BTU Cottbus, Chair of IT Security
Prof. Dr.-Ing. A. Panchenko
T. Ziemann
E. Strehle, B.Sc.

Winter term 2019/2020

Part III: Outlier detection

Deadline: 04.12.2019

Study Project: Adversarial Machine Learning

Website Fingerprinting

Introduction

In the previous task we have collected a lot of network traces (e.g., TCP/IP-traces, TLS records). However, they may not all be useful for application in the machine learning phase of the project. Many of the may corrupted data which is not applicable for the learning, can be removed directly during the crawling process. We have already done this in the previous steps. However there still will be fetches which not yielded in error during the crawling of the website itself but still are very different from the mean ones. In this milestone of the project we want to remove such outliers. Thus, we will end up with a data set as clean as possible. The so obtained instances will then be used for training in the next part of the project.

Preliminaries

Please consider the following preliminaries before starting the actual implementation.

a) What is a outlier? How to define it in the context of our project? – Think about a reasonable mathematical model on the term outlier.

b) At which stages in your algorithm is outlier detection possible? (e.g. raw traffic, extracted records, generated features) When does it make the most sense?

c) Think about the consequences of removing outliers with respect to balance and size of the data set.

Tasks Outlier Detection

In this task we will implement three different approaches to detect and remove outliers. Each of these approaches will have its own advantages and disadvantages. Therefore we should evaluate and compare them to each other. Anyway, before we need to understand the term outlier at all and find a suitable mathematical model for it. The following sub-tasks will guide you during this milestone.

Subtask 01: Research

Research about the term outlier detection and the mathematical model behind it. Thereby find a reasonable model of outliers in the context of our project. Further, inform yourself about three different approaches to detect outliers. It is also possible that you will have different models of outliers for the different algorithms proposed.

Subtask 02: Implementation

Now it gets time to implement the outlier removal strategies researched before. Therefore we will extend our toolbox by the program `remove_outliers`. Your are not allowed to use any already existing outlier detection library/implementation. The researched/proposed algorithm should be implemented on your own.
The program should provide the following command-line arguments to the user:

Option	Parameter	Description
-h	-	Shows an overview with all implemented command line parameters.
-l	-	Shows the log-file. If more than one exists, list all existing logs.
-s	-	Simulation mode.
-i	input file	Single feature file to detect/remove outliers from.
-a	algorithm name	Name of the Outlier detection algorithm.
-n	Minimum nb.	Minimum number of feature vectors to remove. (default=0)
-N	Maximum nb.	Maximum number of feature vectors to remove. (conflicts with *-v*)
-v	variance boundary	Upper boundary of variance after outlier removal. (conflicts with *-N*)
-o	output file	File to store the adjusted features.

During execution the program should print out at least the following information:

a) Number of outliers removed. Number of feature vectors in the input file.

b) Variation of the input feature vectors and the adjusted ones. Think *carefully* about how to represent/compare this variations correctly with respect to the changed amount of data, i.e. count of vectors int the data-set, due to the removal.

c) Min, Max and Mean curve, i.e. feature vector, of the input data set as well as of the adjusted one as well as their respective distances.

May it makes also sense to write this information to an additional file named the same as the input but with different file-suffix, e.g. `feature_vectors.ori`. The developed outlier remover should also offer an so called simulation mode. In this mode your program only detects the outliers but either does not change the input file nor write an output file. In this case the program just prints out the information mentioned above (resp. write them to a `*.ori` file).
Think about how to integrate the developed outlier removal tool in the already existing toolbox. Also provide a script to process the outlier removal for many traces!

Subtask 03: Evaluation and Comparison

Evaluate your outlier detection approaches by the following means: For each algorithm analyze

a) Number of outliers removed in total. / Number of outliers removed to reach certain variance in the data. Fraction of this number with respect to the total amount of feature vectors.

b) Variation of the input feature vectors and the adjusted ones. Think *carefully* about how to represent/compare this variations correctly with respect to the changed amount of data, i.e. count of vectors int the data-set, due to the removal.

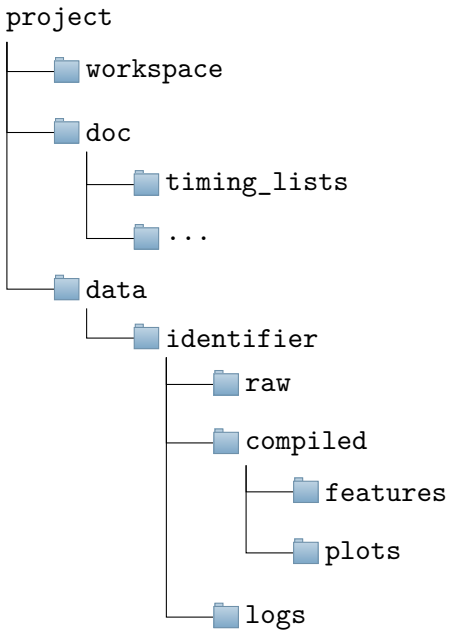
c) Min, Max and Mean feature vector of the input data set as well as of the adjusted one as well as their respective distances.

d) Computational properties of the algorithms, e.g. CPU cycles, memory consumption, etc.

May support your discussion using graphs whenever possible. Next, compare the algorithm with respect to the information gathered before. Which algorithm fits your data-set best?

Directory tree

Use the following directory tree for your git repository:



Hint

Please note that the effort for this project is calculated for three persons. Therefore it makes sense to distribute the workload.

Preparation of consultation

Prepare yourself for a short consultation of about 15 minutes. You shall give a short overview of libraries, scripts or already existing tools you have used. Furthermore present and explain your implementations. Give a proof of work through a demonstration of your program/scripts.

Good luck!