# UNPCC: A Novel Unsupervised Classification Scheme for Network Intrusion Detection

Zongxing Xie, Thiago Quirino, Mei-Ling Shyu
Department of Electrical and Computer Engineering, University of Miami
Coral Gables, FL 33124, USA
z.xie1@umiami.edu, t.quirino@umiami.edu, shyu@miami.edu

Shu-Ching Chen
Distributed Multimedia Information System Laboratory
School of Computing and Information Sciences
Florida International University, Miami, FL, USA
chens@cs.fiu.edu

LiWu Chang
Center for High Assurance Computer Systems
Naval Research Laboratory
Washington, DC, USA
lchang@itd.nrl.navy.mil

## Abstract

*The development of effective classification techniques, particularly unsupervised classification, is important for real-world applications since information about the training data before classification is relatively unknown. In this paper, a novel unsupervised classification algorithm is proposed to meet the increasing demand in the domain of network intrusion detection. Our proposed UNPCC (Unsupervised Principal Component Classifier) algorithm is a multi-class unsupervised classifier with absolutely no requirements for any $a\ priori$ class related data information (e.g., the number of classes and the maximum number of instances belonging to each class), and an inherently natural supervised classification scheme, both which present high detection rates and several operational advantages (e.g., lower training time, lower classification time, lower processing power requirement, and lower memory requirement). Experiments have been conducted with the KDD Cup 99 data and network traffic data simulated from our private network testbed, and the promising results demonstrate that our UNPCC algorithm outperforms several well-known supervised and unsupervised classification algorithms.*

## 1 Introduction

Nowadays, very few effective unsupervised classification methods capable of adapting to various domains have been proposed and developed. Unsupervised classification usually requires a combination of clustering and supervised classification algorithms to be employed, when information is relatively unknown about the training data before classification. It is a process that possesses a contrasting challenge to that of supervised classification, where the main goal is to determine a measure from instances belonging to the same class that is large enough to reject instances belonging to other various classes, while at the same time low enough so as to take into account the variability found among the instances belonging to the same class. However, most of the existing unsupervised classification algorithms, especially those which do not require any known class related information such as the number of classes and the maximum number of instances in each class, suffer from the lack of high classification accuracy [4][5] and broad effectiveness in applications with data sets with different inter-class variability.

Recently, network intrusion detection has become more and more important in order to safeguard the network systems and crucial information being stored and manipulated through the Internet, and several intrusion detection algorithms using advanced machine learning techniques have been developed for this purpose [2][7][9]. Unsupervised classification is particularly useful in detecting previously unobserved attacks in network intrusion detection domain since new attacks on computers and networks can occur any time. Furthermore, unsupervised classification algorithms are usually employed after a failure of a misuse detection classifier [1] to identify an instance as belonging to a known attack type, in order to uncover new types of intrusions. At the same time, a robust unsupervised classification algorithm could possibly eliminate the need for a human analyst in the assignment of labels to unknown attack types by fully automating the labeling process. Moreover, it is important to realize the two main issues associated with pre-label processing in intrusion detection as was concluded by [6], i.e., it can be extremely difficult or impossible to obtain labels, and one can never be completely sure that a set of available labeled examples reflect all possible existing attacks in a real-world application.

In this paper, in an attempt to meet the above increasing demands, a novel unsupervised classification algorithm called UNPCC (Unsupervised Principal Component Classifier) which is based on Principal Component Analysis (PCA) [12] is proposed. Various training and testing data sets composed of KDD Cup 99 [3] and the simulated real-network traffic generated through the Relative Assumption Model [16] were used to evaluate the performance of UNPCC and its inherently natural supervised classification stage, in comparison to the other well-known unsupervised and supervised classification algorithms. The experimental results demonstrate that (i) the supervised classification stage of the UNPCC algorithm performs better than C4.5 decision tree, Decision Table (DT), NN, and KNN, and (ii) the unsupervised classification performance of the UNPCC algorithm outperforms K-Means, Expectation Maximization (EM), FarthestFirst, Cobweb, and Density Based Clustering with K-Means.

