# Threshold-based Clustering for Intrusion Detection Systems[§]

Vladimir Nikulin

Computer Sciences Laboratory, RSISE, Australian National University,
Canberra, Australia

## ABSTRACT

Signature-based intrusion detection systems look for known, suspicious patterns in the input data. In this paper we explore compression of labeled empirical data using threshold-based clustering with regularization. The main target of clustering is to compress training dataset to the limited number of signatures, and to minimize the number of comparisons that are necessary to determine the status of the input event as a result. Essentially, the process of clustering includes merging of the clusters which are close enough. As a consequence, we will reduce original dataset to the limited number of labeled centroids. In a complex with $k$-nearest-neighbor ($kNN$) method, this set of centroids may be used as a multi-class classifier. Clearly, different attributes have different importance depending on the particular training database. This importance may be regulated in the definition of the distance using linear weight coefficients. The paper introduces special procedure to estimate above weight coefficients.

The experiments on the KDD-99 intrusion detection dataset have confirmed effectiveness of the proposed methods.

**Keywords:** distance-based clustering, $k$-nearest-neighbor method, intrusion detection

## 1. INTRODUCTION

Most deployed intrusion detection systems follow a signature-based approach[1] where attacks are identified by matching each input event against predefined signatures that model malicious activity. In difference, the basic idea of anomaly-based systems[2] is: if a new sample is "close" enough to the set of normal clusters, it is considered normal; otherwise, it is classified as abnormal.

The paper proposes an indirect $kNN$[3] method which includes the following 2 steps. Firstly, we compress training dataset to the limited set of centroids or signatures with labels computed according to the data within corresponding clusters as an averages. Then, we form multi-classifier using $kNN$ clustering method applied to the above set of centroids.

Model-based and distance-based clustering algorithms have a heuristical nature. Respectively, their performance depends essentially on the selection of initial settings if number of clusters is bigger than one. The threshold-based clustering[4] model is not immune completely against this problem as well, but, in most cases, dependence between initial settings and final results tends to zero asymptotically as sample size grows.

As a consequence of dynamic nature of clustering process some particular centroids may become too close to be considered as an independent elements of the classifier, and it was noticed that bigger number of signatures or centroids (after some point) will lead to worse results. Respectively, we will merge similar clusters, and will reduce number of clusters as a result. Experiments on a large real datasets have confirmed effectiveness of this operation.[5] The primary target of this research was to develop an automatical system in order to improve substantially contest criterion using KDD-99 datasets.

A special hard regularization represents an essential element of the proposed system. Using this regularization we will make clusters uniform in terms of labels as a main targets of classification. A soft regularization with additional regulation parameters will require special tuning depending on the particular dataset, and may be regarded as a further development.

In a real-time environment, the efficiency of data analysis is critical. For example, on-line fraud detection systems need to respond with an approval or rejection on the transaction within a few seconds. In this paper, we focus on the automatical systems for classification of different patterns. The advantage of signature-based designs is the fact that

[§] Email: vladimir.nikulin@anu.edu.au; Web: http://rsise.anu.edu.au/csl/

Data Mining, Intrusion Detection, Information Assurance, and Data Networks Security 2006,
edited by Belur V. Dasarathy, Proc. of SPIE Vol. 6241, 62410E, (2006) · 0277-786X/06/$15 · doi: 10.1117/12.665326

they can identify attacks with required accuracy and they tend to produce fewer alarms comparing with anomaly-based systems.[1] The problem of alert analysis has become very important.[6] The large amount of alerts can overwhelm security administrators and prevent them from adequately understanding and analyzing the security state of the network, and initiating appropriate response in a timely fashion.

## 2. THE MODEL

Let $\mathbf{X} = (x_t, y_t), t = 1..N$, be a training sample of observations where $x_t$ is $\ell$-dimensional vector of attributes, $y_t$ is the label in a form of $m$-dimensional vector of non-negative components: $y_{ti} \in \{0, 1\}, \sum_{i=1}^{m} y_{ti} = 1$.

For example, in the following below Section 3 we consider KDD-99 database with $m = 5$ labels: 1) "normal" and 4 types of intrusions 2) "PRB", 3) "DOS", 4) "U2R" and 5) "R2L". Accordingly, we define matrix $Y = \{y_{ij}, i, j = 1..m\}$ :

**Table 1.** Matrix of labels $Y$.

| Label | normal | PRB | DOS | U2R | R2L |
|---|---|---|---|---|---|
| normal | 1 | 0 | 0 | 0 | 0 |
| PRB | 0 | 1 | 0 | 0 | 0 |
| DOS | 0 | 0 | 1 | 0 | 0 |
| U2R | 0 | 0 | 0 | 1 | 0 |
| R2L | 0 | 0 | 0 | 0 | 1 |

As a base, we employ in this study a reasonably economical technique named Leader Algorithm which was introduced in Ref.7, pp. 75-76, and later was used[4] in application to the intrusion detection. The following below Algorithm 1 has two significant differences comparing with Leader Algorithm. Firstly, the Algorithm 1 use flexible centroids (instead of permanent leaders) which will be recomputed after any change of the corresponding cluster. Secondly, the Algorithm 1 will be used in application to the labelled data, and we would be interested to make clusters uniform in the terms of labels. Respectively, we will apply special regularization $R$ in (1) and (2), which should be powerful enough so that clustering will be conducted inside particular subsets with the same labels.

