RIT

Rochester Institute of Technology

CSEC 520/620: Cyber Analytics & Machine Learning

Assignment 2 - Anomaly Detection

Due Date - September 30, 2021 11:59 PM

Purpose

The purpose of this assignment is to learn about two clustering algorithms that can be used to perform anomaly detection and act as a simple Network Intrusion Detection System (NIDS).

Description

You and your team will develop two anomaly detectors:

- *k* means, and
- Density-based spatial clustering of applications with noise (DBSCAN).

You are NOT allowed to use any third-party libraries (e.g. scikit-learn, etc.) to implement the clustering algorithms. Other usage of third-party libraries is fine, and it is in-fact recommended that you use libraries to perform some operations such as dimensionality reduction (e.g., PCA, etc.) and distance calculations (e.g., euclidean distance, etc.).

Dataset

Get the dataset from here. Read the readme file to learn about the data and familiarize yourself with the dataset.

Steps

1. Dataset Loading

Each data file can be loaded into a Python 3 program with the numpy library via the load() function.Loading each data file will return a 2D numpy matrix in which each row represents a traffic sample and each column represents a feature. If you are unfamiliar with working with numpy arrays, refer to this quickstart tutorial.

2. Dimensionality Reduction

The data cannot be easily visualized. To solve this issue, you will use Principal Component Analysis (PCA) to project the 41-dimensional data into two dimensions. You are NOT expected to do this manually. Libraries such as *sklearn* provide convenient PCA decomposition functions that you can use to project your data. Only fit the PCA model to your training data. The model should *not* be fitted against unseen samples! Plot a subset of samples from your testing dataset on a 2D scatter plot.

3. Anomaly Identification using Clustering Algorithms

In this step, you will implement the *k*-Means and DBSCAN clustering algorithms. Review supplemental materials provided in the lecture slides for details.

- Key points:
  - *k*-Means: Identify the cluster *centroids* for your *k* clusters.
  - DBSCAN: Identify the *core points* for your clusters.
- These clustering techniques can be modified to perform anomaly identification in the ways explained in the following subsections.

(3.1) *k*-Means

- Use *k*-Means clustering to identify the centroids of the clusters in the normal traffic training set.
- Select a distance threshold value, *t*.
- For each test sample, find the cluster centroid to which the sample is closest using a distance function (e.g. Euclidean distance). If the distance is *less* than your threshold value *t*, then classify the sample as normal. If it is *greater* than your threshold value *t*, then classify the sample as anomalous.

(3.2) DBSCAN

- Use DBSCAN to identify the core-points for the clusters in the normal traffic training set.
- Using the *core points* for each cluster and the *epsilon* value you used to build initial clusters determine whether or not each testing sample. If it belongs to a cluster, then classify the sample as normal. If it is an outlier, then classify the sample as anomalous.

4. Hyperparameter Tuning

For each algorithm you will need to perform some hyperparameter tuning. The critical hyperparameters for each algorithm are:

- *k*-Means
  - Number of *k* clusters to build.
  - Distance threshold value *t*.
- DBSCAN
  - The *epsilon* value used to determine neighbors.
  - The *minimum number of neighbors* to be labeled as a core point.
- The number of dimensions to reduce your features to (if at all) - for **both** of the clustering algorithms.

5. Performance metrics

To analyze the performances of your implemented *cluster algorithms*, you will need to include the following metrics in your code:

- Accuracy = (TP + TN) / (TP + FP + TN + FN)
- TPR = TP / (FN + TP)
- FPR = FP / (TN + FP)
- F1-score = 2TP / (2TP + FP +FN)

Deliverables

1. All source code of your implementations.
   - Document your code! Your code documentation should demonstrate you understand what your code is doing (and that you did not just copy-and-paste from an external source).
2. A `readme` file that should contain:.
   - A clear description of the directions to set up and run your code.
   - If your code requires external libraries, or if not written in Python, provide the additional references/direction.

