Datasets used
In this paper, we use three datasets for evaluation of the proposed method, viz., (i) UCI ML repository datasets, (ii) TUIDS datasets, and (iii) KDDcup99 datasets. We use four UCI machine learning repository datasets [25]: Zoo, Glass, Abalone, and Shuttle to initially validate clusters generated by TreeCLUS. Table 5 describes the details of UCI datasets and their characteristics. To establish the effectiveness of
TreeCLUS when using intrusion datasets, we used TUIDS [26] and KDDcup99 [27] intrusion datasets. Details of both these datasets are given in Table 6.

**Table 5: Characteristics of real-life UCI datasets**

<table>
<thead>
<tr>
<th>Sl No.</th>
<th>Datasets</th>
<th>Dimension</th>
<th>No. of instances</th>
<th>No. of classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Zoo</td>
<td>18</td>
<td>101</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>Glass</td>
<td>10</td>
<td>214</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>Abalone</td>
<td>8</td>
<td>4177</td>
<td>29</td>
</tr>
<tr>
<td>4</td>
<td>Shuttle</td>
<td>9</td>
<td>14500</td>
<td>3</td>
</tr>
</tbody>
</table>

**Table 6: Distribution of Normal and Attack connections instances in both real time packet, flow level TUIDS and KDDcup99 intrusion datasets**

<table>
<thead>
<tr>
<th>Connection type</th>
<th>Dimensions</th>
<th>No. of instances</th>
<th>No. of classes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TUIDS packet level</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Normal</td>
<td>50</td>
<td>47895</td>
<td>1</td>
</tr>
<tr>
<td>DoS</td>
<td></td>
<td>30613</td>
<td>15</td>
</tr>
<tr>
<td>Probe</td>
<td></td>
<td>7757</td>
<td>5</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>86265</td>
<td>21</td>
</tr>
<tr>
<td><strong>TUIDS flow level</strong></td>
<td></td>
<td>16770</td>
<td>1</td>
</tr>
<tr>
<td>DoS</td>
<td>25</td>
<td>14475</td>
<td>15</td>
</tr>
<tr>
<td>Probe</td>
<td></td>
<td>9480</td>
<td>5</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>40725</td>
<td>21</td>
</tr>
<tr>
<td><strong>KDDcup99 corrected</strong></td>
<td></td>
<td>60593</td>
<td>1</td>
</tr>
<tr>
<td>Normal</td>
<td>41</td>
<td>228953</td>
<td>12</td>
</tr>
<tr>
<td>DoS</td>
<td></td>
<td>4166</td>
<td>6</td>
</tr>
<tr>
<td>Probe</td>
<td></td>
<td>16189</td>
<td>12</td>
</tr>
<tr>
<td>U2R</td>
<td></td>
<td>228</td>
<td>6</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>311029</td>
<td>37</td>
</tr>
</tbody>
</table>

4.2 Results

In this section, we report the performance of the proposed method using various real-life and benchmark datasets.

4.2.1 UCI ML repository datasets

The proposed method was initially tested using UCI ML repository datasets. We assume that larger clusters are normal and smaller clusters are anomalous. Based on our CLUSLab cluster labelling technique, we label each cluster obtained from the TreeCLUS. We compare the performance in terms of detection rate (DR) and false positive rate (FPR). The detailed results are given in Table 7. The proposed method has better performance than several other existing algorithms that we compared with.

**Table 7: Experimental results on UCI ML repository datasets**

<table>
<thead>
<tr>
<th>Dataset</th>
<th>No. of clusters</th>
<th>Correctly detected</th>
<th>Miss detected</th>
<th>Detection rate (%)</th>
<th>False positive rate (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zoo</td>
<td>8</td>
<td>95</td>
<td>6</td>
<td>0.9406</td>
<td>0.0594</td>
</tr>
<tr>
<td>Glass</td>
<td>20</td>
<td>206</td>
<td>8</td>
<td>0.9626</td>
<td>0.0373</td>
</tr>
<tr>
<td>Abalone</td>
<td>22</td>
<td>4002</td>
<td>175</td>
<td>0.9581</td>
<td>0.0418</td>
</tr>
<tr>
<td>Shuttle</td>
<td>3</td>
<td>14256</td>
<td>204</td>
<td>0.9850</td>
<td>0.0141</td>
</tr>
</tbody>
</table>