**Algorithm 1** Threshold-based Clustering

```
1: Select threshold parameter $H$ and distance $\Phi$ (for example, it may be squared distance or sum of absolute differences);
2: initialize $t := 1$, number of clusters $\tau := 1$, and the first cluster with centroid $q_\tau := x_t$ as a first element in the training dataset;
3: $t := t + 1$, obtain a sequential data-instance $x_t$ and compute
   $$\begin{cases} D = \min\limits_{c=1..\tau} \left[\Phi(x_t, q_c) + R(y_t, f_c)\right]; \\ j = \operatorname*{argmin}\limits_{c=1..\tau} \left[\Phi(x_t, q_c) + R(y_t, f_c)\right]; \end{cases} \qquad (1)$$
4: if $D \leq H$, then assign $x_t$ to the cluster $j$ and recompute $q_j$ and $f_j$ as a sample averages of the internal data;
5: if $D > H$, then create a new cluster with centroid $q_{\tau+1} := x_t$, $\tau := \tau + 1$;
6: repeat steps 3-5, until no instances are left in the training set.
```

Then, we compute vector of weight coefficients and matrix of frequencies

$$\begin{cases} \overline{w} = \{w_c = \frac{N_c}{N}, c = 1..\tau\}; \\ F = \{f_{ci} = \frac{n_{ci}}{N_c} \geq 0, c = 1..\tau, i = 1..m, \sum_{i=1}^{m} f_{ci} = 1\} \end{cases}$$

where any row in the matrix $F$ represents particular cluster as an average of internal labels, $n_{ci}$ is the number of elements with label $i$ in the cluster $\mathbf{X}_c$, and $N_c = \sum_{i=1}^{m} n_{ci}$ is the total number elements in the cluster $\mathbf{X}_c$, $N = \sum_{c=1}^{\tau} N_c$.

DEFINITION 2.1. *We define 1) "a distance to the cluster" or 2) "a distance between clusters" as 1) a distance to the corresponding centroid or 2) a distance between corresponding centroids assuming that the distance represents a sum of two terms: 1) for attributes and 2) for labels.*

DEFINITION 2.2. *Assuming that $k \leq \tau$, we form an indirect kNN classifier as a complex of 2 matrices: 1) centroids and 2) frequencies. The classification includes 2 steps: for any data-instance*

1) *find $k$ closest centroids $q_c, c = 1..k$, according to the regularized distance $\Phi$;*
2) *compute vector $\sum_{c=1}^{k} w_c \overline{f}_c$ as a weighted average of $c = 1..k$ rows of the matrix of frequencies, and define label according to the maximal component (empirical maximum likelihood).*

## 2.1. Threshold-based Clustering with Merging

We can develop the Algorithm 1 by including "backward" merging operation of the existing clusters which are close enough. The algorithm requires update of the matrix of distances between clusters (or centroids) after any transaction. This operation will double the required computation time assuming that the number of clusters remains constant. But, in fact, the backward operation may reduce significantly (subject to the properly selected regulation parameters) the number of clusters or clustering size and, as a consequence, the Algorithm 2 may be faster comparing with the Algorithm 1 for the same forward threshold parameter.

**Algorithm 2** Threshold-based Clustering with Merging

```
1: Select forward and backward threshold parameters $H_F, H_B, H_F \geq H_B$, and distance $\Phi$;
2: initialize $t := 1$, number of clusters $\tau := 1$, the first cluster with centroid $q_\tau := x_t$ as a first element in the training
   dataset;
3: $t := t + 1$, obtain a sequential data-instance $(x_t, y_t)$ and compute
```

$$\begin{cases} D = \min\limits_{c=1..\tau} \left[\Phi(x_t, q_c) + R(y_t, f_c)\right]; \\ j = \operatorname*{argmin}\limits_{c=1..\tau} \left[\Phi(x_t, q_c) + R(y_t, f_c)\right]; \end{cases} \tag{2}$$

```
4: if $D \leq H_F$, then assign $x_t$ to the cluster $j$ and recompute $q_j$ and $f_j$ as a sample averages;
5: if $D > H_F$, then create a new cluster with centroid $q_{\tau+1} := x_t$, $\tau := \tau + 1$;
6: if $\tau \geq 2$, compute triangle matrix of distances between centroids, find minimal distance $d_{min}$ and corresponding
   centroids;
7: merge 2 nearest clusters if $d_{min} < H_B, \tau := \tau - 1$;
8: repeat steps 3-7, until no instances are left in the training set.
```

REMARK 2.1. *The simplest definition of $R$ may be as follows*

$$R(y_t, f_c) = \begin{cases} 0 & \textit{if} \quad y_t = f_c; \\ A, & \textit{otherwise} \end{cases} \tag{3}$$

*where the constant $A$ represents an upper bound for the loss function $\Phi$ (in the following Section 3.3 we used value of the parameter $A = 75$ which is slightly bigger comparing with the upper bound for $\Phi$).*

As a next step after Algorithm 2 we can apply suitable adjustment in order to conduct further smoothing. For example, it may be an algorithm within the $CM$ framework.[8]

The algorithms within $CM$-framework have a heuristical nature. Respectively, their performance depend essentially on the selection of initial settings if number of clusters is bigger than one. Ref.[9] proposed an adaptive technique that grows the clusters automatically. Adaptive Algorithm 3 can identify $\tau$ clusters in an input data set by merging existing clusters and by creating new ones while keeping the number of clusters constant. In difference to the above Algorithms 1 and 2 this algorithm use a flexible threshold parameter which should be re-computed as a minimal distance between existing centroids after "service" of any data-instance.

**Algorithm 3** Adaptive Clustering with Merging

```
1: Select distance $\Phi$ and number of clusters $\tau$;
2: $t := 0$, and select $\tau$ initial centroids;
3: compute triangle matrix of distances between centroids, find minimal distance $d_{min}$ and corresponding centroids;
4: $t := t + 1$, compute vector of distances between data-instance $x_t$ and centroids, find minimal distance $d_t$;
5: if $d_t \leq d_{min}$, assign the element to the nearest cluster;
6: if $d_t > d_{min}$, merge 2 nearest clusters and create new cluster with $x_t$ as a centroid;
7: repeat steps 3-6, until no instances are left in the training set.
```

## 3. THE KDD-99 INTRUSION DETECTION DATABASE

The 1998 DARPA Intrusion Detection Evaluation Program was prepared and managed by MIT Lincoln Labs. The objective was to survey and evaluate research in intrusion detection. A standard set of data to be audited, which includes a wide variety of intrusions simulated in a military network environment, was provided. The KDD-99 data mining contest[10] used a version of this dataset.