   *If your instructions are not clear, your work will **NOT** be graded!*
3. A short report describing your experiments and the results of your evaluations (~3 pages).

Expected Report Elements:

1. Dimension Reduction
   a. Provide the 2D scatter plot that you generated in step 2.
   b. Does it look like the anomalous and normal samples separate into discernable clusters?
   c. Are you able to manually draw a decision boundary on those samples?
2. Hyperparameter Tuning
   a. What process did you follow when tuning your hyperparameters?
   b. Provide some of the performance vs. hyperparameter plots for your models.
   c. What was the best performance you were able to achieve (you can provide several 'bests')?
3. Clustering Algorithms
   a. Which clustering technique do you think works best for anomaly detection (or are both equally effective)? Provide reasoning and evidence for your choice.
   b. The DBSCAN clustering algorithm is sensitive to two parameters. What are those? Report the evidence about this based on your implementation. *Hint: try different combinations of those parameter values and report the scores in a table/figure.*
4. Performance
   a. Provide the accuracy, TPR, FPR, and F1-score values for both the algorithms.
   b. Discuss if your classifier has any overfit/underfit issue based on the accuracy of training and testing dataset.
   c. Which performance metrics (e.g. Accuracy, DR, FAR, etc.) are the most important? *Provide reasoning and evidence for your choice.*
5. Include any citations in an appropriate and consistent format.
6. ***Not Expected*:** Background on the clustering algorithm or KDD'99 dataset. Especially if it leads you to copy other material (see "A Note on Plagiarism" below!).

Grading Rubric

Criteria	1 Poor	2 Basic	3 Proficient	4 Distinguished
NIDS Implementation (25%)	No or non-functioning code for anomaly identification techniques.	Code for one or both classifiers are functional, but one (or both) implementations have significant errors.	Minor errors or significant inefficiencies exist in implementation, but are otherwise mostly correct.	Techniques are perfectly correct. Code is cleanly written and handles most edge cases.
Experiments & Results (25%)	Minimal results are reported.	Results are incomplete and/or wrong, or are not presented in a way where its meaning is clear.	A reasonable set of results showing expected performance. Some presentation issues such as confusing graphs/tables or unnecessary detail.	Detectors achieve expected performance and all experiments are presented clearly.
Analysis & Questions (25%)	No or nonsensical analysis. Missing most questions.	Analysis is inaccurate or hard to understand. Some, but not all, questions are addressed.	Analysis is sensible and covers questions well. May be missing appropriate references or is weak in some areas.	Clear and accurate analysis that is backed up with appropriate references.
Writing Quality (15%)	Very poorly written and hard to follow.	Major points are visible, but writing may include many errors and/or lack focus and is disorganized.	Writing is clear enough to be understood, but some points may lack focus. Relatively few writing errors.	The paper is clear and well organized. Writing is smooth and polished with very few errors.
Code Documentation (10%)	Inadequate documentation provided both in the code and with the code.	Acceptable documentation for running the code, but lacking in-code documentation and helpful descriptions to demonstrate understanding.	Code includes some documentation (e.g. function docstrings, inline comments), however quality may be weak or unclear.	High quality documentation is provided. The purpose and function of all segments of code can easily be understood.

A: 3.25 average or higher
B: 2.5 - 3.25 average.
C: 1.75 - 2.5 average.

A Note on Plagiarism

When writing, you must include in-line citations whenever you are reporting on information that is not "general knowledge," i.e. anything you learned for this project and didn't know in advance. This is **NOT** just for quoted information. Failure to do this is plagiarism.

This article on plagiarism is good and covers the line between common knowledge and other material: https://writingcenter.unc.edu/tips-and-tools/plagiarism/. Also: https://www.plagiarism.org/ has a ton of additional information.

I have had students fail to follow these guidelines and get caught nearly every time I've taught my research seminar. These students get put on probation and have even been suspended from the university for this serious academic violation. *Please do not be the next!*