4.2.2 TUIDS and KDDcup99 datasets

In these experiments, we test our proposed method for network anomaly detection using TUIDS and KDDcup99 network intrusion datasets. In case of both datasets, it converts all categorical attributes into numeric form and then computes \(\log_b(x_{ij})\) to normalize larger attribute values, where \(x_{ij}\) is a large attribute value and \(b\) depends on the attribute values. The TUIDS datasets contain both packet and flow level feature data prepared in our own testbed. We initially apply TreeCLUS on a subset of relevant features extracted using the MMIFS algorithm [19] in both TUIDS packet and flow level datasets to generate a stable number of clusters and label each cluster using CLUSLab as normal or anomalous. Then, it performs some steps with the KDDcup99 datasets. The KDDcup99 dataset comprises of 36 attacks except the normal class. The experimental results for both TUIDS and KDDcup99 datasets in terms of detection rate and false positive rate are given in Table 8. A comparison of our proposed method with the other competing algorithms viz., C4.5 [28], ID3 [28], CN2 [28], CBUID [11], TANN [29], HC-SVM [30], is given in Figure 4. It can be easily seen from the figure that the proposed TreeCLUS technique outperforms other competing algorithms in the terms of detection rate and false positive rate, especially in case of probe, U2R, and R2L attacks.

**Table 8: Results on both TUIDS and KDDcup99 intrusion datasets using proposed method**

<table>
<thead>
<tr>
<th>Type of traffic</th>
<th>No. of clusters</th>
<th>Correctly detected</th>
<th>Miss detected</th>
<th>Detection rate (%)</th>
<th>False positive rate (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TUIDS packet level</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Normal</td>
<td>9</td>
<td>37109</td>
<td>786</td>
<td>0.9835</td>
<td>0.0164</td>
</tr>
<tr>
<td>DoS</td>
<td>16</td>
<td>29997</td>
<td>616</td>
<td>0.9799</td>
<td>0.0166</td>
</tr>
<tr>
<td>Probe</td>
<td>5</td>
<td>7637</td>
<td>120</td>
<td>0.9845</td>
<td>0.0014</td>
</tr>
<tr>
<td>Overall</td>
<td>26</td>
<td>84743</td>
<td>1522</td>
<td>0.9826</td>
<td>0.0114</td>
</tr>
<tr>
<td><strong>TUIDS flow level</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Normal</td>
<td>5</td>
<td>16486</td>
<td>284</td>
<td>0.9830</td>
<td>0.0169</td>
</tr>
<tr>
<td>DoS</td>
<td>14</td>
<td>229796</td>
<td>57</td>
<td>0.9997</td>
<td>0.0016</td>
</tr>
<tr>
<td>Probe</td>
<td>4</td>
<td>9235</td>
<td>255</td>
<td>0.9731</td>
<td>0.0149</td>
</tr>
<tr>
<td>Overall</td>
<td>23</td>
<td>40092</td>
<td>640</td>
<td>0.9832</td>
<td>0.0161</td>
</tr>
<tr>
<td><strong>KDDcup99</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Normal</td>
<td>5</td>
<td>59901</td>
<td>692</td>
<td>0.9885</td>
<td>0.0113</td>
</tr>
<tr>
<td>DoS</td>
<td>14</td>
<td>229796</td>
<td>57</td>
<td>0.9997</td>
<td>0.0016</td>
</tr>
<tr>
<td>Probe</td>
<td>5</td>
<td>4018</td>
<td>148</td>
<td>0.9645</td>
<td>0.0160</td>
</tr>
<tr>
<td>U2R</td>
<td>13</td>
<td>151</td>
<td>77</td>
<td>0.9623</td>
<td>0.1973</td>
</tr>
<tr>
<td>R2L</td>
<td>5</td>
<td>14007</td>
<td>2182</td>
<td>0.8652</td>
<td>0.1335</td>
</tr>
<tr>
<td>Overall</td>
<td>42</td>
<td>307873</td>
<td>3156</td>
<td>0.9898</td>
<td>0.0102</td>
</tr>
</tbody>
</table>

Figure 4: Comparison of our proposed method with competing algorithms

5. CONCLUSION AND FUTURE WORK

In this work, we present a tree based subspace clustering technique for unsupervised network anomaly detection in high dimensional datasets. We generate and analyze of clusters stability for each cluster by using an ensemble of multiple cluster indices. We also introduced a multi-objective cluster labelling technique to label each stable cluster as normal or anomalous. The major attractions of our proposed method are: (i) TreeCLUS does not require the number
of clusters initially. (ii) It is independent of any proximity measure. (iii) CLUSLab is a multi-objective cluster labelling technique, and (iv) It exhibits a high detection rate and low false positive rate, especially in case of probe, U2R, and R2L attacks. Hence, our proposed method is superior compared with existing unsupervised network anomaly detection techniques. Work on a faster, incremental version of TreeCLUS is underway for both numeric and mixed type network intrusion data.

6. ACKNOWLEDGMENTS

This work is supported by Department of Information Technology, MCT and Council of Scientific & Industrial Research (CSIR), Government of India. The research is also funded by the National Science Foundation (NSF), USA under grants CNS-0958767 and CNS-0851783. The authors are thankful to the funding agencies.

7. REFERENCES