Lincoln Labs set up an environment to acquire nine weeks of raw TCP (Transmission Control Protocol) data for a local-area network (LAN) simulating a typical U.S. Air Force LAN. They operated the LAN as if it were a true Air Force environment, but peppered it with multiple attacks.

The raw training data was about four gigabytes of compressed binary TCP data from seven weeks of network traffic. This was processed into about five million connection records. Similarly, the two weeks of test data yielded around two million connection records.

A connection is a sequence of TCP packets starting and ending at some well defined times, between which data flows to and from a source IP (Internet Protocol) address to a target IP address under some well defined protocol. Each connection is labelled as either normal, or as an attack with exactly one specific attack type. Each connection record consists of about 100 bytes.

Attacks fall into four main categories:

1) **DOS**: denial-of-service;

2) **R2L**: remote-to-local, unauthorized access from a remote machine, e.g., guessing password;

3) **U2R**: user-to-root, unauthorized access to local superuser (root) privileges, e.g., various “buffer overflow” attacks;

4) **PRB**: surveillance and other probing, e.g., port scanning.

It is important to note that the test data is not from the same probability distribution as the training data, and it includes specific attack types which are absent in the training data. This makes the task more realistic. Some intrusion experts believe that most novel attacks are variants of known attacks and the ”signatures” of known attacks can be sufficient to catch novel variants. The datasets contain a total of 24 training attack types, with an additional 14 types in the test data.

More specifically, we used the complex of two datasets: 1) training (485797 rows); 2) testing (311029 rows).

There are 5 types of labels including “normal”, and 4 types of intrusions “DOS”, “R2L”, “U2R” and “PRB” which were identified according to the given classification.

Given a training set of records with known labels, the problem is to learn a model in terms of attributes. Traditionally, the goal has been to minimize the number of misclassified records.

Any data-unit represents a 42-dimensional vector with label plus 41 attributes, that usually characterize network traffic behaviour. Detailed description of the attributes may be found in Ref.11, 12 and 13.

All attributes are numerical except second, third and fourth which we replaced by vectors of 3, 24 and 8 binary-valued variables.

We removed variables NN25 and 26, because they equal to zero. As a result, we formed completely numerical dataset where any row represents a 71 dimensional vector of non-negative attributes.

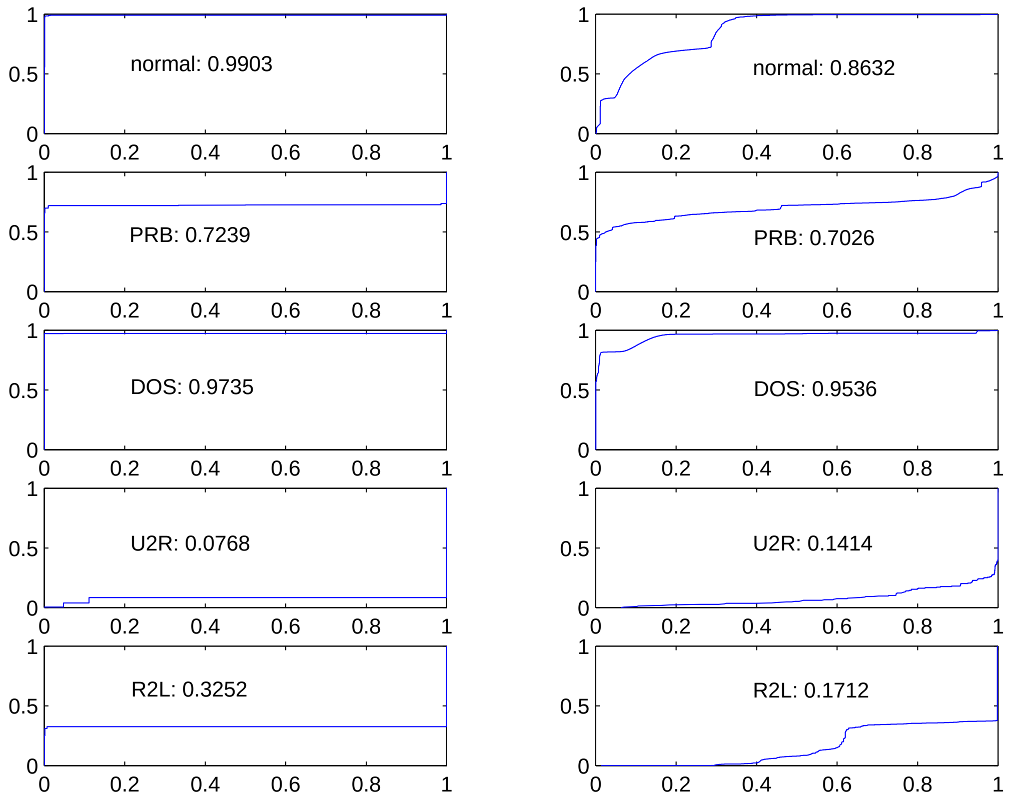


**Figure 1.** $ROC$ curves (used Algorithm 2) where numbers correspond to the $AUC$ -Area Under Curve. Left column represents expected probabilities which were computed according to the direct $kNN$ method with $k = 1$, right column represents smoothed expected probabilities which were computed according to the Bayesian method (see for more details Section 3.3).