This paper is organized as follows. The motivations and concepts associated with the proposed unsupervised classification algorithm are presented in Section 2. Section 3 introduces the proposed UNPCC algorithm. Section 4 narrates the experimental setup and illustrates the experimental results. Finally, in Section 5, we conclude our paper.

## 2 Motivation

Principal Component Analysis (PCA), the core of the PCC (Principal Component Classifier) anomaly detection algorithm that we developed [10][11], is employed to reduce the dimension of a training data set, allowing for further data analysis and easier exploration of the statistical information present in the data set. Principal components are particular linear combinations of the original variables with two important properties:

1. The first principal component is the vector pointing in the direction of the largest correlation of the original features and the second principal component is the vector pointing in the direction of the second largest correlation of the original features, and so on.

2. The total variation represented by the the principal components is equal to the total variation present in all original variables.

The distribution of the original data features contributes to the direction of the principal components, so it is necessary in PCA to use all original features to generate principal components which account for all the variability of the original training data set. Once the principal component set is found, only a few principal components and their related eigen-values and scores [12] need to be stored and employed to represent a large percentage of the variance in the original training data. Moveover, suitable selection and combination of principal components with certain specific features is required to represent specific characteristics of the training data set. PCC is a good example to demonstrate the uses of the major components representing the overall structure of the training data set and the minor components capturing observations that do not conform to the training data correlation structure. In PCC, two distance measures $C1$ and $C2$ based on the major and minor components respectively are defined as classification threshold values and produced promising classification accuracy with low false alarm rates.

Inspired by the concept of similarity representation of the training data set in PCC, which is assumed to be composed of a fixed class with a known label, and also by the promising performance of the combination of the distance measures $C1$ and $C2$ in PCC, it is highly possible that suitable selection and combination of certain principal components could be used to represent the dissimilarity present in a training data set which is composed of several classes of data sets with unknown labels. For this purpose, a distance measure, denoted as $C0$, capable of partitioning various classes of data sets based fully on their dissimilarity features in the transformed principal component space is defined.

Our main idea is validated through a series of operations in the transformed principal component space:

- Plotting and observation of principal component score arrays of KDD Cup 1999 data [3] through MATLAB, including various combination of four kinds of network attack types: teardrop, back, neptune, and smurf;

- Selection of certain principal components with high variability through graphical analysis;

- Definition of the dissimilarity measure $C0$ of an instance using Equation (1), where $i$ indexes the selected representative principal components, and $\lambda_i$ and $y_i$ correspond to the eigenvalue of the $i^{th}$ principal component and the score of the $i^{th}$ feature in the principal component space respectively. The term "score" means the projection of an instance onto the eigenspace composed of all principal components acquired from the training data set.

$$C0 = \sum(\frac{y_i^2}{\lambda_i}) \quad (1)$$

- Plotting, in MATLAB, the distribution of the $C0$ measure, given by the vector ***C0*** = $\left\{C0^{(k)}\right\}$, where $k = 1, 2, \ldots, N$, and $N$ is the number of unlabeled instances in a training data set.

