A Data-Driven Metric Learning Based Scheme for Unsupervised Network Anomaly Detection

Roya Aliakbarisani [a] Abdorasoul Ghasemi [a,*]
Shyhtsun Felix Wu [b]

[a] *Faculty of Computer Engineering, K. N. Toosi University of Technology, Tehran, Iran*
[b] *Department of Computer Science, University of California Davis, CA*

Abstract

Most network anomaly detection systems (NADSs) rely on the distance between the connections' feature vectors to identify attacks. Traditional distance metrics are inefficient for these systems as they deal with heterogeneous features of network connections. In this paper, we address a clustering-based NADS employing a data-driven distance metric. This metric is outcome of a proposed metric learning (ML) method, which extracts its required side information from the training samples. The learned transformation matrix maps the connections' features to a new feature space in which normal and attack connections are more well-separated while the local neighborhood information of the connections' features is preserved using the Laplacian Eigenmap technique. The proposed NADS is evaluated over the Kyoto 2006+ and NSL-KDD datasets. The experimental results show that it has superior performance in comparison with a recent SVM-clustering based NADS that employs the traditional Euclidean distance function.

Key words: Network anomaly detection; Metric learning; Linear feature transformation; Clustering method; Similarity/dissimilarity constraints.

1 Introduction

Intrusion detection systems (IDSs) are pieces of software or hardware supplying the security of connections in computer networks. These security tools, which are used to detect suspicious traffic, have received a great amount of attention in recent decades. IDSs apply two common techniques: misuse detection and anomaly detection. Misuse detection

* Corresponding author.
Email addresses: R.Aliakbarisani@ee.kntu.ac.ir (Roya Aliakbarisani), arghasemi@kntu.ac.ir (Abdorasoul Ghasemi), wu@cs.ucdavis.edu (Shyhtsun Felix Wu).

schemes employ a predefined set of attack signatures to determine anomalous activities. These schemes compare each network connection against the predefined set to detect attacks. Hence, they are not able to detect zero-day attacks. On the other hand, an anomaly detection scheme models the normal traffic behavior and each deviation from this identified model is considered as a potential attack [1]. Anomaly detection schemes can detect new attacks and due to the growth of zero-day attacks in recent years more attention have been paid to them. Apart from network security, detection of anomalies as unexpected patterns in a dataset has wide variety of applications such as target detection in image processing [2] or fraud detection of credit cards in finance [3].

Machine learning techniques such as clustering and classification are the most common methods to group similar network connections and model normal traffic behavior. Distance-based classifiers such as support vector machine (SVM) [4] and k-nearest-neighbors [5] as well as distance-based clustering methods, e.g., k-means algorithm and its modifications [6, 7], have been frequently used for designing network anomaly detection systems (NADSs). On the other hand, the distance functions deployed by these clustering and classification methods strongly affect the performance of the underlying NADSs [8].

Traditional distance metrics like the Euclidean or Mahalanobis distance functions are usually used by the recent proposed NADSs. However, the network connections have heterogeneous features, and these metrics cannot properly reflect the distance between the connections' feature vectors to determine attacks. Furthermore, these metrics don't consider the importance of each feature in the underlying NADS and due to the different scales the effects of some features may dominate the effects of the others. Metric learning (ML) approach can overcome the disadvantages of the traditional metrics. It is the task of learning a distance function tuned to a specific application using provided side information [9]. Side information can be in the form of similarity/dissimilarity constraints. Similarity constraints are pairwise observations, which should be close to each other by applying the learned distance function. On the other hand, dissimilarity constraints are pairwise observations that should be far from each other in the new distance metric [9]. ML provides a proper application specific feature weighting. Different types of ML methods have been proposed and applied in real-world applications so far, e.g., LMLML [10], SERAPH [11] and COMID-SADL [12]. Learning a good distance metric has a vital role on the performance of the NADSs, which employ distance-based machine learning algorithms. These NADSs usually rely on a distance metric to measure the distance between the network connections' feature vectors. As a result, learning a proper distance metric taking into account the constraints of the NADSs could provide a reliable and fair comparison of the network connections' feature vectors, which improves the accuracy of the employed machine learning methods and eventually the performance of the underlying NADSs. Metric learning algorithms can also be useful for improving the performance of anomaly detection in other fields of study such as image processing [13].

In this paper, we propose a linear ML method for unsupervised NADSs, which employs distance-based clustering or classification methods. We suppose that there is not any predefined side information and we directly extract the similarity/dissimilarity constraints and finally a new distance metric from the training samples, i.e., the unlabeled feature vectors of the network connections. This metric is a feature weighting, which transforms the connections' feature vectors from one feature space into another. Besides, the proposed scheme reduces the dimensionality of the connections' features. Therefore, it can be useful to handle the challenge of processing high-dimensional data sets. It uses the Laplacian Eigenmap technique to preserve the intrinsic structure of the connections' feature vectors in the transformed space. The proposed ML method extracts a set of informative and discriminative features from the input features of the network connections, and hence the underlying NADSs can better classify similar and dissimilar connections.

The rest of this paper has been organized as follows. Section 2 reviews previous feature transformation methods for IDSs as well as some backgrounds on ML. Section 3 describes the details of the proposed unsupervised linear ML method. Section 4 presents the overall schema of an unsupervised clustering-based NADS using the designed ML entity. The performance of this NADS is evaluated in section 5 over the Kyoto 2006+ and NSL-KDD datasets before concluding the paper in section 6.

2 Related Works and Backgrounds

The performance of an IDS is directly related to training samples and their features. When these features contain discriminative information about normal or attack connections, the underlying system can effectively determine anomalous activities. Therefore, IDSs can employ proper feature transformation methods to extract informative features and improve their performance.

2.1 Feature Transformation Methods for IDSs

Feature transformation is a method of data pre-processing. This method applies a linear or nonlinear transformation to map the input features to a more discriminative feature space. Using these new features improves the performance of the underlying systems [14].

Principal component analysis (PCA) [15], linear discriminant analysis (LDA) [15] and its extensions such as semi-supervised LDA (SLDA) [16] and split and combine LDA (SC-LDA) [17] are some examples of the linear feature transformation methods which are employed by different applications. These methods may also reduce the dimensionality of feature space [15].

An IDS based on the back-propagation algorithm for training a multi-layer perceptron classifier is proposed in [18]. In the training phase, this system employs the vectors of labeled connections and it applies LDA for feature transformation and reduction. Using LDA, the connections' features are classified into maximally separated classes of data. For this objective, LDA maximizes the ratio of the between-class and the within-class distances of the connections' vectors to find a proper transformation matrix. Training and testing sets are selected from the NSL-KDD dataset with 41 features where deploying LDA reduces them to four features. The experimental results indicate that applying LDA leads to 97% data reduction and 94% time while the attack detection rate remains unchanged.

An IDS based on genetic algorithm (GA) is proposed in [19]. This IDS applies misuse detection technique and identifies the attacks by constructing a set of rules. It employs a modification of PCA for feature reduction. Instead of using the principal components (PCs), it extracts some information from the PCs to select a subset of features. It first calculates the covariance matrix of the samples and then computes and sorts the their eigenvalues. Finally, it selects the eigenvector corresponding to the smallest eigenvalue and deletes the feature with the largest coefficient in this eigenvector. The last step is repeated until only d features remain where d is the dimensionality of the output space. The feature reduction accelerates the process of GA, which creates classification rules to categorize the network traffic into normal or attacks. The resulting IDS can detects 98.38% of the normal data and 94.48% of the attack samples in the testing phase when it employs only three out of 41 features of the KDD cup 99 selected by the modified PCA.

