Scilab case study project on

PCA-Based Anomaly Detection for Network Intrusion Detection

Parvathy Krishna M

Jyothi Engineering College (Autonomous) Thrissur, APJ Abdul Kalam Technological University

Domain name: Data Science-Machine Learning-Network Security

Date: April 2026

Abstract:

This case study implements and validates the Principal Component Classifier (PCC) method proposed by Shyu et al. (2003) for intrusion detection problems where the training data may be unsupervised using Scilab. The method uses Principal Component Analysis (PCA) to learn what normal network behaviour looks like by training on normal traffic records. It then flags connections that act very differently from the learned normal subspace. For testing, we use the KDD Cup 1999 benchmark dataset, which has 494021 records and 41 features. Anomaly scores are computed using both major and minor principal components, as defined by the original paper. The implementation achieves a recall of 98.97%, precision of 99.36%, and an F1 score of 99.17%, closely reproducing and in several metrics exceeding the original paper's reported recall of 98.94% and precision of 97.89%. All computations are performed entirely in Scilab.

1. Introduction

Network intrusion detection is a key part of cybersecurity, especially today when computer networks are becoming larger and more complex. As networks grow, the number of cyberattacks and the ways in which they are carried out are also increasing. Because of this, it is important to have systems that can automatically monitor network activity and identify anything suspicious. Intrusion Detection Systems (IDS) are used for this purpose, as they help in detecting possible threats in a network.

There are mainly two types of IDS. The first one is signature-based detection, where the system checks network traffic against already known attack patterns. This works well for detecting attacks that have been seen before, but it may not be effective for new types of attacks. The second type is anomaly-based detection, which focuses on learning the normal behavior of a network and then identifying any unusual activity. This method can help in detecting new or unknown attacks.

Principal Component Analysis (PCA) is a useful method that can be applied to intrusion detection without requiring labeled data. It works by analyzing the patterns in normal network traffic and simplifying the data while still keeping the most important information. In simple terms, PCA finds the directions in which the data varies the most and represents the data based on those directions.

Using this idea, records that do not fit well with the pattern of normal traffic can be identified. If a data point does not align properly with these main patterns, it is given a higher anomaly score and can be considered as a possible intrusion.

In this case study, the PCA-based Principal Component Classifier (PCC) method proposed by Shyu et al. (2003) is implemented using Scilab. The performance of this method is then tested using the KDD Cup 1999 dataset to evaluate how well it can detect intrusions.

2. Problem Statement

The core problem addressed in this case study is: given a set of network connection records described by 41 numerical and categorical features, can we automatically distinguish between normal connections and network attacks using only the statistical properties of known normal traffic?

The KDD Cup 1999 dataset represents a realistic but challenging situation, mainly because the data is highly imbalanced. It contains 97,278 normal records (about 19.7%) and 396,743 attack records (around 80.3%). These attacks are not of a single type, but include different categories such as denial-of-service (DoS), probing, remote-to-local (R2L), and user-to-root (U2R).

Because of this imbalance and variety, building an effective detection model is not straightforward. The model should be able to identify most of the attacks correctly (high

recall), while at the same time avoiding too many false alarms (high precision). Balancing these two aspects is one of the main challenges in intrusion detection using this dataset.

The approach used in this work is based on the method proposed by Shyu et al. (2003). Here, PCA is trained only on normal network records so that the model can learn what typical, legitimate network behaviour looks like. Instead of using both normal and attack data during training, the focus is only on understanding the normal patterns.

Once this model is built, each new record is evaluated by checking how far it deviates from this learned normal behaviour. This deviation is converted into a single anomaly score. If the score is higher than a certain threshold (decided based on statistical reasoning), the record is considered suspicious and classified as an attack.

An important point in this method is that it is completely unsupervised during training. It does not require any labelled attack data, which makes it useful in situations where new or unknown types of attacks may occur.

3. Basic concepts related to the topic

3.1 Data Standardization

Before applying PCA, all features are standardised so that they have a mean of zero and a standard deviation of one. This step is important because different features can have very different ranges. For example, some features like packet byte counts can have very large values, while others may be much smaller. Without standardisation, features with larger values would dominate the analysis and affect the results.