## 3.1. The 1999 KDD Data Mining Contest Results

The winning algorithms of the KDD-99 Contest were built using principles of decision trees.[14]

In particular, C5.0 decision trees were employed by the winner[15] of the KDD-99 Contest. The training process utilized 50 data subsets each having all records from both the U2R and the R2L attack categories, 4000 records from the PRB category, 80000 from normal category, and 400000 from the DOS category. This was done to make sure that there were sufficient records present from each attack category to build decision tree models. For each of the above training subsets, the researchers created ten C5.0 decision trees using error cost and boosting options. The final predictions were computed on top of the 50 predictions each obtained from one decision tree, by minimizing the conditional risk.

The second contestant used model named $LLSoft$.[16] The corresponding algorithm generated a set of locally optimal decision trees (called the decision forest) from which the optimal subset of trees (called the subforest) was selected for predicting new cases. $LLSoft$ used only 10% of the KDD training data randomly sampled from the entire training dataset.

**Table 2.** The confusion matrices of the contestants of the KDD-99 Cup. From the top: first and second results.

| Method | Label | normal | PRB | DOS | U2R | R2L | TOTAL | Correct |
|---|---|---|---|---|---|---|---|---|
| C5.0 | normal | 60262 | 243 | 78 | 4 | 6 | 60593 | 99.45% |
| C5.0 | PRB | 511 | 3471 | 184 | 0 | 0 | 4166 | 83.32% |
| C5.0 | DOS | 5299 | 1328 | 223226 | 0 | 0 | 229853 | 97.12% |
| C5.0 | U2R | 168 | 20 | 0 | 30 | 10 | 228 | 13.16% |
| C5.0 | R2L | 14527 | 294 | 0 | 8 | 1360 | 16189 | 8.40% |
| C5.0 | TOTAL | 80767 | 5356 | 223488 | 42 | 1376 | 311029 | |
| C5.0 | False | 25.39% | 35.19% | 0.12% | 28.57% | 1.16% | | |
| LLSoft | normal | 60244 | 239 | 85 | 9 | 16 | 60593 | 99.42% |
| LLSoft | PRB | 458 | 3521 | 187 | 0 | 0 | 4166 | 84.52% |
| LLSoft | DOS | 5595 | 227 | 224029 | 2 | 0 | 229853 | 97.47% |
| LLSoft | U2R | 177 | 18 | 4 | 27 | 2 | 228 | 11.84% |
| LLSoft | R2L | 14994 | 4 | 0 | 6 | 1185 | 16189 | 7.32% |
| LLSoft | TOTAL | 81468 | 4009 | 224305 | 44 | 1203 | 311029 | |
| LLSoft | False | 26.05% | 12.17% | 0.12% | 38.64% | 1.50% | | |

One of the contestants used $PNrule$ model which is based on the principles of association rules.[17] The model consists of positive rules (P-rules) that predict presence of the class, and negative rules (N-rules) that predict absence of the class. The model is learned in two phases. The first phase discovers a few P-rules that capture most of the positive cases for the target class while keeping the false positive rate at a reasonable level. The goal of the second phase is to discover a few N-rules that remove most of the false positives introduced by the union of all P-rules while keeping the detection rate above an acceptable level. The sets for P- and N-rules are ranked according to certain statistical measures.

**Table 3.** The confusion matrix of the contestant of the KDD-99 Cup with method named NPrule,[18] second confusion matrix was obtained using Algorithm 1.

| Method | Label | normal | PRB | DOS | U2R | R2L | TOTAL | Correct |
|---|---|---|---|---|---|---|---|---|
| PNrule | normal | 60316 | 175 | 75 | 13 | 14 | 60593 | 99.54% |
| PNrule | PRB | 889 | 3042 | 26 | 3 | 206 | 4166 | 73.02% |
| PNrule | DOS | 6815 | 57 | 222874 | 106 | 1 | 229853 | 96.96% |
| PNrule | U2R | 195 | 3 | 0 | 15 | 15 | 228 | 6.58% |
| PNrule | R2L | 14440 | 12 | 1 | 6 | 1730 | 16189 | 10.69% |
| PNrule | TOTAL | 82655 | 3289 | 222976 | 143 | 1966 | 311029 | |
| PNrule | False | 27.03% | 7.51% | 0.05% | 89.51% | 12.00% | | |
| Alg.1 | normal | 58037 | 1349 | 359 | 784 | 64 | 60593 | 95.78% |
| Alg.1 | PRB | 291 | 3560 | 232 | 5 | 78 | 4166 | 85.45% |
| Alg.1 | DOS | 5343 | 412 | 223621 | 0 | 477 | 229853 | 97.29% |
| Alg.1 | U2R | 80 | 47 | 69 | 22 | 10 | 228 | 9.65% |
| Alg.1 | R2L | 10864 | 49 | 8 | 9 | 5259 | 16189 | 32.49% |
| Alg.1 | TOTAL | 74615 | 5417 | 224289 | 820 | 5888 | 311029 | |
| Alg.1 | False | 22.22% | 34.28% | 0.3% | 97.32% | 10.68% | | |

The main goal is to discover a small number of rules in terms of attributes which cover most of positive examples of the target class and very few of the negative examples. For numerical attributes, mechanism to find an effective ranges of values is critical in the formation of rules. The authors of $PNrule$[18] used intuitive clustering mechanism: 1) form a small number of ranges of equal span; 2) merge or split the ranges such that the number of records in each range satisfies certain pre-specified minimum and maximum requirements on the cluster size. Similar technique may be found in Ref.19.