Reduced Class-dependent Feature Transformation (RCDFT) is proposed in [20] for IDSs based on classification methods. In the training phase, this system applies a feature transformation method such as PCA or LDA to the samples of each class and calculates M different f-by-f transformation matrices. M is the number of classes in the training samples, and f is the dimensionality of the input space. For each sample, RCDFT finds M different mapped data using all the transformation matrices and generates a new sample with $M \times f$ features. Next, RCDFT applies PCA to map the new set of data to an f dimensional feature space. Finally, it trains a classifier using the mapped training samples. This method extracts new features from a larger set of features so these features are more informative. The experimental results over KDDTrain20%+ and KDDTest+ datasets demonstrate that RCDFT based IDS has higher detection rate and lower false alarm compared to IDSs, which employ LDA or PCA feature transformation techniques.

ML is also a feature transformation method. The main difference between the classical feature transformation techniques such as LDA or PCA and ML approach is that the former implicitly derive a metric through the projections while the latter deploys side information to achieve its key goal, i.e., learning the underlying metric [9].

Normal profiles of NADSs consist of some different areas in the feature space using the typical extracted network connections' features. Each of these areas represents a set of similar normal connections. A ML method adjusts the locations of the network connections in the new feature space in such a way that the similar connections tend to get closer and the dissimilar ones tend to move away. As a result, it can be the best feature transformation method for improving the performance of a NADS through the extraction of a more accurate normal profile. As well as, we can assign some properties to the learned feature transformation matrix using the objective function and constraints of ML scheme's optimization problem.

2.2 Backgrounds on Metric Learning

ML is the process of automatically learning an application-specific distance function. This method employs prior knowledge in the form of relative distance constraints or similarity/dissimilarity constraints [21]. A linear ML method calculates a transformation matrix for mapping data to the output feature space for better discrimination in the new space. After data transformation, a traditional distance function is used to measure the distance between the mapped data. As an example, the distance between $\mathbf{x}_i \in R^n$ and $\mathbf{x}_j \in R^n$ is measured as follows:

$$d(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{G}\mathbf{x}_i - \mathbf{G}\mathbf{x}_j\|_2^2 \tag{1}$$

Where $\mathbf{G} \in R^{n\times n}$ is the learned transformation matrix, and the Euclidean distance is the selected traditional metric. [9].

Given the dissimilarity constraints in D and the similarity constraints in S, the problem of learning a distance metric is formulated as a convex optimization problem in [22] as given by (2).

$$\begin{aligned} &\min_{\mathbf{A}} \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in S} \|\mathbf{x}_i - \mathbf{x}_j\|_{\mathbf{A}}^2 \\ &s.t. \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in D} \|\mathbf{x}_i - \mathbf{x}_j\|_{\mathbf{A}}^2 \geq 1, \qquad \mathbf{A} \geq 0 \end{aligned} \tag{2}$$

Where $\boldsymbol{A}$ is the desired transformation matrix. $\mathbf{A} \geq 0$ ensures the non-negativity and triangle inequality conditions of the metric. The objective is to minimize the distance between data points within the similar set, while the sum of the distance between data points in the dissimilar set should be more than an optional constant, e.g., 1.

The optimization problem of another ML method minimizes the distance between data points in the similarity constraints and maximizes the distance between data points within the dissimilarity constraints simultaneously. The corresponding optimization problem is given by:

$$\mathbf{W}^{*} = \underset{\mathbf{W}^{T}\mathbf{W}=\mathbf{I}}{\arg\max} \frac{\mathrm{tr}(\mathbf{W}^{T}\hat{\mathbf{S}}_{b}\mathbf{W})}{\mathrm{tr}(\mathbf{W}^{T}\hat{\mathbf{S}}_{w}\mathbf{W})} \tag{3}$$

Here, tr is the trace operator. $\mathbf{I}$ is an identity matrix of size n and $\mathbf{W}^{T}\mathbf{W} = \mathbf{I}$ is a constraint that prevents from degenerate solutions. $\hat{\mathbf{S}}_{w}$ and $\hat{\mathbf{S}}_{b}$ are the covariance matrices of the similar and dissimilar observations respectively. The numerator of (3) is the squared Euclidean distance between the dissimilar data points after transformation by matrix, $\mathbf{W}^{T}$. In the same way, the denominator of (3) is the squared Euclidean distance between the similar mapped data points.

Finally, $\mathbf{W}^{*} \subset R^{n\times d}$ is the desired transformation matrix where d is the dimensionality of the output space so that $d < n$. Dimensionality reduction is a significant property of this ML scheme. It is effective especially when we deal with high dimensional data. In [23] a heuristic iterative search framework is presented to solve the optimization problem of (3).

3 Proposed Metric Learning Method for Unsupervised Clustering-based NADSs

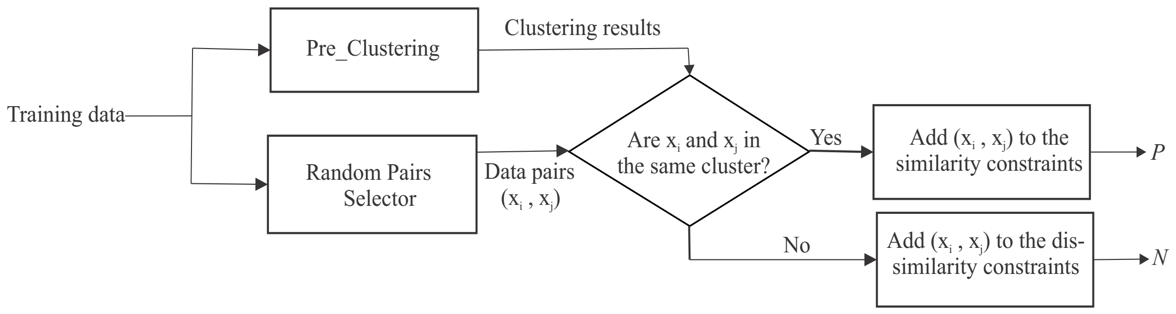
Many distance-based clustering methods have been proposed and applied for unsupervised NADSs [8]. The performance of the clustering algorithms has a crucial role in the accuracy of these NADSs. When a clustering method precisely determines the existing clusters in the feature space, testing data are also assigned to the true clusters. As a result, the underlying NADS will have acceptable performance. This issue emphasizes the importance of learning a proper distance metric in the performance of unsupervised clustering-based NADSs.

3.1 Motivation

As an example, consider a randomly selected subset of Kyoto 2006+ dataset, which is a well-known network dataset employed for the evaluation of this paper. Assume that the underlying classification problem is to determine to which of two classes (normal or attack) a connection's feature vector belongs using the first 13 features of each connection. A distance function is required to compare the feature vectors of the connections, where using the Euclidean distance would be the most obvious choice. However, a simple statistical analysis of the data shows that it will be incompetent. The average values of nine features across the dataset are from the interval [0,10], three features are from the interval [10,300] and the average value of one feature is 4773. Obviously, the computation of the Euclidean distance is directly affected by the single largest feature, and the effects of the other 12 features are almost ignored. Furthermore, we notice that normalization of the features' values does not significantly improve the performance of the traditional metrics since the importance of the features depends on the applications and each application requires its special feature weighting [9]. As a result, employing the traditional distance functions, such as the Euclidean metric, in conjunction with clustering and classification methods lead to inefficient approaches.

On the other hand, the network connections have heterogeneous features. For instance, "Duration", "Service" and "Source bytes" are some of the important features of the network datasets, which show the duration of the connections in seconds, the connections' service types, e.g., http or telnet and the number of data bytes sent by the source IP addresses, respectively. It is clear that these features have different units. Therefore, the Euclidean distance between two connections' feature vectors as the square root of the summation of some terms with different units and scales may not properly reflect the similarity of the connections' feature vectors. In this paper, we develop a ML scheme as a feature weighting method, which scales and rotates the data in order to consider these issues and can be used for unsupervised clustering-based NADSs.