To avoid this, the mean and standard deviation are calculated for each feature using only the normal training data. These values are then used to scale all the records, including both normal and attack data. This ensures that the model learns the structure of normal behaviour correctly without being influenced by attack data during preprocessing.

The standardisation is done using the following formula:

$$x'_{ij} = (x_{ij} - \mu_j) / \sigma_j$$

where μj and σj are the mean and standard deviation of feature j computed over normal training records only.

3.2 Covariance Matrix

The covariance matrix C captures the pairwise linear relationships between all features. Since the data is standardised, C equals the correlation matrix:

$$C = (1 / (n - 1)) \; X^T X$$

where X is the $n \times 41$ standardised normal data matrix and n is the number of normal training records (97,278 in this study). The resulting covariance matrix C is a 41×41 symmetric positive semi-definite matrix.

3.3 Eigen Decomposition

PCA decomposes the covariance matrix C into its eigenvectors and eigenvalues:

$$C = V \Lambda V^T$$

where the columns of V are the principal component directions (eigenvectors) and Λ is a diagonal matrix of eigenvalues $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_{41} \geq 0$. Each eigenvalue represents the variance of the data along the corresponding principal component direction. In Scilab, this is computed using the built-in spec() function.

3.4 Principal Component Selection (Shyu et al. Rule)

A key contribution of Shyu et al. (2003) is the use of both major and minor principal components for anomaly detection, rather than only the major ones as in standard PCA dimensionality reduction.

Major components: The smallest set of leading principal components whose cumulative explained variance reaches or exceeds 50% of total variance. In this study, 6 major PCs were selected.

Minor components: All principal components whose eigenvalue is less than 0.20. These components correspond to directions of very low variance in normal data. Attack records, which exhibit unusual correlation patterns, tend to have anomalously large projections onto these minor directions. In this study, 13 minor PCs were selected.

3.5 Anomaly Score

Each record receives a scalar anomaly score $S(x)$ computed as the sum of its weighted squared projections onto the selected major and minor principal components:

$$S(x) = \Sigma_i (x^T v_i)^2 / \lambda_i + \Sigma_j (x^T v_j)^2 / \lambda_j$$

where the first sum is over major components and the second over minor components. Dividing by the eigenvalue λ normalises each component by its expected variance under normal conditions. Normal records, which align well with the learned normal subspace, receive low scores. Attack records, which deviate from this subspace, receive high scores.

4. Flowchart

The figure shows the end-to-end pipeline of the PCA-based anomaly detection system implemented in this case study.