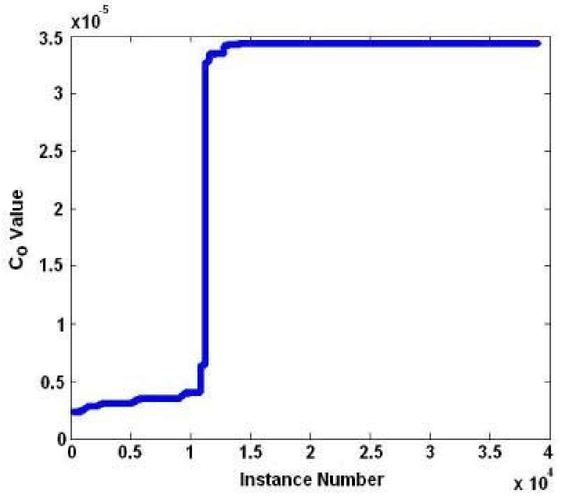

**Figure 1. Sorted $C0$ array of mixed two attacks (neptune and smurf)**

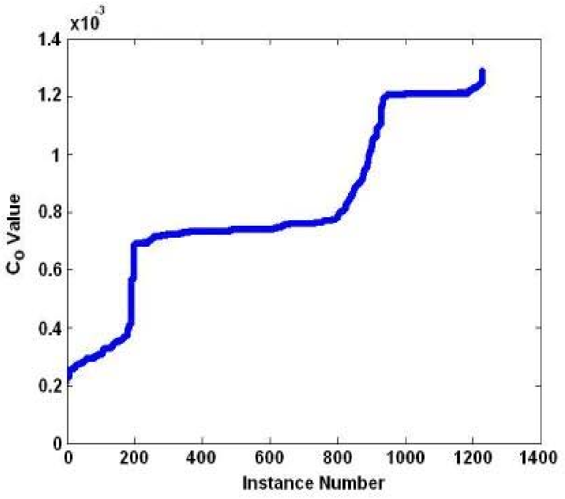

**Figure 2. Sorted $C0$ array of mixed three attacks (teardrop, neptune, and back)**

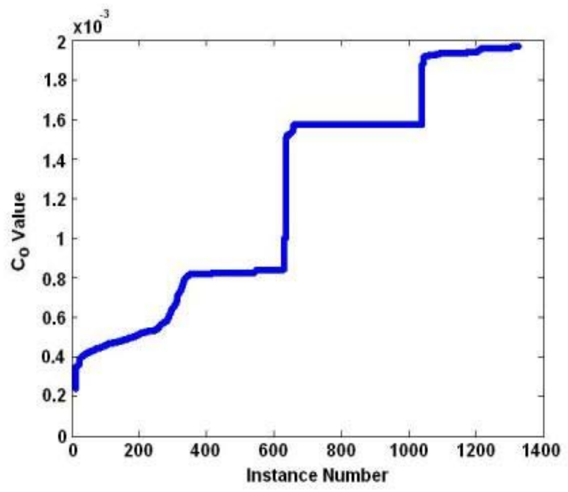

**Figure 3. Sorted $C0$ array of mixed four attacks (neptune, teardrop, back, and smurf)**

Figures 1, 2, and 3 correspond to the plots of the $C0$ distributions of the training data sets with two, three, and four different attack classes, respectively. An interesting observation from these three figures is that each of the classes composing the training data set matches a distinct zone of fracture which is approximately horizontal in the axis of the $C0$ distance measure in the principal component space. That is, these figures exhibit distinguishable differences among the $C0$ values of different classes. In Figure 1, the steep slope separating the two nearly horizontal lines depicts the large differences between the $C0$ scores of the instances belonging to the two different classes. The same pattern can also be observed in Figure 2 and Figure 3, where a steep slope separates nearly horizontal sections representing the scores of instances belonging to the same class type. Therefore, it can be concluded from these results that, theoretically, the distinct $C0$ segments, which can be categorized by suitable $C0$ measure threshold values, can be employed to cluster the related class instances into nearly horizontal subsections of $C0$ values. In addition, the distinct $C0$ threshold values acquired from the unsupervised classification process, which designate different class types, can be employed in a supervised classification scheme. That is, if the $C0$ score value calculated for a certain testing instance falls within one of the distinct $C0$ segments categorized by a certain selected $C0$ threshold value, then it can be concluded that the testing instance in question belongs to the corresponding clustered class. All these characteristics make the proposed classification scheme a complete unsupervised classification method consisting of a natural combination of clustering and supervised classification algorithms.

Motivated by these observations, a novel unsupervised classification scheme is proposed, leaving two challenges to be studied: (i) How to identify and select the representative principal components in an automated manner using the statistical information present in the data set, thus replacing the manual effort of the application of graphical analysis? and (ii) How to automatically identify suitable $C0$ threshold values to represent different classes, a step which is vital to both the clustering and supervised classification phases?

## 3 The Proposed UNPCC Algorithm

Based on the previously presented theoretical analysis and graphical presentation of Figures 1, 2, and 3, the proposed UNPCC algorithm is defined as an ordered combination of the following processes:

1. Normalize the data in the training data set with multiple classes to minimize the issue of scaling features across multivariate data.