In the following, we assume that $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m\}$ is the set of the connections' feature vectors as the training data and $Y = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_m\}$ is the set of the corresponding mapped data using the transformation $\mathbf{y}_i = f(\mathbf{x}_i)$. $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m]$ and $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_m]$ are the matrices of the training data and the mapped data, respectively. To the best of our knowledge, learning a proper distance metric is an almost new idea for improving the security of networks. In section 3.2 we present a method to extract the required side information of ML procedure which makes it applicable for unsupervised systems. As well as, in section 3.3 we



Training data
Pre_Clustering
Clustering results
Random Pairs Selector
Data pairs (x_i, x_j)
Are x_i and x_j in the same cluster?
Yes
No
Add (x_i, x_j) to the similarity constraints
Add (x_i, x_j) to the dissimilarity constraints
P
N

Fig. 1. The unsupervised proposed method for generating the similarity/dissimilarity constraints

highlight the importance of preserving the intrinsic structure of the samples in the course of ML process. Based on this idea, in section 3.4, we formulate a specific optimization problem as a new ML method which employs Laplacian Eigenmap technique to preserve local neighborhood information of samples. Finally, in section 4, we will design an unsupervised clustering-based NADS which derives a desired distance metric by applying the proposed ML scheme. Locality-preserving property strongly affects the performance of this NADS, as we will prove it by simulation.

3.2 Extracting Similarity/Dissimilarity Constraints

Side information is a necessary component of each ML scheme. It is usually in the form of prior knowledge about similar and dissimilar data points. However, this work focuses on unsupervised clustering-based NADSs, and we attempt to extract side information from unlabeled connections' feature vectors.

Fig. 1 shows the unsupervised proposed method in which the sets of similarity and dissimilarity constraints from the input training vectors are generated. A subset of connections' features is used in the clustering process. The features with the higher variances are selected for the *Pre-Clustering* since they are more discriminative in comparison with the features that show low variances. The number of these features can be determined by trial and error. The *Pre-Clustering* step groups the similar training data using the selected set of features and the results are employed for generating the similarity/dissimilarity constraints.

Random Pairs Selector randomly selects some pairs of the training connections' feature vectors, e.g., $(\mathbf{x}_i, \mathbf{x}_j)$. If both $\mathbf{x}_i$ and $\mathbf{x}_j$ have been assigned to the same cluster in the *Pre-*

Clustering step, $(\mathbf{x}_i, \mathbf{x}_j)$ is added to the similarity constraints, and otherwise, it is appended to the dissimilarity constraints. Finally, the sets of similarity, $P = \{(\mathbf{x}_i, \mathbf{x}_j) |\ \mathbf{x}_i \text{ and } \mathbf{x}_j \text{ are in the same class}\}$, and dissimilarity constraints, $N = \{(\mathbf{x}_i, \mathbf{x}_j) |\ \mathbf{x}_i \text{ and } \mathbf{x}_j \text{ are in two different classes}\}$, are constructed.

3.3 *Locality-Preserving Property*

The intrinsic structure of the network connections' feature vectors gives us some precious information about the similar and dissimilar patterns. For example, the normal connections, which have similar features' values in the input space, are more likely to represent the same type of normal patterns. This valuable information should not be lost in the transformation to the new feature space in the course of ML process.

Specifically, the local neighborhood information of connections' feature vectors should be preserved such that the adjacent vectors in the input feature space should also be close to each other in the mapped output space. We use the Laplacian Eigenmaps algorithm [24] to consider this issue in the designed NADS.

Laplacian Eigenmap method regards the input training vectors as the vertices of a graph. The edges of this graph connect each vertex to all its k-nearest-neighbors connections. The Laplacian Eigenmap algorithm minimizes the following cost function to ensure the locality-preserving property:

$$J_{Lap}(\mathbf{X}, f) = \frac{1}{2} \sum_i \sum_j \|\mathbf{y}_i - \mathbf{y}_j\|^2 s_{ij} \tag{4}$$

Where $\mathbf{y}_i$ is the mapped vector corresponding to the input vector $\mathbf{x}_i$ of the data matrix $\mathbf{X}$, i.e., $\mathbf{y}_i = f(\mathbf{x}_i)$, and s_{ij} is an element of the graph's adjacency matrix. This cost function assigns heavy penalties to the neighboring vectors in the input space which are far apart in the output space. Minimization of this cost function preserves the relative distance between each vector and its k-nearest-neighbors. This function can be written as follows [24]:

$$
\begin{aligned}
J_{Lap}(\mathbf{X}, f) &= \text{tr}\,(\sum_i \mathbf{y}_i d_i \mathbf{y}_i^T - \sum_i \sum_j \mathbf{y}_i s_{ij} \mathbf{y}_j^T) \\
&= \text{tr}\,(\mathbf{Y}(\mathbf{D}-\mathbf{S})\mathbf{Y}^T) = \text{tr}(\mathbf{YLY}^T)
\end{aligned}
\tag{5}
$$

$\mathbf{S}$ is the adjacency matrix and $\mathbf{D} = diag(d_1, \dots, d_m)$ is the degree matrix of the resulting graph where d_i is degree of vertex i. $\mathbf{L}_{m\times m} = \mathbf{D} - \mathbf{S}$ is the Laplacian matrix of this graph. We use the cost function of of locality preserving to formulate our ML scheme in section 3.4.

3.4 Unsupervised Metric Learning Optimization Problem

This section employs the extracted similarity/dissimilarity constraints described in section 3.2 and the locality-preserving method described in section 3.3 to formulate the proposed ML scheme.

ML is formulated as an optimization problem in which the objective function considers both the distance of the similarity constraints and the cost function of (5) to preserve the local neighborhood information as well as the distance between the pairs of the dissimilarity constraints in the output feature space simultaneously as given by (6).

$$
\mathbf{W}^* = \underset{\mathbf{W}^T\mathbf{W}=\mathbf{I}}{\arg\max} \frac{\sum_{(\mathbf{x}_i,\mathbf{x}_j)\in N} \|\mathbf{W}^T\mathbf{x}_i - \mathbf{W}^T\mathbf{x}_j\|^2}{\sum_{(\mathbf{x}_i,\mathbf{x}_j)\in P} \|\mathbf{W}^T\mathbf{x}_i - \mathbf{W}^T\mathbf{x}_j\|^2 + \alpha\,\text{tr}(\mathbf{W}^T\mathbf{XLX}^T\mathbf{W})}
\tag{6}
$$

Where $f(\mathbf{x}_i) = \mathbf{W}^T\mathbf{x}_i$ is the linear transformation function and constraint $\mathbf{W}^T\mathbf{W} = \mathbf{I}$ prevents from degenerate solutions. the numerator of (6) is sum of the distance between the pairs of the dissimilarity constraints in the output feature space. The denominator of (6) is sum of the distance between the pairs of the similarity constraints in the output feature space and the cost function of (5). We substitute $\mathbf{Y} = \mathbf{W}^T\mathbf{X}$ as the transformation function for $\mathbf{Y} = f(\mathbf{X})$ in $J_{Lap}(\mathbf{X}, f)$. As a result, the cost function can be written in the form of $J_{Lap}(\mathbf{X}, f) = \text{tr}(\mathbf{W}^T\mathbf{XLX}^T\mathbf{W})$. Finally, $\alpha \geq 0$ balances between pairwise distance of the mapped similarity constraints and the cost function.