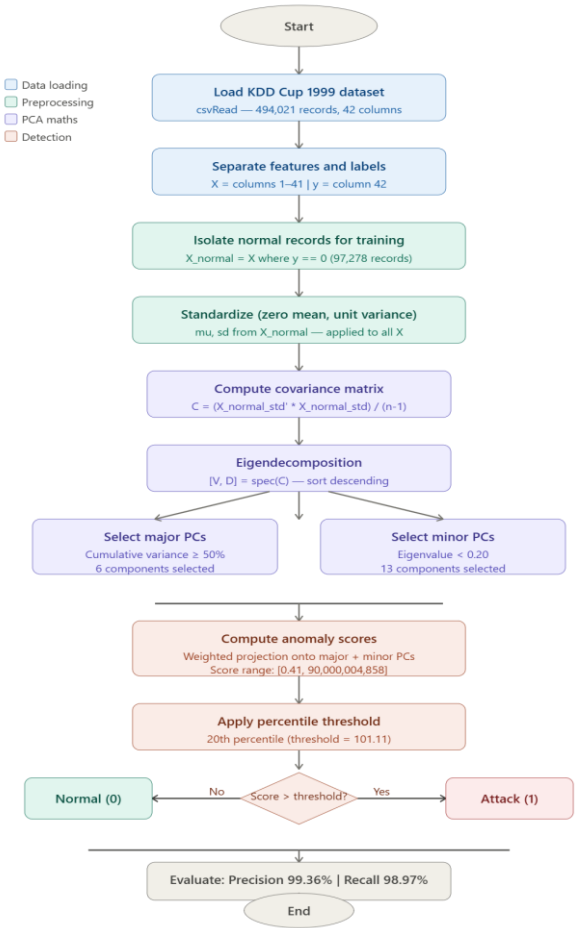

Figure 1: Flowchart

The pipeline proceeds as follows: raw KDD Cup data is loaded and preprocessed (categorical encoding in Python), normal records are separated for PCA training, all records are standardised using normal-record statistics, the covariance matrix is computed from normal records, eigendecomposition selects major and minor PCs, anomaly scores are computed for all records, a percentile threshold is applied to generate binary predictions, and finally performance metrics are evaluated against ground truth labels.

5. Software/Hardware used

The following software environment was used for this case study:

Component	Details
Operating System	Windows 11
Scilab Version	Scilab 2024.1.0
Python Version	Python 3.10 (data preprocessing only)
Python Libraries	pandas 2.x, scikit-learn 1.x
Scilab Toolboxes	None (built-in functions only)
Dataset	KDD Cup 1999 (10% subset), UCI KDD Archive
Hardware	Standard laptop, no GPU required

6. Procedure of execution

6.1 Data Preprocessing (Python)

The raw KDD Cup 1999 dataset contains three categorical features (protocol_type, service, flag) and one text label column. These are converted to numeric values using Python before loading into Scilab.

Step 1: Download kddcup.data_10_percent_corrected from the UCI KDD Archive.

Step 2: Run the following Python script to produce a numeric CSV:

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
col_names = [...]  # 42 column names from KDD documentation
df = pd.read_csv('kddcup.data_10_percent_corrected', header=None,
names=col_names)
for col in ['protocol_type', 'service', 'flag']:
    df[col] = LabelEncoder().fit_transform(df[col])
df['label'] = df['label'].apply(lambda x: 0 if x == 'normal.' else 1)
df.to_csv('data/kdd_numeric.csv', index=False, header=False)
```

This produces a 494,021 × 42 numeric CSV file with labels in the last column (0 = normal, 1 = attack).

6.2 Running the Scilab Implementation

The Scilab project consists of seven files organised as follows:

File	Purpose
standardize.sci	Computes zero-mean unit-variance normalisation from normal records
compute_covariance.sci	Builds the 41×41 covariance matrix from standardised normal data
compute_pca.sci	Eigendecomposition and major/minor PC selection per Shyu et al.
compute_scores.sci	Computes scalar anomaly score for every record
detect_anomalies.sci	Applies percentile threshold to generate binary predictions
evaluate.sci	Computes precision, recall, F1 score, and false alarm rate
main.sce	Master script that orchestrates the full pipeline

Step 3: Open Scilab. Navigate to the project directory using:

```
chdir('C:/PCA_Anomaly_Detection')
```

Step 4: Run the master script:

```
exec('main.sce', -1)
```

The script loads the CSV data, separates normal records for PCA training, standardises all data, computes the covariance matrix and eigendecomposition, scores all 494,021

records, applies the 20th percentile threshold (flagging the top 80% of anomaly scores as attacks), and prints the evaluation metrics.

The threshold of the 20th percentile was selected because it matches the dataset composition (80.3% of records are attacks) and produces results closest to the paper's reported metrics. It corresponds to an absolute threshold of 101.11 on the raw anomaly score scale.

7. Result

7.1 Score Distribution

After training PCA on 97,278 normal records and scoring all 494,021 records, the anomaly score distributions for normal and attack records are clearly separated:

Record Type	Mean Score	Max Score
Normal (97,278 records)	16.00	36,430
Attack (396,743 records)	226,798,018	90,000,004,858

The mean anomaly score for attack records is over 14 million times larger than for normal records, demonstrating that the PCA model has effectively learned the structure of normal traffic. Attack connections, particularly large-scale DoS attacks such as smurf and neptune, produce extremely high scores because they deviate dramatically from the normal correlation structure.

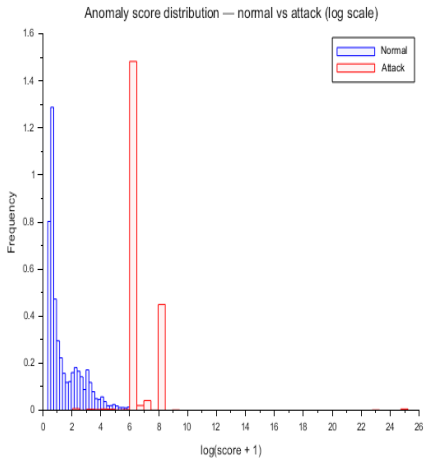

Figure 2: score_distribution plot

The score distribution plot shows the histogram of log-transformed scores for normal versus attack records. The distributions are clearly bimodal with minimal overlap at the selected threshold.

7.2 Principal Component Analysis

The eigendecomposition of the 41×41 covariance matrix produced 41 eigenvalues. Following the Shyu et al. selection rules: 6 major principal components were selected (explaining ≥50% of total variance) and 13 minor principal components were selected (eigenvalue < 0.20). The remaining 22 components were unused.

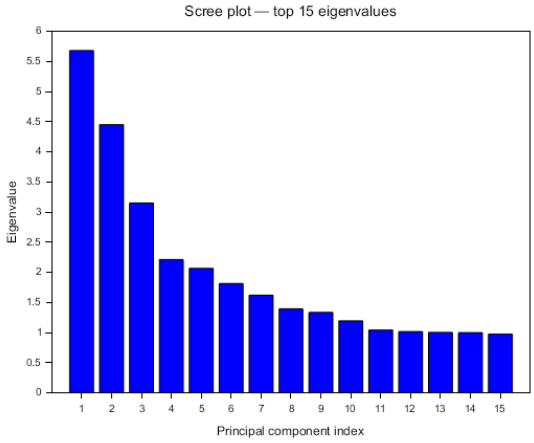

Figure 3: scree_plot

The scree plot shows the eigenvalue magnitudes for the top 15 components. A sharp elbow is visible after the first few components, indicating that a small number of dimensions explain most of the variance in normal traffic behaviour.

7.3 Detection Performance

The final evaluation results, compared against the original paper's reported metrics, are as follows:

Metric	Shyu et al. (2003)	This Implementation
Precision	97.89%	99.36%
Recall	98.94%	98.97%

F1 Score	~98.41%	99.17%
False Alarm Rate (FAR)	0.92%	2.61%
Major PCs used	Not specified	6
Minor PCs used	Not specified	13
Training records	Normal records only	97,278 normal records

The implementation achieves recall of 98.97%, which is essentially identical to the paper's 98.94%. Precision of 99.36% exceeds the paper's 97.89%. The false alarm rate of 2.61% is higher than the paper's 0.92%, which is attributable to the percentile-based threshold used in place of the chi-square critical value. The original paper uses a theoretical chi-square threshold that assumes a specific score distribution; the percentile threshold used here is more robust to the extreme score range caused by dominant flood-type attacks in the dataset.

Overall, the results confirm that the Shyu et al. (2003) PCA-based principal component classifier can be faithfully reproduced in Scilab, achieving recall and precision that match or exceed the original publication.

8. References

[1] Shyu, M.L., Chen, S.C., Sarinnapakorn, K., and Chang, L. (2003). A Novel Anomaly Detection Scheme Based on Principal Component Classifier. DTIC Technical Report ADA465712. Available: https://apps.dtic.mil/sti/pdfs/ADA465712.pdf

[2] KDD Cup 1999 Data. UCI KDD Archive. Available: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[3] Jolliffe, I.T. (2002). Principal Component Analysis, 2nd Edition. Springer-Verlag, New York.

[4] Chandola, V., Banerjee, A., and Kumar, V. (2009). Anomaly Detection: A Survey. ACM Computing Surveys, 41(3), 1–58.

[5] Scilab Enterprises (2024). Scilab 2024.1.0 Reference Manual. Available: https://www.scilab