**Table 4.** The confusion matrices.

| Method | Label | normal | PRB | DOS | U2R | R2L | TOTAL | Correct |
|---|---|---|---|---|---|---|---|---|
| Alg.2 | normal | 58966 | 468 | 785 | 67 | 307 | 60593 | 97.31% |
| Alg.2 | PRB | 423 | 3209 | 399 | 0 | 135 | 4166 | 77.03% |
| Alg.2 | DOS | 5284 | 183 | 223901 | 0 | 485 | 229853 | 97.41% |
| Alg.2 | U2R | 167 | 24 | 3 | 8 | 26 | 228 | 3.51% |
| Alg.2 | R2L | 10592 | 7 | 12 | 0 | 5578 | 16189 | 34.46% |
| Alg.2 | TOTAL | 75432 | 3891 | 225100 | 75 | 6531 | 311029 | |
| Alg.2 | False | 21.83% | 17.53% | 0.53% | 89.33% | 14.59% | | |
| Alg.3 | normal | 43021 | 3241 | 382 | 0 | 13949 | 60593 | 71% |
| Alg.3 | PRB | 252 | 2061 | 1056 | 0 | 797 | 4166 | 49.47% |
| Alg.3 | DOS | 2163 | 6406 | 220417 | 0 | 867 | 229853 | 95.89% |
| Alg.3 | U2R | 29 | 133 | 1 | 0 | 65 | 228 | 0.0% |
| Alg.3 | R2L | 130 | 8 | 11 | 0 | 16040 | 16189 | 99.08% |
| Alg.3 | TOTAL | 45595 | 11849 | 221867 | 0 | 31718 | 311029 | |
| Alg.3 | False | 5.65% | 82.61% | 0.65% | 0.0% | 49.43% | | |

The quality of detection was measured using the contest criterion ($CC$):

$$Q = \frac{1}{N} Tr\left(C'S\right) = \frac{1}{N} \sum_{i=1}^{m} \sum_{j=1}^{m} c_{ij} s_{ij}, \tag{4}$$

which is based on the given cost matrix (Table 5) and the confusion matrix (see, for example, Tables 2, 3 and 4):

$$S = \{s_{ij}, i = 1..m, j = 1..m, \sum_{i=1}^{m} \sum_{j=1}^{m} s_{ij} = N\}$$

where $s_{ij}$ is the number of detection of the label $j$ in the case of actual label $i$.

**Table 5.** The given cost matrix.

| Label | normal | PRB | DOS | U2R | R2L |
|---|---|---|---|---|---|
| normal | 0 | 1 | 2 | 2 | 2 |
| PRB | 1 | 0 | 2 | 2 | 2 |
| DOS | 2 | 1 | 0 | 2 | 2 |
| U2R | 3 | 2 | 2 | 0 | 2 |
| R2L | 4 | 2 | 2 | 2 | 0 |

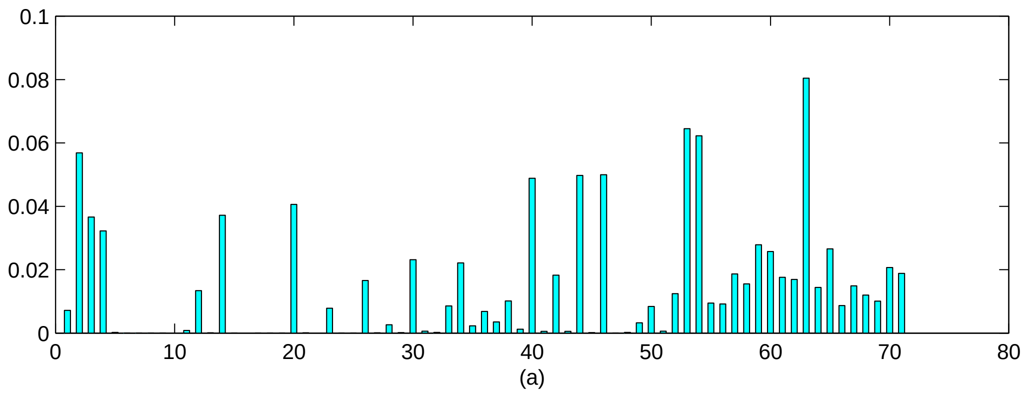
## 3.2. Computation of weights for different attributes and Algorithm 3

We used Algorithms 1 and 2 with

$$\Phi(x_t, q_c) = \sum_{i=1}^{\ell} \lambda_i \cdot \phi_{tci}, \quad \phi_{tci} = |x_{ti} - q_{ci}|, \quad \lambda_i = 1, \quad i = 1..\ell, \tag{5}$$

and $kNN$-method with $k = 1$. Experiments with *Euclidean* metric produced worse results in the sense of the $CC$. Also, attempts to use $k > 1$ did not make any significant improvement.

Clearly, different attributes have different importance depending on the particular training database. This importance may be regulated in the definition of the distance (5) using weight coefficients $\lambda_i, i = 1..\ell$. Particular values of the coefficients may be estimated using similar method as it was proposed in the recent paper.[20]



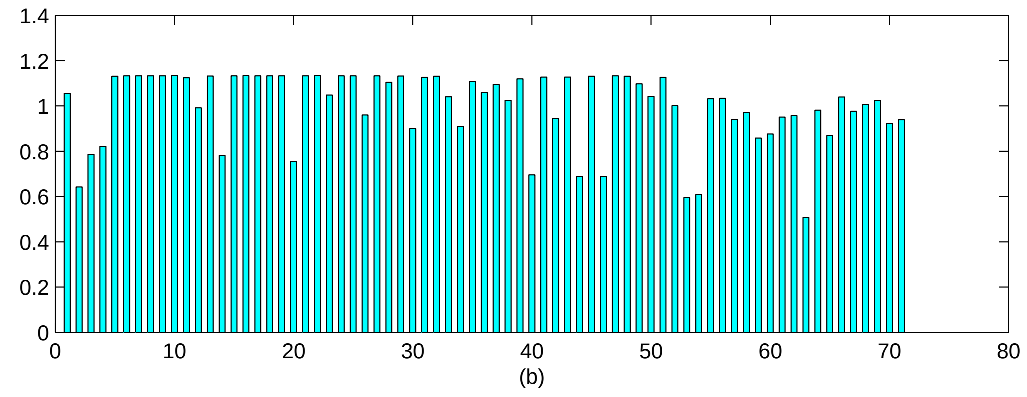


**Figure 2.** (a) support coefficients $\xi$, see (7); (b) corresponding weight coefficients $\lambda$, see (8) where $\gamma = 10$.