2. Apply PCA to the whole training data set to obtain the principal components, their eigenvalues, and data instance scores as was done in [10][11]. Define **Y**= $\{\mathbf{y}_{ij}\}$, $i = 1, 2, \ldots, p$ and $j = 1, 2, \ldots, N$, as the training data score matrix. **Y** is a $p \times N$-dimensional normalized projection of the unlabeled training data matrix onto the $p$-dimensional eigenspace, where the term "score matrix" means the projection of each of the $N$ unlabeled training instances onto the $p$ principal components attained from the training data set through PCA [10][11].

3. Automatically select the principal components which effectively capture the dissimilarities among of various classes in the training data set through a threshold function. This function is based on the standard deviation values of the features in the principal component space to reflect the dissimilarity degree of a score array. Assume that $\boldsymbol{SCORE}_i = (\mathbf{y}_{i1}, \mathbf{y}_{i2}, \cdots, \mathbf{y}_{iN})$, $i$=1,2,. . ., $p$, is the score row vector corresponding to the $i^{th}$ feature in the eigenspace. First, we refine the number of score row vectors by selecting those which satisfy Equation (2) and discarding all others, in order to prevent the impact of score arrays possessing very low standard deviation values, which possibly correspond to very low eigenvalues and their corresponding score row vectors generated by the PCA projection:

$$STD(\boldsymbol{SCORE}_v) > \phi, \text{where} \quad (2)$$

   - $\phi$ is an adjustable coefficient whose value is set to 0.01 as the default value, based on our empirical studies;
   - $STD(\boldsymbol{SCORE}_v)$ is the standard deviation of the score row vector satisfying the refinement equation and corresponding to the the $(v)^{th}$ principal component; and
   - $v \in \mathbf{V}$ is defined as the refined principal component space containing all score row vectors satisfying Equation (2).

   After that, the principal components whose respective score row vectors satisfy the selection function defined in Equation (3) are selected.

$$STD(\boldsymbol{SCORE}_{\mathrm{j}}) > \mathrm{a} \times (\mathrm{Mean}_{\mathrm{STD}}(\boldsymbol{SCORE}_v)) \quad (3)$$

   where:

   - $a$ is the weighting coefficient. Its value is set to 3 in all the experiments based on our empirical studies;
   - $Mean_{STD}(\boldsymbol{SCORE}_v)$ is the average value of all the standard deviation values of the score arrays in the refined principal component space **V**;
   - $STD(\boldsymbol{SCORE}_{\mathrm{j}})$ is the standard deviation of the selected score array satisfying the selection function and corresponding to the $j^{th}$ principal component; and
   - $j \in \mathbf{J}$, which can be defined as the selected principal component space, is composed of those principal components whose corresponding score row vectors satisfy both the refinement and principal component selection functions given by Equation (2) and Equation (3), respectively.

   Accordingly, these selected principal components are then used in calculating the $C0$ dissimilarity measure of all the $N$ unlabeled training instances using Equation (1), effectively yielding a distribution for $C0$ which is entirely based on the statistics of the original training data set and from which suitable $C0$ threshold values can be acquired to categorize the different data classes. The distribution of the $C0$ score measures is given by the sorted vector $\boldsymbol{C0}$ for all $N$ unlabeled training instances.

4. From the $\boldsymbol{C0}$ vector, determine the $C0$ values to be used as the thresholds for both clustering and supervised classification purposes, through the proposed Automated-Cluster-Threshold-Discovery procedure.

   Let $\boldsymbol{C0}_k$ be the $k^{th}$ sorted $C0$ value of the $\boldsymbol{C0}$ distribution of unlabeled training instances, $\boldsymbol{H}_i$ be the expected threshold value of the $i^{th}$ identified class, $N$ be the total number of the unlabeled training data instances, and $Initial$ be a transition variable used to change the selection condition. Also, let $m$ be the index of the current $C0$ value ($C0 \in \boldsymbol{C0}$) in the analysis by the algorithm, $T$ be the total number of the thresholds generated for the $T$ classes identified, and $\boldsymbol{C0}_{m_0}$ and $\boldsymbol{C0}_{m_{max}}$ be the $0.5\%^{th}$ and $99.5\%^{th}$ percentiles of the sorted distribution vector. Here, $m_0$ and $m_{max}$ are indexes corresponding to the nearest integers to $0.005 \times N$ and $0.995 \times N$, respectively, and are used to filter some extreme values from both ends of the $\boldsymbol{C0}$ distribution vector. This filtering step is necessary due to the fact that most of the data sets inevitably contain a few unusual observations which may impact the ability of finding meaningful thresholds in the proposed approach. The Automated-Cluster-Threshold-Discovery procedure for automatically determining the threshold values is given below.

```
Automated-Cluster-Threshold-Discovery
$Initial = \boldsymbol{C0}_{m_0}$;
$\boldsymbol{H}_1 = Initial$;
$i = 2$;
for $(m = 1; m < m_{max}; m++)$ {
    if $(\frac{\boldsymbol{C0}_m - Initial}{Initial} > 1)$ {
        $Initial = \boldsymbol{C0}_{m+m_0}$;
        $\boldsymbol{H}_i = Initial$;
        $i = i + 1$;
    }
}
$T = i - 1$;
```