$\mathbf{W}^* \in R^{n\times d}$ is the desired transformation matrix. Here, n is the dimensionality of the input and d is the dimensionality of the output feature spaces. In the case of $n = d$, we have $\mathbf{W}^T\mathbf{W} = \mathbf{WW}^T = \mathbf{I}$ and the learned metric is $\mathbf{A} = \mathbf{I}$. This special case is the Euclidean

Algorithm 1. The proposed ML method

```
1: Find $k$-nearest-neighbors of each connection's vector in the training data
2: Construct a graph $G = (V, E)$, $|V| = m$, where $m$ is the size of the training data and $E = \{(\mathbf{v}_i, \mathbf{v}_j)|\ \mathbf{v}_i$ is in the $k$-nearest-neighbors of $\mathbf{v}_j$or viceversa$\}$
3: Find the Laplacian matrix of $G$, $\mathbf{L} = \mathbf{D} - \mathbf{S}$, where $\mathbf{D}$ is the degree matrix and $\mathbf{S}$ is the adjacency matrix of $G$
4: Employ the proposed method in section 3.2 to generate the similarity/dissimilarity constraints
5: Calculate $\mathbf{S}_b$ and $\mathbf{S}_W$ according to (7) and (8)
6: Obtain the new matrix $\hat{\mathbf{S}} = \mathbf{S}_W + \alpha\mathbf{X}\mathbf{L}\mathbf{X}^T$
7: Apply the search algorithm in [23] to solve the optimization problem in (9). (The inputs of this algorithm are $\mathbf{S}_b$, $\hat{\mathbf{S}}$, dimensionality of the output feature space ($d$) and the error constant $\varepsilon$)
8: Calculate the new metric $\mathbf{A} = \mathbf{W}^*(\mathbf{W}^*)^T$
```

distance metric which is ignored. Consequently, the dimensionality of the transformed space should be smaller than the input one, i.e., $d < n$, which shows that the dimensionality of the vectors is reduced after the transformation.

The covariance of the point pairs in the similarity and dissimilarity constraints are calculated by (7) and (8) respectively.

$$\mathbf{S}_w = \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in P} (\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^T \tag{7}$$

$$\mathbf{S}_b = \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in N} (\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^T \tag{8}$$

Substitution of (7) and (8) into (6) leads to the following optimization problem:

$$\mathbf{W}^* = \underset{\mathbf{W}^T\mathbf{W}=\mathbf{I}}{\arg\max} \ \frac{\mathrm{tr}(\mathbf{W}^T\mathbf{S}_b\mathbf{W})}{\mathrm{tr}(\mathbf{W}^T(\mathbf{S}_w + \alpha\mathbf{X}\mathbf{L}\mathbf{X}^T)\mathbf{W})} \tag{9}$$

We can solve this optimization problem by applying the iterative search algorithm in [23]. The result is the learned transformation matrix $\mathbf{W}^*$ and the learned distance metric $\mathbf{A} = \mathbf{W}^*\mathbf{W}^{*T}$, which are directly extracted from the training samples. Algorithm 1 summarizes the overall process of the proposed ML method.

In the next section, we develop an unsupervised NADS that employs this ML scheme to justify the effectiveness and efficiency of applying an appropriate metric in a NADS.

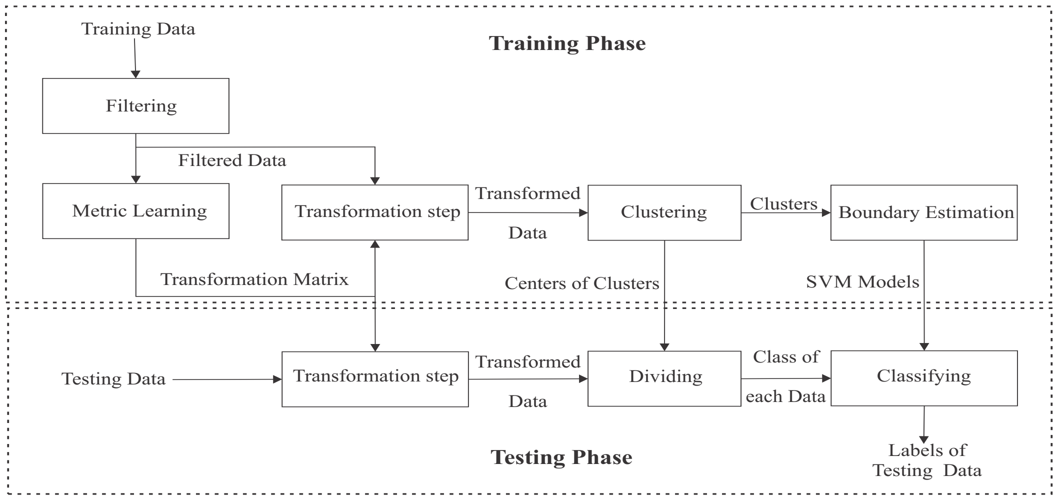


Fig. 2. The block diagram of the proposed unsupervised clustering-based NADS, including the training and the testing phases and the metric learning module

4 Proposed Unsupervised Clustering-based NADS

Fig. 2 depicts the overall schema of the training and testing phases of the proposed unsupervised clustering-based NADS. The training phase has five steps: *Filtering*, *Metric Learning*, *Transformation*, *Clustering* and *Boundary Estimation*.

As other NADSs, the system models the normal traffic behavior and each deviation from this model is considered as an attack. The system first needs a method for selecting the normal connections' feature vectors from the training data which include normal and attack connections. In the *Filtering* step, an unsupervised method is employed to filter out most of the attacks from the training samples. Hence, the output of this step is the normal network connections, which are called filtered data.

The filtered data are the input of the *Metric Learning* step. At this stage, two sets of similarity/disimilarity constraints are extracted from the filtered data as described in section 3.2 and they are used to formulate the optimization problem of (9). The solution of this problem is the desired n-by-d transformation matrix, $\mathbf{W}^*$ as the result of the *Metric Learning* step.

The *Transformation* step maps the n-dimensional filtered data to a smaller d-dimensional output feature space using the linear mapping function, $\mathbf{y}_i = \mathbf{W}^{*T}\mathbf{x}_i$. The *Clustering* step groups the similar filtered data in the same cluster and the dissimilar filtered data in the different clusters. The results are k well-separated clusters of the normal connections. In

Boundary Estimation step, an unsupervised one-class classifier based on boundary method [25] estimates the border of each cluster separately. Then the insides of these borders are considered as the profile of the normal traffic behavior in the proposed clustering-based NADS.

The testing phase consists of three steps: *Transformation*, *Dividing* and *Classifying*. The *Transformation* step employs the learned transformation matrix, $\mathbf{W}^*$, to map testing connections' feature vectors to the output feature space. The *Dividing* step assigns the mapped testing data to their nearest clusters, which are detected in the *Clustering* step of the training phase. Finally, the *Classifying* step classifies the testing data into normal or attacks. If a connection's feature vector locates within the boundary of its nearest cluster, it is regarded as normal, and otherwise it is considered as an attack.

In the following, we adopt a specific implementation of the *Filtering*, *Clustering Pre-Clustering* in the *Metric Learning* step from [26] to develop a NADS, which uses a particular distance metric that is derived from the connections' feature vectors, based on the overall schema of Fig. 2. We will compare our results against [26] in the next section to show the effect of using data specific distance metric in the design of NADSs. We should note that the developed scheme can be used in conjunction with other *Filtering*, *Pre-Clustering*, *Clustering* and *Boundary Estimation* schemes.

In the *Boundary Estimation* step, we employ the one-class SVM classifier which is an unsupervised learning method that finds a hyper-sphere around the samples of each cluster. This hyper-sphere encompasses most of these samples. See [26] for more details on the adopted *Filtering*, *Clustering*, and *Boundary Estimation* processes.

5 Experimental Results

This section explains the experiments were performed to investigate the effects of the proposed ML method on the performance of the clustering-based NADSs and also the NADSs based on some other machine learning schemes.

5.1 Evaluation Metrics

Six performance measures, including accuracy of normal (true negative rate/TNR), accuracy of attack (true positive rate/TPR), false positive rate/FPR, false negative rate/FNR, correctly classified instances/CCI and F-score are evaluated in the following experiments. TNR is the fraction of normal connections correctly classified as normal; TPR is the fraction of attack connections correctly classified as attacks; FPR is the fraction of normal connections incorrectly classified as attacks; FNR is the fraction of attack connections incorrectly classified as normal; CCI is the fraction of correctly classified network connections and F-score is the harmonic mean of two metrics: TPR and Precision where Precision is the fraction of detected attacks, which are correctly classified as attacks.