We can split training dataset into $m$ subsets $\Lambda_u$ according to $m$ different labels. Then we compute 3D matrix

$$\psi_{uvi} = \frac{1}{n_u} \sum_{\mathbf{x}_t \in \Lambda_u} \min_{L(q_c)=v} \phi_{tci} \tag{6}$$

where $n_u = \#\Lambda_u$, and symbol $L(q_c) = v$ means that label of the cluster with centroid $q_c$ is $v$.

It is assumed that initial clustering configuration was computed using the Algorithm 1 or 2 with distance (5). As a next step we make an assessment according to the (6) of the rule $u \to v$ for any particular attribute (note that bigger value corresponds to the smaller support).

Finally, we will compute general support taking into account given cost matrix

$$\xi_i = \sum_{u=1}^{m} \sum_{v=1}^{m} \psi_{uvi} \cdot c_{uv}, \tag{7}$$

and recompute weight coefficients

$$\lambda_i \propto \exp\{-\gamma \cdot \xi_i\}, \quad \sum_{i=1}^{\ell} \lambda_i = \ell. \tag{8}$$

DEFINITION 3.1. *We define Algorithm 3 as an identical to the Algorithm 2 with distance (5) and weight coefficients (8).*

## 3.3. Experimental Results

According to Ref.21, and in order to give the same importance to each attribute, the attributes were normalized to the range between 0 and 1 using the maximum values obtained. Another normalization[4] which is based on the complex of 2 operations: 1) compute the difference between observation and *sample mean*, 2) divide the difference by the *standard deviation*, was not efficient in this particular case.

**Table 6.** Probability of detection (PD) $p_i$ and false alarm rate (FAR) $g_i, i = 1..5$, as a function of the Tables 2, 3 and 4.

| Method | | normal | PRB | DOS | U2R | R2L |
|---|---|---|---|---|---|---|
| Alg.2 | PD | 0.973 | 0.77 | 0.974 | 0.035 | 0.345 |
| Alg.2 | FAR | 0.218 | 0.175 | 0.0053 | 0.893 | 0.146 |
| Alg.1 | PD | 0.958 | 0.855 | 0.973 | 0.096 | 0.325 |
| Alg.1 | FAR | 0.222 | 0.343 | 0.003 | 0.973 | 0.107 |
| C5.0 | PD | 0.995 | 0.833 | 0.971 | 0.132 | 0.084 |
| C5.0 | FAR | 0.254 | 0.352 | 0.0012 | 0.286 | 0.012 |
| LLSoft | PD | 0.994 | 0.845 | 0.975 | 0.118 | 0.073 |
| LLSoft | FAR | 0.261 | 0.122 | 0.0012 | 0.386 | 0.015 |
| PNRule | PD | 0.995 | 0.73 | 0.97 | 0.066 | 0.107 |
| PNRule | FAR | 0.271 | 0.075 | 0.0005 | 0.895 | 0.12 |
| 1NN | PD | 0.996 | 0.75 | 0.973 | 0.035 | 0.0059 |
| 1NN | FAR | 0.275 | 0.161 | 0.0018 | 0.111 | 0.078 |

In general, we want a very high detection rate with a very low false alarm rate. However, there is a trade-off between these two measures.[2] This trade-off can be shown using ROC curves.

Figure 1 illustrates ROC curves of $1-\widehat{F}(1-\theta)$ where $\widehat{F}$ is an empirical distribution function of the event (“normal”, “PRB”, “DOS’, “U2R”, “R2L”) and $\theta$ is an expected probability of the corresponding event. In the first case (left column) we used expected probabilities obtained by the $kNN$ method with $k = 1$. In the second case (right column) we applied smoothing according to Bayesian formula

$$p_{ti} \propto \sum_{c=1}^{\tau} f_{ci} \cdot \exp\{-2\Phi(x_t, q_c)\}, i = 1..m,$$

where centroids $q_c$ and corresponding frequencies $f_{ci}$ were defined in the Section 2.

The cost matrix represents subjective preferences. As an alternative, we can use more traditional and objective characteristics: probability of correct detection

$$P_C = \sum_{i=1}^{m} \pi_i p_i = \frac{1}{N} \sum_{i=1}^{m} s_{ii},$$

and standard deviation

$$\mathcal{S}_{\text{dev}} = \sqrt{\frac{1}{N-1} \sum_{i=1}^{m} \sum_{j=1}^{m} s_{ij}(c_{ij} - Q)^2}$$

where $\pi_i = \frac{h_i}{N}$ is a prior empirical probability of the label $i$; $p_i = \frac{s_{ii}}{h_i}$ is an empirical probability of correct detection of the label $i$; $h_i = \sum_{j=1}^{m} s_{ij}, N = \sum_{i=1}^{m} h_i$. The values of $P_C$, $CC$ and $\mathcal{S}_{\text{dev}}$ may be found in the Table 9.