The threshold values yielded in vector $\boldsymbol{H}$ by the procedure above can be used for two independent tasks: $(i)$ clustering the training data into the $T$ different classes identified, and $(ii)$ supervised classification of unlabeled testing instances, without any additional classifier training requirements. Please note that this clustering/classification scheme does not require any *a priori* knowledge of class related information such as the number of classes in the training data set or the maximum number of instances in each class.

Let us focus on the supervised classification task that naturally rises via the comparison of the $C0$ measure of an incoming testing instance, which is given by Equation (1), with the $T$ different threshold values in vector $\boldsymbol{H}$ that were obtained through the Automated-Cluster-Threshold-Discovery procedure. As was previously elaborated and illustrated in Figures 1, 2, and 3, the values in vector $\boldsymbol{C0}$ are sorted in ascending order. As a result, vector $\boldsymbol{H}$ is inherently sorted in ascending order of values because the Automated-Cluster-Threshold-Discovery procedure traverses vector $\boldsymbol{C0}$ in a linear fashion. With these concepts in mind, let us define the column vector $\boldsymbol{Y'}$=$(\mathbf{y}'_1, \mathbf{y}'_2, \ldots, \mathbf{y}'_p)'$ as the normalized projection of a $p$-dimensional testing instance, normalized using parameters such as the mean feature value and the feature standard deviation value obtained previously from the training data set matrix [10][11], onto the $p$-dimensional eigenspace. The class $i$ of the testing instance will be identified through a linear search through vector $\boldsymbol{H}$, starting at index $i = 1$ and increasing toward the maximum value of $i = T$ corresponding to the threshold of the last cluster, until the last element $\boldsymbol{H}_i$ is found which satisfies Equation (4) below:

$$\sum_{j \in \mathbf{J}} \left(\frac{(\mathbf{y}'_j)^2}{\lambda_j}\right) > \boldsymbol{H}_i \tag{4}$$

where $\mathbf{y}'_j$ is the $j^{th}$ row (or eigenspace feature) of $\boldsymbol{Y'}$, and $\boldsymbol{H}_i$ is the last ordered element in the threshold vector $\boldsymbol{H}$ whose value is less than the testing instance's computed $C0$ measure. In other words, class $i$ corresponds to the last threshold value $\boldsymbol{H}_i$ which is less than the $C0$ measure of the testing instance.

## 4 Experiments

The experiments are conducted in two parts, corresponding to the two phases of an unsupervised classification algorithm. First, for the clustering phase (the core phase of UNPCC), the UNPCC algorithm is compared to different methods such as K-Means, Expectation Maximization (EM), Farthest First (FF), Cobweb, and Density Based Clustering with K-Means. Second, for the supervised classification phase, the UNPCC algorithm is evaluated against various supervised classification algorithms such as the C4.5 decision tree, Decision Table (DT), NN, and KNN (K=5).

### 4.1 Experimental Setup

KDD Cup 1999 [3] data set and three different types of network attacks (namely, $attack1$, $attack2$, and $attack3$) generated in our network test-bed [16] through the application of the Relative Assumption Model algorithm [16] with different parameter values, are employed to assess the performance of the proposed unsupervised scheme. Four groups of data sets with variable numbers of class types are used in the experiments to evaluate both the performance of the unsupervised clustering scheme and the performance of the supervised classification scheme in comparison to some existing methods in the Waikato Environment for Knowledge Analysis (WEKA) package [13].