These measures are used to compare the performance of the proposed ML based NADS against the Euclidean SVM-Clustering Based NADS (ESC-NADS) proposed in [26], which employs the Euclidean distance in its clustering and classification phases. The proposed ML based NADS, which its architecture is shown in Fig. 2, uses the *Metric Learning* and the *Transformation* steps in the training phase, as well as the *Transformation* step in the testing phase in contrast to the ESC-NADS. Both NADSs apply a modification of K-means clustering algorithm in the *Clustering* step and one-class SVM classifier in the *Boundary Estimation* step of the training phase.

5.2 Dataset Description and Data Preparation

The proposed NADS was evaluated over the Kyoto 2006+ dataset [27]. Each network connection in this dataset is specified by 24 features. The first 14 features were directly selected from the most informative features of the KDD cup 99 dataset [28]. The other 10 features were extracted to investigate efficiently what happens on the networks. This dataset has advantages compared to the other datasets like the KDD cup 99 and NSL-KDD. The Kyoto 2006+ dataset was collected more recently from the real computer network connections. That is, it includes more recent attacks and reflects the behavior of normal connections more precisely. In this work, the first 13 features of the Kyoto 2006+ dataset were used to select the training and testing data. See [29] for the list of the features and detailed explanation of them.

In the training phase of the proposed NADS, the attack ratio was set to 1% . As a result, 58294 connections' feature vectors from the traffic of (November. 1-3, 2007) including 582 attacks were randomly selected to prepare the training data. Testing data were also prepared from the traffic of 10 days: December 1, 8, 15 and 22 2007, January 10, 17 and 23 2008, February 9, 16 and 23 2008.

5.3 System Setup

In the *Metric Learning* step, two sets of 5000 pairs were randomly selected from the filtered data using the proposed method in section 3.2 to provide the similarity/dissimilarity constraints. As well as, the performance measure were averaged over the results of employing 10 different sets of randomly selected constraints in the *Metric Learning* step. The dimensionality of the output feature space was fixed to $d = 9$; the regularization parameter was set to $\alpha = 0.2$, and the number of the nearest neighbors in the Laplacian Eigenmap approach was set to $k = 10$.

The Libsvm [30] library was used to implement the one-class SVM classifier in the *Boundary Estimation* step. Radial basis function was chosen as the kernel function of the one-class SVM. The parameters of the kernel function were adaptively selected for each cluster. The ratio of the training samples, which do not locate inside the discovered hyper-sphere of the one-class SVM classifier, is denoted by ν. The performance measures of the proposed NADSs and the ESC-NADS were evaluated for different values of ν in the training step. Both in the training and the testing phases, the features' values of the network connections were normalized to zero mean and unit standard deviation.

5.4 Evaluation and Results

The proposed NADS was evaluated by using different selected features in the *Pre-Clustering* of the *Metric Learning* step. Using all features in the *Pre-Clustering* lead to very small and separated clusters in the *Clustering* step which reduce the accuracy of normal. For this rea-

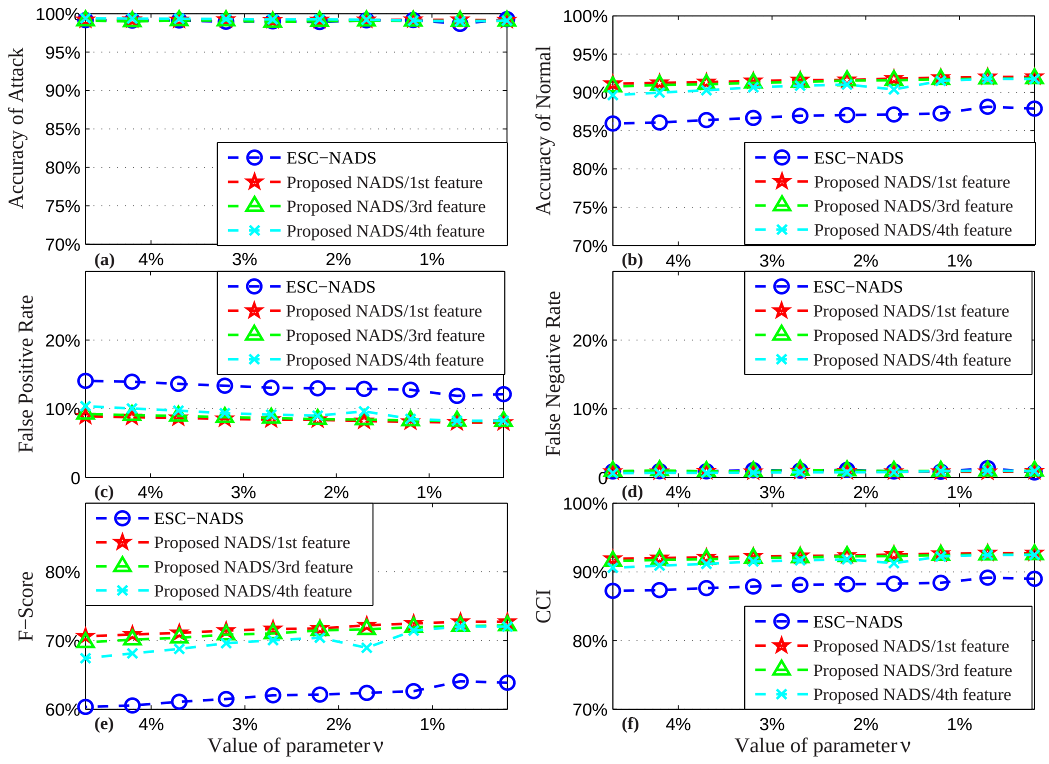


Fig. 3. Performance comparisons of the proposed NADSs and ESC-NADS in terms of a) Accuracy of attack, b) Accuracy of normal, c) False positive rate, d) False negative rate, e)F-Score and f) CCI

son, the *Pre-Clustering* was performed using one of the connections' features. According to the experiments, employing features with the higher variances lead to better results. Therefore, the first (duration), the third (source bytes) and the forth (destination bytes) features, were selected from the 13 features of the training data for *Pre-Clustering* and three different NADSs were trained. In the following, these systems are called the *Proposed NADS/i-th feature* that indicates the *i-th* feature was used in the *Pre-Clustering* step.

Fig. 3 shows the performance measures of the proposed NADSs against the ESC-NADS for different values of parameter ν. As the value of ν is decreased, the normal connections inside the hyper-spheres corresponding to the detected clusters are increased. It leads to the larger sized hyper-spheres and hence they could better represent the normal connections' behaviors. Therefore, by decreasing the value of ν, the TNRs in all NADSs are increased (Fig. 3(b)) and their FPRs are decreased (Fig. 3(c)).

The performance of all NADSs is approximately the same in terms of the accuracy of attack and FNR (Fig. 3(a) and Fig. 3(d)). However, the proposed NADSs, regardless of the selected features in the *Pre-Clustering* step, outperform the ESC-NADS in terms of the accuracy of normal and FPR (Fig. 3(b) and Fig. 3(c)). The reason is that the proposed NADSs apply a distance metric, which could precisely measure the distance of the connections' features. Specifically, the proposed NADSs could better group the similar normal connections. It leads to a more accurate clustering in the *Clustering* step compared to the ESC-NADS. According to Fig. 3(f), higher *TNRs* of the proposed NADSs lead to the greater *CCI* measures compared to the ESC-NADS. Also, lower *FPRs* of the proposed NADSs increase the Precision measures and as a result lead to the greater F-Scores compared to the ESC-NADS as shown in Fig. 3(e). Using the first feature for *Pre- Clustering* improves the performance of the proposed system for greater values of ν.