**Table 7.** Some simulation results and used regulation parameters. Columns "Create" and "Merge" display numbers of clusters which were created and merged during clustering process. Both Algorithms 1 and 2 used the same forward threshold parameter $H_F$.

| Algorithm 1 | | | Algorithm 2 | | | | |
|---|---|---|---|---|---|---|---|
| $H_F$ | $\tau$ | **CC** | $H_B$ | $\tau$ | Create | Merge | **CC** |
| 4 | 130 | 0.212282 | 1 | 110 | 183 | 73 | 0.231519 |
| 4.5 | 109 | 0.223076 | 1 | 92 | 145 | 53 | 0.222548 |
| 4.8 | 93 | 0.197133 | 1 | 79 | 126 | 47 | 0.192834 |
| 4.9 | 91 | **0.195532** | 1 | 77 | 125 | 48 | 0.192593 |
| 5 | 88 | 0.19658 | 1 | 73 | 123 | 50 | **0.189728** |
| 5.1 | 87 | 0.196593 | 1 | 72 | 118 | 46 | 0.191018 |
| 5.5 | 73 | 0.206296 | 1 | 68 | 110 | 42 | 0.195503 |
| 6 | 64 | 0.205534 | 1 | 54 | 90 | 36 | 0.205061 |
| 7 | 44 | 0.252285 | 1 | 40 | 56 | 16 | 0.197776 |
| 8 | 31 | 0.239544 | 1 | 30 | 42 | 12 | 0.237563 |

REMARK 3.1. *Note that good quality classifier requires a sufficient level of smoothing. Experiments with bigger number of clusters (with $\tau$ up to 2000) produced worse results.*

**Table 8.** Results which were obtained using Algorithm 3, see, also, Table 7.

| $H_F$ | $H_B$ | $\tau$ | Create | Merge | **CC** |
|---|---|---|---|---|---|
| 4.5 | 1 | 78 | 128 | 50 | 0.34083 |
| 4.8 | 1 | 69 | 113 | 44 | 0.204447 |
| 4.9 | 1 | 66 | 113 | 47 | 0.179752 |
| 5 | 1 | 64 | 110 | 46 | 0.179752 |
| 5.1 | 1 | 59 | 98 | 39 | 0.163088 |
| 5.2 | 1 | 59 | 99 | 40 | **0.158731** |
| 5.3 | 1 | 59 | 97 | 38 | 0.18552 |
| 5.5 | 1 | 57 | 98 | 41 | 0.178951 |
| 5.7 | 1 | 51 | 85 | 34 | 0.18188 |
| 5.9 | 1 | 47 | 75 | 28 | 0.427378 |
| 6 | 1 | 48 | 80 | 32 | 0.418598 |

REMARK 3.2. *The Table 8 demonstrates non-monotonical property of $\tau$ as a function of $H_F$ assuming that $H_B$ is a constant. This property follows from heuristical nature of the clustering process. As a consequence, there may be some small fluctuations.*

Similar statistical characteristics (as it is demonstrated in the Tables 2, 3, 4, 7, 8 and 9) may be found in Ref.22 for the following models:

1) Multilayer perceptron with **CC** = 0.2393;

2) Gaussian classifier with **CC** = 0.3622;

3) K-means clustering with **CC** = 0.2389;

4) Nearest cluster algorithm with **CC** = 0.2466;

5) Incremental radial basis with **CC** = 0.4164;

6) Leader algorithm with **CC** = 0.2528;

7) Hypersphere algorithm with **CC** = 0.2486;

8) Fuzzy ARTMAP with **CC** = 0.2497;

9) C4.5 decision tree with **CC** = 0.2396.

10) Multiclass-classifier model with **CC** = 0.2285

where the best Multiclass-classifier model represents a complex of Multilayer perceptron (more effective for PRB attacks), K-means (more effective for DOS and U2R attacks) and Gaussian classifier (more effective for R2L attacks) models.

**Table 9.** The Competition Criterion and other statistical characteristics.

| Method | $P_C$ | **CC** | $\mathcal{S}_{\text{dev}}$ |
|---|---|---|---|
| Algorithm 3 | 0.905186 | 0.158731 | 0.513872 |
| Algorithm 2 | 0.937732 | 0.189728 | 0.783618 |
| Algorithm 1 | 0.933993 | 0.195532 | 0.791480 |
| C5.0 | 0.927081 | 0.233097 | 0.883425 |
| $LLSoft$ | 0.929193 | 0.235628 | 0.894503 |
| $PNrule$ | 0.925885 | 0.238106 | 0.888375 |
| $1NN$ | 0.923332 | 0.252343 | 0.919711 |

REMARK 3.3. *Relations between training and test sub-sets are not symmetrical. So, probably, random selection procedure is an appropriate for test sample, but, clearly, we should optimize selection of the training sample. For example, we can use Algorithm 1 or 2.*

REMARK 3.4. *Comparing Tables 2, 3 and 4 we can draw conclusion that correct detection of the intrusion $R2L$ is the most important reason for the relatively good performance of the Algorithms 1 and 2. This fact may be explained by 3 factors 1) large sample size of the data labeled $R2L$; 2) similarity between "normal" and "R2L" types in terms of attributes and 3) high cost of the corresponding incorrect detection.*

*The Algorithm 3 with specially computed weight coefficients $\lambda$ (see Figure 2(b)) make nearly perfect detection of $R2L$ but, as a result, detection of other types will suffer. Consequently, probability of correct detection for the Algorithm 3 has the smallest value comparing with other algorithms (see Table 9).*

A Pentium 4, 2.8GHz, 512MB RAM, computer was used for the computations which were conducted according to the special program written in C. The separate computation time for training and testing was about 170 sec. for both Algorithms 1 and 2.

## 4. CONCLUDING REMARKS

The Table 9 demonstrates an absolute advantage of the Algorithms 1 and 2 in the sense of all 3 used criterions where the Algorithm 3 represents the only exception. The Algorithms 1 and 2 are automatical subject to the selection of distance and threshold parameters as an input where threshold parameters may be tuned as a result of simple experiments on the training dataset.