The four groups of the training and testing data sets are as follows.

1. Group 1: Two types of network attacks from the KDD data set, namely back (441 training and 111 testing in-

stances) and teardrop (194 training and 77 testing instances) for the two-class case.

2. Group 2: Three types of network attacks from the KDD data set, namely back (441 training and 111 testing instances), smurf (400 training and 20000 testing instances), and neptune (300 training and 10000 testing instances) for the three-class case.

3. Group 3: Four types of network attacks, namely neptune (300 training and 10000 testing instances), teardrop (194 training and 77 testing instances), back (441 training and 111 testing instances), and smurf (400 training and 20000 testing instances) for the four-class case.

4. Group 4: Three types of network attacks generated through our network testbed for the three-class case: 1187 $attack1$ abnormal network connections (300 for training and the remaining ones for testing), 1093 $attack2$ abnormal network connections (300 for training and the remaining ones for testing), and 441 $attack3$ network connections (300 for training and the remaining ones for testing).

### 4.2 Selected Algorithms in Comparison

In our experiments, various clustering algorithms [8][14][15] in WEKA [13] are selected in the performance comparison. Some of them are further discussed as follows.

- Expectation Maximization (EM): This algorithm finds the maximum likelihood estimate of the parameters of probabilistic models. In WEKA, the EM algorithm utilizes a finite Gaussian mixture model to learn the trends present in the training data sets. Some of the algorithm's drawbacks are its assumptions that all attributes are independent and normally distributed and its high processing power requirements, which makes it poorly suited for real-time applications with large data sets.

- K-Means: This is a simple iterative algorithm. It randomly chooses the initial locations of k-clusters and assigns instances to the clusters based on their proximity to the center of each cluster. Next, new k-cluster means are calculated using the attribute values of the instances belonging to each cluster. Finally, the process is repeated until no more instances need to be re-assigned to a different cluster. This algorithm has many drawbacks such as its assumption that attributes are independent and normally distributed, it requires the number of clusters to be specified manually, and its training time can be very high for a large number of training instances.

- Cobweb: In Cobweb, the clustering phase is performed on an instance by instance basis, updating the clustering instance by instances. The end result is a tree whose leaves represent the training instances, nodes represent the clusters and sub-clusters, and root node represents the whole data set. The major drawback of the algorithm is the impact that the order which training instances are read as the input to the algorithm may have on the formation of the tree.

- Farthest First (FF): Similarly to K-Means, this algorithm requires the number of clusters to be specified.

- Density Based Clustering with KNN algorithm (KNNDB): This method produces a probability distribution for each testing instance which estimates the membership of the instance in each of the clusters estimated by the K-Means algorithm.

### 4.3 Experimental Results

Table 1 compares of the overall clustering accuracy of the UNPCC, K-Means, Expectation Maximization (EM), Cobweb, Farthest First (FF), and KNN Density Based (KNNDB) algorithms when each of the training and testing data groups is employed. Table 1 indicates that our proposed UNPCC algorithm outperforms all the other unsupervised classification methods, in addition to maintaining an accuracy rate of over 90% in the experiments with all four groups of training data sets. Another observation that can be attained from the experimental results shown in Table 1 is the fact that the classification accuracy of some methods vary significantly among the data groups employed. For example, the accuracy of the K-Means algorithm ranges from 79.16% to one single occurrence of 100%, and the accuracy of the EM algorithm ranges from 56.06% to 78.01%. These results indicate that the predicative models generated do not always work well for various types of training data sets used in the experiments. On the other hand, our proposed UNPCC algorithm maintains a high accuracy of over 90% for all the data groups, indicating that the predictive models work well for all these different types of data sets.

Furthermore, it has been observed that UNPCC has lower training time, lower classification time, and less memory and storage requirements to process and store the components required by the classification stage than all the other algorithms. Particularly, the instance-based methods such as K-Means and FF require large memory and storage when hundreds of training data set instances are involved.