Fig. 4 illustrates how dependent are the performance measures of the proposed NADSs on the employed side information. Each column of Fig. 4 depicts the obtained experimental results for one of the proposed NADSs. This figure shows the average performance measures of each proposed NADS (over 10 different sets of similarity/dissimilarity constraints) versus the different values of ν and the vertical line segments show the corresponding confidence intervals. We found that the *Proposed NADS/3rd feature* has the minimum sensitivity

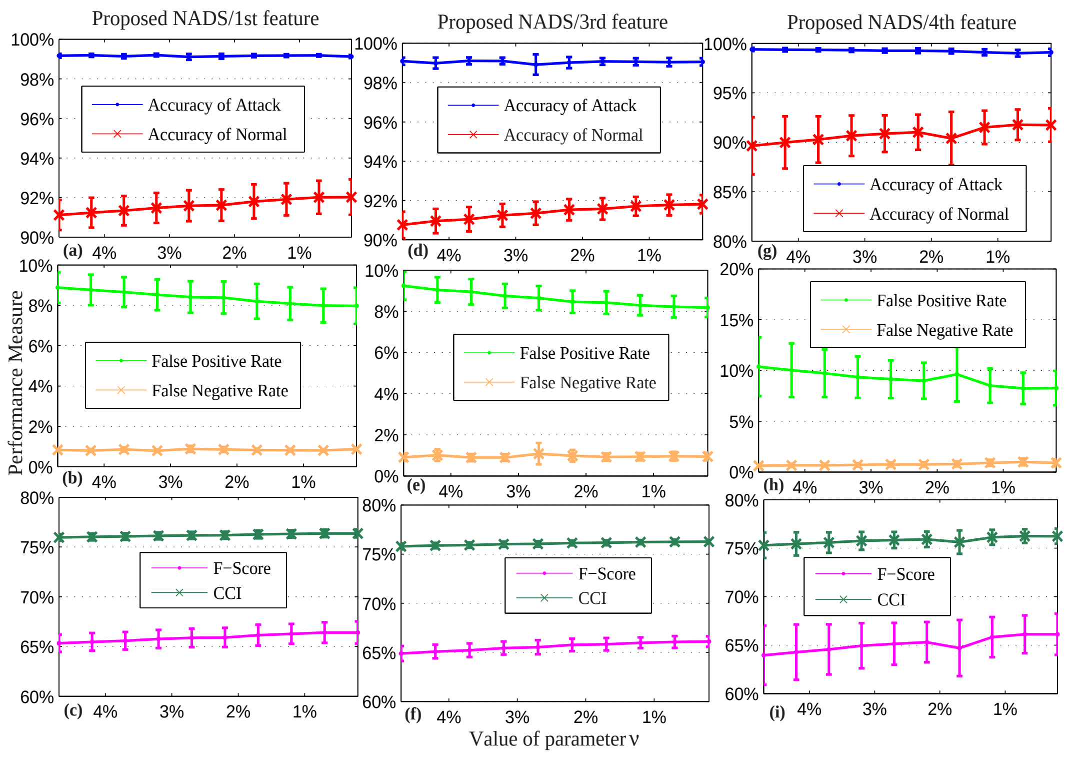


Fig. 4. Sensitivities of the proposed NADSs to the selected similarity/dissimilarity constraints

to the selected side information, Fig. 4(d), Fig. 4(e) and Fig. 4(f). On the other hand, the *Proposed NADS/4th feature* has the maximum sensitivity to the changes of the employed similarity/dissimilarity constraints, Fig. 4(g), Fig. 4(h) and Fig. 4(i).

We also investigated the effect of the locality preserving on the performance of ML based scheme. Fig. 5 depicts the performance measures of the *Proposed NADS/3rd feature* for two different scenarios. In the first one, the locality-preserving property was considered in the proposed ML scheme by setting $\alpha = 0.2$ in the optimization problem of (9). In the second one, the locality-preserving property was ignored by setting $\alpha = 0$. Note that in the second scenario the optimization problem of (9) turned into the problem (3). According to this figure, preserving the local neighborhood information in the proposed ML method significantly improves the performance of the underlying NADS. In fact, when $\alpha \neq 0$, ML optimization proble considers the intrinsic structure of connections' features, which gives some valuable information about the adjacent samples, and leads to a more efficient and reliable distance metric.

In the next experimental study, PCA and LDA were used as the linear feature transformation stage of the ESC-NADS, and the results of the proposed NADS, which applies ML based feature transformation method, were compared against them. ESC+PCA applies PCA to the

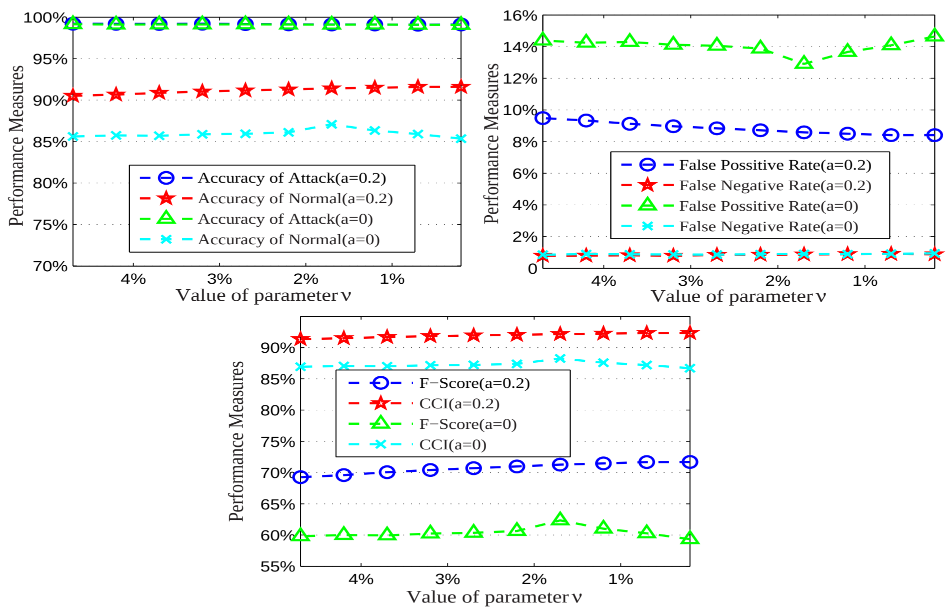


Fig. 5. The performance of the *Proposed NADS/3rd feature* with and without locality-preserving property

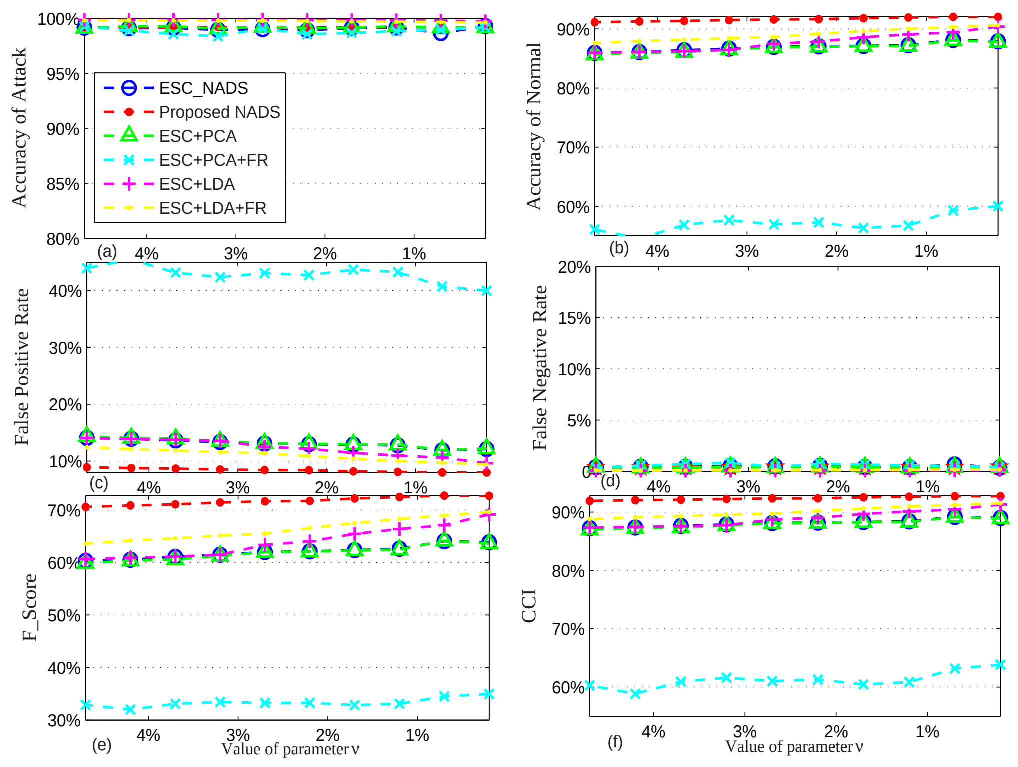


Fig. 6. Performance comparisons of the *Proposed NADS/1st feature*, ESC-NADS, ESC+PCA, ESC+PCA+FR, ESC+LDA and ES+LDA+FR in terms of a) Accuracy of attack, b) Accuracy of normal, c) False positive rate, d) False negative rate, e) F-Score and f) CCI

training data and finds a feature transformation matrix for mapping the data to a new feature space. Next, similar to the ESC-NADS, the *Clustering* and *Boundary Estimation* steps are employed to determine the borders of the normal profile.