Section 3.2 introduces a very promising general approach for distance adjustment in order to improve performance of an algorithm according to the given criterion.

Further adjustment and smoothing may be conducted according to the adaptive[9] or $CM$ methods.[8] Besides, using the principles of universal clustering[23] we can test stability of the clustering configuration.

## ACKNOWLEDGMENTS

This work was supported by a grant of the Australian Research Council.

## REFERENCES

1. C. Kruegel and T. Toth, "Using Decesion Trees to Improve Signature-based Intrusion Detection," in *RAID*, G. Vigna, E. Jonsson, and C. Kruegel, eds., pp. 173–191, Springer-Verlag Berlin Heidelberg, 2003.
2. F. Gonzales and D. Dasgupta, "Anomaly detection using real-valued negative selection," *Genetic Programming and Evolvable Machines* **4**, pp. 383–403, 2003.
3. W. Henley and D. Hand, "A k-Nearest-Neighbour classifier for assessing consumer credit risk," *The Statistician* **45**(1), pp. 77–95, 1996.
4. L. Portnoy, E. Eskin, and S. Stolfo, "Intrusion detection with unlabeled data using clustering," in *ACM CSS Workshop on Data Mining Applied to Security, Philadelphia, PA: November 5-8*, 2001.
5. V. Nikulin, "Threshold-based clustering with hard regularization for intrusion detection systems," in *Workshop on Learning Algorithms for Pattern Recognition, AI-2005, Sydney*, R. Ghosh, B. Verma, and X. Li, eds., pp. 78–87, UTS, 2005.
6. X. Qin and W. Lee, "Statistical Causality Analysis of INFOSEC Alert Data," in *RAID*, G. Vigna, E. Jonsson, and C. Kruegel, eds., pp. 73–93, Springer-Verlag Berlin Heidelberg, 2003.
7. J. Hartigan, *Clustering Algorithms*, John Wiley  Sons, 1975.
8. V. Nikulin and A. Smola, "Parametric model-based clustering," in *Data Mining, Intrusion Detection, Information Assurance, and Data Network Security, 28-29 March 2005, Orlando, Florida, USA*, B. Dasarathy, ed., **5812**, pp. 190–201, SPIE, 2005.
9. S. Bhatia, "Adaptive k-means clustering," *The 17th International FLAIRS Conference, Miami Beach, Florida, May 17-19* , pp. 695–699, 2004.
10. KDD, "KDD-99 Classifier Learning Contest," in *Results of the 1999 KDD Classifier Learning Contest: http://www-cse.ucsd.edu/users/elkan/clresults.html*, 1999.
11. R. Lippmann, J. Haines, D. Fried, J. Korba, and K. Das, "Analysis and Results of the 1999 DARPA Off-Line Intrusion Detection Evaluation," in *RAID*, H. Debar, L. Me, and F. Wu, eds., pp. 162–182, Springer-Verlag Berlin Heidelberg, 2000.
12. A. Lazarevic, L. Ertoz, V. Kumar, and J. Srivastava, "A comparative study of anomaly detection schemes in network intrusion detection," in *Proceedings of the 4th International Conference on Data Mining, San Francisco, USA, May 1-3*, SIAM, 2003.
13. M. Mahoney and P. Chan, "An Analysis of the 1999 DARPA/Lincoln Laboratory Evaluation Data for Network Anomaly Detection," in *RAID*, G. Vigna, E. Jonsson, and C. Kruegel, eds., pp. 220–237, Springer-Verlag Berlin Heidelberg, 2003.
14. C. Elkan, "Results of the KDD'99 Classifier Learning," in *SIGKDD Explorations, 2* **1**, pp. 63–64, ACM SIGKDD, 2000.
15. B. Pfahringer, "Winning the KDD-99 Classification Cup: Bagged Boosting," in *SIGKDD Explorations, 2* **1**, pp. 65–66, ACM SIGKDD, 2000.
16. I. Levin, "KDD-99 Classifier Learning Contest LLSoft's Results Overview," in *SIGKDD Explorations, 2* **1**, pp. 67–75, ACM SIGKDD, 2000.
17. R. Agrawal, T. Imielinski, and A. Swami, "Mining association rules between sets of items in large databases," in *Proceedings of the ACM SIGMOD Conference Washington DC, USA, May 1993*, 1993.
18. R. Agarwal and M. Joshi, "PNrule: A New Framework for Learning Classified Models in Data Mining (A Case-Study in Network Intrusion Detection)," in *Proceedings of the 1st International Conference on Data Mining*, pp. 225–232, SIAM, 2001.
19. W. Lee, S. Stolfo, and K. Mok, "Mining audit data to build intrusion detection models," in *KDD-Proceedings*, 1998.
20. J. Friedman and J. Meulman, "Clustering objects on subsets of attributes," *Journal of Royal Statistical Society* **66**, pp. 815–849, 2004.
21. J. Gomez, D. Dasgupta, and O. Nasraoui, "A new gravitational clustering algorithm," in *Proceedings of the 4th International Conference on Data Mining, San Francisco, USA, May 1-3*, SIAM, 2003.
22. M. Sabhnani and G. Serpen, "Application of Machine Learning Algorithms to KDD Intrusion Detection Dataset within Misuse Detection Context," in *Proceedings of the International Conference on Machine Learning, Models, Technologies and Applications (MLMTA 2003), Las Vegas, NV, June 2003*, pp. 209–215, 2003.
23. V. Nikulin, "Universal clustering with family of power loss functions in probabilistic space," in *IDEAL 2005*, M. Gallagher, J. Hogan, and F. Maire, eds., **LNCS 3578**, pp. 311–318, Springer-Verlag, 2005.