In order to assess the performance of the supervised classification stage without the cascading error effects of the previous unsupervised clustering stage, the class label set used to assign labels to the instances in the different clusters and generated by the UNPCC method (which has been

**Table 1. Overall clustering accuracy (%) comparison among UNPCC, K-Means, EM, Cobweb, FF and KNNDB (K=5) with each group of data**

| Accuracy | Group 1 | Group 2 | Group 3 | Group 4 |
|---|---|---|---|---|
| **UNPCC** | 100% | 100% | 92.81% | 90.60% |
| **K-Means** | 100% | 92.44% | 92.47% | 79.16% |
| **EM** | 56.06% | 78.01% | 67.10% | 73.87% |
| **Cobweb** | 24.88% | 45.48% | 34.90% | 22.09% |
| **FF** | 99.68% | 99.80% | 81.57% | 89.85% |
| **KNNDB** | 96.53% | 90.38% | 64.55% | 83.45% |

proven to possess the highest clustering accuracy in Table 1) is used in the experiments to compare with the other supervised classification algorithms in order to set the same initial conditions in the other algorithms as those in the UNPCC method.

Table 2 shows a comparison of the overall supervised classification accuracy of the UNPCC, C4.5, Decision Table (DT), NN, and KNN (K=5) algorithms for each group of data. It can be clearly concluded from the presented results that the UNPCC algorithm always outperforms the other methods in the supervised classification stage, even when all the algorithms use the same high accuracy clustering method. In addition, it has been observed that UNPCC has a lower classification time than all the other algorithms in the comparison.

**Table 2. Overall supervised classification accuracy (%) comparison among UNPCC, C4.5, DT, NN and KNN (K=5) with each group data**

| Accuracy | Group 1 | Group 2 | Group 3 | Group 4 |
|---|---|---|---|---|
| **UNPCC** | 100% | 99.62% | 98.55% | 88.20% |
| **C4.5** | 100% | 99.55% | 86.26% | 87.95% |
| **DT** | 100% | 99.55% | 76.00% | 85.58% |
| **NN** | 100% | 99.55% | 86.26% | 83.30% |
| **KNN** | 99.46% | 95.98% | 92.35% | 76.04% |

Based on the performance comparison results, UNPCC seems to offer an excellent lightweight and highly accurate solution to the domain of unsupervised classification. In comparison with all the unsupervised classification algorithms in the experiments, our proposed UNPCC algorithm has four advantages:

- An inherently natural combination of clustering and supervised classification: UNPCC can perform either clustering or/and supervised classification. In fact, the supervised classification phase naturally rises after the clustering step, which can save additional processing efforts in terms of additional training efforts.

- Purely clustering or/and unsupervised classification algorithm: The proposed method does not require any $a\ priori$ class related information such as the number of the classes (clusters) and the maximum number of the instances per class.

- Lightweight characteristics: UNPCC does not require training instances to be stored for use in the classification phase. Instead, only the principal components, eigen values, and threshold values are stored for the classification phase. This amount of data is usually far less than hundreds of training data instances, which makes it a lightweight algorithm and thus is feasible to be employed in computers with less processing power for most of the real-world applications.

- Fast computation: Experimental results have demonstrated that UNPCC requires significantly less predictive model training time and testing time than all the algorithms in the performance comparison experiments, under the same operational conditions. This is due to the fact that the most costly computations required for clustering and supervised classification are only based on the single threshold value $C0$, eigen-values, and the score matrix, once the principal components are generated. This advantage makes it particularly useful and suitable to be used in real-time demanding applications.

## 5 Conclusion

In the network intrusion detection application domain, there are great challenges and increasing demands coexisting in the unsupervised classification techniques. In this paper, we propose a novel unsupervised classification algorithm called UNPCC with promising experimental results for network intrusion detection. The proposed algorithm is based on the concepts of robust PCC and uses an automated method to select representative principal components from the data set to generate a single classification measure $C0$. It is a multi-class unsupervised classification algorithm with an inherently natural supervised classification stage, which both present high classification accuracy as the experimental results clearly indicate, and various operational benefits such as lower memory and processing power requirements than the other supervised classification algorithms used in the experiments. These are achieved due to the fact that only very little information about the principal components, eigen values, and threshold values have to be stored for the proper execution of the classification phase. Furthermore, UNPCC was observed to perform significantly faster than algorithms such as K-Means and Expectation Maximization (EM) which are based on iterative processes potentially requiring prohibitive training time and testing time. Various training and testing data sets are employed to evaluate the performance of the UNPCC algorithm and its inherently natural supervised classification stage, in comparison to some existing well-known unsupervised and supervised classification algorithms. As the experimental results have shown, the UNPCC algorithm outperforms all the other supervised and unsupervised methods in the performance comparison experiments, maintaining high classification accuracies throughout all the experiments with the four groups of training and testing data sets. From all the experimental results, it can also be clearly shown that the operational benefits of the proposed novel UNPCC algorithm make it a suitable algorithm for real-time applications.