In Fig. 6, the performance measures of the *Proposed NADS/1st feature* are compared against ESC-NADS and ESC+PCA for different values of parameter ν. This figure shows that using PCA as the feature transformation does not improve the performance of the ESC-NADS. The reason is that the typical extracted features of the network connections are not highly correlated and consequently applying PCA, which converts a set of correlated features to a set of uncorrelated ones, could not significantly affect the performance of the underlying NADS. Furthermore, applying feature reduction does not help as it is shown in the performance measures of the corresponding NADS entitled ESC+PCA+FR.

LDA was also applied as the feature transformation stage of the *ESC-NADS*. As shown in Fig. 6, the ESC+LDA outperforms the ESC-NADS in terms of accuracy of normal, FPR, F-Score and CCI. The reason is that LDA maps the network connections' feature vectors

Table 1
Performance comparisons of classification-based NADSs with and without applying the proposed ML method

Method	TPR	TNR	FPR	FNR	F-Score	CCI
J48	0.788	0.999	0.001	0.212	0.875	0.9774
J48+ML	**0.805**	0.993	0.007	**0.195**	0.86	0.9745
Naive Bayes	0.099	0.982	0.018	0.901	0.156	0.8957
Naive Bayes+ML	**0.753**	0.969	0.031	**0.247**	**0.74**	**0.9481**
Bayes Net	0.18	0.996	0.004	0.82	0.296	0.916
Bayes Net+ML	**0.726**	0.991	0.009	**0.274**	**0.801**	**0.9646**
Random Forest	0.696	0.991	0.009	0.304	0.784	0.9625
Random Forest+ML	**0.81**	**0.992**	**0.008**	**0.19**	**0.855**	**0.9737**
MLP	0.788	0.98	0.02	0.212	0.797	0.9607
MLP+LP	0.714	**0.991**	**0.009**	0.286	**0.797**	**0.9642**
KNN(k=10)	0.124	0.999	0.001	0.876	0.219	0.9132
KNN(k=10)+ML	**0.998**	0.97	0.03	**0.002**	**0.878**	**0.9729**
ICO	0.704	0.999	0.001	0.296	0.82	0.9697
ICO+ML	**0.791**	**0.999**	**0.001**	**0.209**	**0.878**	**0.9784**

into a new feature space in which the ratio of the between-class and within-class distances of the training samples is maximized. This leads to a more accurate normal profile construction in the new feature space and enhances the performance. However, the *Proposed NADS/1st feature* outperforms ESC+LDA. According to this figure, feature reduction by ESC+LDA+FR leads to the performance improvement of the ESC+LDA. For a fair comparison, the same number of the features were employed in the *Proposed NADS/1st feature* and ESC+LDA+FR. Again, the *Proposed NADS/1st feature* outperforms the ESC+LDA+FR in terms of accuracy of normal, FPR, F-Score and CCI.

The proposed ML method not only can improve the performance of distance-based algorithms, e.g., SVM and K-means clustering techniques, but also feature mapping, which leads to well-separated classes in the output space enhances the performance of the other machine learning techniques. In this experimental study, seven different classification algorithms, including j48 decision tree, Naive Bayes, Bayes Net, Random Forest, Multi-layer Perceptron

(MLP), K-nearest-neighbors (KNN) and Iterative classifier optimizer (ICO) [31] were employed to identify to which of normal or attack classes a testing connection's feature vector belongs. Weka software was used for implementation of these algorithms and the values of all the algorithms' parameters were fixed to the default ones except the number of the nearest neighbors in KNN algorithm, which was fixed to $k = 10$. Their classification performance was evaluated in two different scenarios. In the first scenario, the training and testing data were used without applying a feature transformation method as presented in section 5.2. The resulting models, which were built based on the training data, were evaluated using the testing samples. In the second scenario, to investigate the effect of applying a proper feature transformation method, first a transformation matrix was learned based on the proposed unsupervised ML method, and it was used to map the training and testing data to a new feature space. Then, the models were built and evaluated in the output feature space. Table 1 compares the performance metrics of the NADSs based on these classification methods with and without applying the learned transformation matrix. Note that, the attack ratio of the training samples was set to 1%. As a result, the imbalanced dataset leads to poor "accuracy of attack" measures of these NADSs without applying a transformation matrix. However, employing the feature transformation matrix by the NADSs, leads to more accurate models, which discriminate normal and attack classes well while preserving the local neighborhood information of the connections. The results show that the proposed ML method can improve the accuracy of attack, FNR, F-Score and CCI measures of the underlying NADSs based on these classification methods, as shown in Table 1.

Finally, the performance of ESC-NADS and *Proposed NADS/1st feature* were also evaluated over two other datasets, NSL-KDD and updated version of Kyoto2006+ (Kyoto2015), to verify the effectiveness of the proposed ML scheme. For Kyoto2015 dataset, the attack ratio was set to 1% and 60000 connections' feature vectors were randomly selected from the traffic of November 1-3, 2014 including 600 attacks. Testing data were also prepared from the traffic of 40 days: December 1, 8, 15 and 22 2014, January 10, 17 and 23 2015, February 9, 16 and 23 2015 and March 1-30 2015. Randomly selected set of testing data contained 592016 normal and 65779 attack. System's parameters were identical to what is described

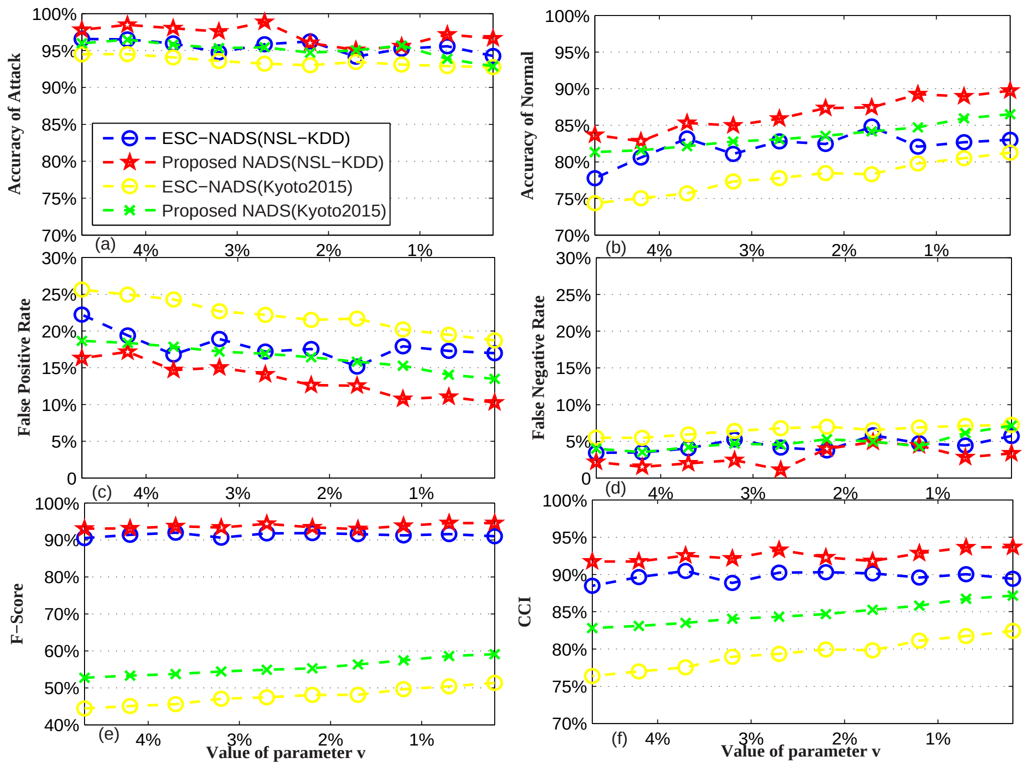


Fig. 7. Performance comparisons of the *Proposed NADS/1st feature* and ESC-NADS over NSL-KDD and Kyoto2015 in terms of a) Accuracy of attack, b) Accuracy of normal, c) False positive rate, d) False negative rate, e) F-Score and f) CCI

by section 5.3. In the case of NSL-KDD, the KDDTrain+ and KDDTest+ were used for training and testing phases respectively. The dimensionality of the output feature space was set to $d = 37$, the regularization parameter was fixed to $\alpha = 0.1$ and the other parameters of the system were identical to section 5.3. Experimental results over these new datasets, which are depicted by Fig. 7, indicate that the *Proposed NADS/1st feature* outperforms the ESC-NADS in terms of all the performance measures, for different values of parameter ν.

6 Conclusions and Future Work

We suggest to learn and use a new distance metric in the NADSs to deal with heterogeneous features of the network connections. An unsupervised ML method has been proposed, which directly derives a proper distance metric from the training connections' feature vectors. The learned feature transformation matrix is a proper feature weighting, which is tuned for the underlying NADS using extracted side information. We present a clustering-based NADS, which deploys the learned distance metric and evaluate its performance over the

Kyoto 2006+ and NSL-KDD datasets. The results show that using a proper distance metric; we can afford the comparison of the heterogenous features, which should be dealt in the NADSs. In the future works, we will investigate other forms of linear ML as well as non-linear ML methods for improving the performance of clustering-based NADSs. We also intend to develop a new architecture to use online ML schemes in the NADSs to consider the effects of the new discovered attacks.

References

[1] Wang L, Jones R (2017) Big Data Analytics for Network Intrusion Detection: A Survey. International Journal of Networks and Communications 7(1):24-31

[2] Du B, Zhang L (2011) Random-Selection-Based Anomaly Detector for Hyperspectral Imagery. IEEE Transaction on Geoscience and Remote Sensing 49(5):1578-1589

[3] Huang D, Mu D, Yang L, Cai X (2018) CoDetect: Financial Fraud Detection with Anomaly Feature Detection. IEEE Access 6: 19161-19174

[4] Yi Y, Wu J, Xu W (2011) Incremental SVM based on Reserved Set for Network Intrusion Detection. Expert Systems with Applications 38(6):7698-7707

[5] Singh S, Silakari S (2009) An Ensemble Approach for Cyber Attack Detection System: A Generic Framework. International Journal of Computer Science and Information Security 6(2):297-302

[6] Olukanmi PO, Twala B (2017) Sensitivity Analysis of an Outlier-aware k-means Clustering algorithm. Pattern Recognition Association of South Africa and Robotics and Mechatronics (PRASA-RobMech), 68-73

[7] Vincent OR, Makinde AS, Salako OS, Oluwafemi OD (2018) A self-adaptive k-means classifier for business incentive in a fashion design environment. Applied Computing and Informatics 14(1): 88-97

[8] Bhuyan MH, Bhattacharyya DK, Kalita JK (2013) Network Anomaly Detection: Methods, Systems and Tools. IEEE Communications Surveys & Tutorials 16(1):303-336

[9] Kulis B (2012) Metric Learning: A Survey. Foundations and Trends in Machine Learning 5(4):287-364

[10] Bohne J, Ying Y, Gentric S, Pontil M (2014) Large Margin Local Metric Learning. In: 13th European Conference. Switzerland, Zurich

[11] Niu G, Dai B, Yamada M, Sugiyama M (2014) Information-theoretic Semi-supervised Metric Learning via Entropy Regularization. Neural Computation 26(8):1717-1762

[12] Greenewald K, Kelly A, Hero A (2016) Nonstationary Distance Metric Learning. arXive preprint arXive:1603.03678

[13] Du B, Zhang L (2014) A Discriminative Metric Learning Based Anomaly Detection Method. IEEE Transaction on Geoscience and Remote Sensing 52(11):6844-6857

[14] Zamani B, Akbari A, Nasersharif B (2014) Evolutionary Combination of Kernels for Nonlinear Feature Transformation. Information Sciences 274:95-107

[15] Duda R, Hart PE, Stork DG (2001) Pattern Classification. John Wiley & Sons, New York

[16] Wang S, Lu J, Gu X, Du H, Yang J (2016) Semi-supervised Linear Discriminant Analysis for Dimension Reduction and Classification. Pattern Recognition, 57: 179-189

[17] Seng JKP, Ang KLM (2017) Big Feature Data Analytics: Split and Combine Linear Discriminant Analysis(SC-LDA) for Integration Toward Decision Making Analytics, IEEE Access 5:14056-14065

[18] Datti R, Verma B (2010) Feature Reduction for Intrusion Detection Using Linear Discriminant Analysis. International Journal on Engineering Science and Technology 2:1072-1078

[19] Bankovic Z, Stepanovic D, Bojanic S, Nieto-Taladriz O (2007) Improving Network Security Using Genetic Algorithm Approach. Computers & Electrical Engineering 33(5-6):438-451

[20] Mohammadi M, Raahemi B, Akbari A, Nassersharif B (2012) New Class-dependent Feature Transformation for Intrusion Detection Systems. Security and Communication Networks 5(12):1296-1311

[21] Baghshah MS, Shouraki SB (2010) Non-linear Metric Learning Using Pairwise Similarity and Dissimilarity Constraints and the Geometrical Structure of Data. Pattern Recognition 43(8):2982-2992

[22] Xing EP, Jordan MI, Russell S, Ng A (2002) Distance Metric Learning, with Application to Clustering with Side-information. Advances in Neural Information Processing Systems 14:505-512

[23] Xiang S, Nie F, Zhang C (2008) Learning a Mahalanobis Distance Metric for Data Clustering and Classification. Pattern Recognition 41(12):3600-3612

[24] Belkin M, Niyogi P (2003) Laplacian Eigenmaps for Dimensionality Reduction and Data Representation. Neural Computation 15(6):1373-1396

[25] Mazhelis O (2006) One-class Classifiers : A Review and Analysis of Suitability in the Context of Mobile-masquerader Detection. South African Computer Journal 36:29-48

[26] Song J, Takakura H, Okabe Y, Nakao K (2013) Toward a More Practical Unsupervised Anomaly Detection System. Information Sciences 231:4-14

[27] Song J, Takakura H, Okabe Y (2009) Traffic Data from Kyoto University's Honeypots. http://www.takakura.com/kyoto_data/

[28] Mukkamala S, Sung AH (2003) Identifying Significant Features for Network Forensic Analysis Using Artificial Intelligent Techniques. International Journal of Digital Evidence 1(4):1-17

[29] Song J, Takakura H, Okabe Y (2006) Description of Kyoto University Benchmark Data. http://www.takakura.com/kyoto_data/BenchmarkData-Description-v5.pdf

[30] Chang C, Lin C (2014) LIBSVM: A Library for Support Vector Machines. https://www.csie.ntu.edu.tw/ cjlin/libsvm/

[31] Nadiammai GV, Hemalatha M (2012) Perspective Analysis of Machine Learning Algorithms for Detecting Network Intrusions. In: Third International Conference on Computing Communication & Networking Technologies. India, Coimbatore