## 6 Acknowledgments

For Mei-Ling Shyu, this research was supported in part by NSF ITR (Medium) IIS-0325260. For Shu-Ching Chen, this research was supported in part by NSF EIA-0220562 and NSF HRD-0317692.

## References

[1] D. Barbara, N. Wu, and S. Jajodia. Detecting novel network intrusions using Bayes estimator. In *First SIAM Conference on Data Mining*, Chicago, IL, USA, April 2001.

[2] A. Ghosh, A. Schwartzbard, and M. Schatz. Learning program behavior profiles for intrusion detection. In *The 1st USENIX Workshop on Intrusion Detection and Network Monitoring*, pages 51–62, Santa Clara, USA, 1999.

[3] KDD. KDD Cup 1999 Data. *http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html*, 1999.

[4] R. Lark. A reappraisal of unsupervised classification. I Correspondence between spectral and conceptual classes. *International Journal of Remote Sensing*, 16:1425–1443, 1995.

[5] R. Lark. A reappraisal of unsupervised classification. II Optimal adjustment of the map legend and a neighbourhood approach for mapping legend units. *International Journal of Remote Sensing*, 16:1445–1460, 1995.

[6] P. Laskov, P. Dussel, C. Schafer, and K. Rieck. Learning intrusion detection: supervised or unsupervised? In *ICIAP*, Cagliari, ITALY, October 2005.

[7] W. Leea, S. Stolfo, and K. Mok. A data mining framework for building intrusion detection models. In *IEEE Symposium on Security and Privacy*, pages 120–132, Santa Clara, USA, 1999.

[8] MNSU. The EM algorithm for unsupervised clustering. *http://grb.mnsu.edu/grbts/doc/manual/Expectation-Maximization-EM.html*.

[9] S. Mukkamala, G. Janoski, and A. Sung. Intrusion detection using neural networks and support vector machines. *IEEE Internation Joint Conference on Neural Networks*, pages 1702–1707, 2002.

[10] M.-L. Shyu, S.-C. Chen, K. Sarinnapakorn, and L. Chang. A novel anomaly detection scheme based on principal component classifier. In *IEEE Foundations and New Directions of Data Mining Workshop, in conjunction with ICDM'03*, pages 171–179, Melbourne, Florida, USA, November 2003.

[11] M.-L. Shyu, S.-C. Chen, K. Sarinnapakorn, and L. Chang. Principal component-based anomaly detection scheme. In *Foundations and Novel Approaches in Data Mining*, volume 9, pages 311–329. T.Y. Lin, S. Ohsuga, C.J. Liau, and X. Hu, editors, Springer-Verlag, 2006.

[12] L. I. Smith. A tutorial on principal components analysis. *http://www.snl.salk.edu/ shlens/pub/notes/pca.pdf*, February 2002.

[13] Weka. *http://www.cs.waikato.ac.nz/ml/weka/*.

[14] Weka. Class DensityBasedClusterer. *http://www.dbs.informatik.uni-muenchen.de/∼zimek/diplomathesis/implementations/EHNDs/doc/weka/clusterers/DensityBasedClusterer.html*.

[15] Wikipedia. Expectation-Maximization algorithm. *http://en.wikipedia.org/wiki/Expectation-maximization-algorithm*.

[16] Z. Xie, T. Quirino, M.-L. Shyu, S.-C. Chen, and L. Chang. A distributed agent-based approach to intrusion detection using the lightweight pcc anomaly detection classier. In *IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC2006)*, pages 446–453, Taichung, Taiwan, R.O.C., June 2006.