BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

ANOMALY DETECTION IN NETWORK COMMUNICATION

TERM PROJECT

AUTHOR **Bc. PAVEL YADLOUSKI**

BRNO 2022

Abstract

Keywords

IEC104-5, OC-SVM, Machine Learning

Reference

YADLOUSKI, Pavel. *Anomaly detection in network communication.* Brno, 2022. Term project. Brno University of Technology, Faculty of Information Technology.

Anomaly detection in network communication

Pavel Yadlouski
July 13, 2022

Contents

Chapter 1

Introduction

Main topic of this work is to research possibilities for creating machine learning model for anomaly detection in network communication. This problem might seems to have trivial solution using machine learning algorithms, but in reality there are several challenges that developers need to overcome before the model starts to provide applicable results.

One of the most important questions the developer has to answer is what are the most significant features for the given communication. There is no universal answer for this question. To provide useful set of features developer has to have baggage of experience and deep-enough knowledge of the domain. Without any prior hands-on experience, trial and error method is the best way for that task.

Next question is can we create one universal model that would provide meaningful results for different communication. The short answer is no. The reason for this answer is that each network communication has different 'normal' behaviour, different communication pattern in other words. But what is good in this question, is that we can create universal algorithms for building such models for each communication. So, main goal for anomaly detection in network communication is to build an universal algorithm that we can apply on different networks and this algorithm would provide model for anomaly detection in that particular network.

Before we deep dive into this topic, we should clarify one important thing: any communication is based on some protocol. Each protocol might have different interesting fields for building the model. So, here universal algorithm means universal for particular protocol in any network. In this work we would discuss IEC104-5 [4] protocol as an example.

In the chapter 2 minimal background knowledge is introduced. There you will find information about IEC104-5 protocol, chosen features of this protocol, possible techniques for anomaly detection using machine learning algorithms with description of one's that is used in this work. Next chapter 3 is dedicated to build of the model using OneClass-SVM technique, how parameters for this model set. After that follows validation and testing phase were method of the model is tested on pseudo-random anomalies. At the end, result of this work and the summary are presented.

Chapter 2

Background

In this section we would introduce you background knowledge for this domain: what is IEC104-5 protocol, what is technique used for building the model,

2.1 IEC104-5 protocol

IEC104-5 protocol is used in Supervisory Control And Data Acquisition (SCADA)[3] systems for monitoring purposes. This procol runs over standard TCP/IP stack using TCP as a transport layer. IEC104-5 has typical master-slave communication type where there is one or several control stations that collect data from controlling stations. Controlling station can be some measurement device or the host to which such device connected, or any other device that is interesting got particular application and supports IEC104-5 protocol.

Information is IEC104-5 is formatted in so called Application Protocol Data Unit (APDU) that contains inside Application Protocol Control Information (APCI) and Application Service Data Unit (ASDU). APIC can be one of three formats: *I-format*, *S-format*, *U-format*. Depending on ACPI format, after ASDU can follow APCI or not (*S-format* and *U-format* are not followed by ASDU). More information regarding format of the IEC104-5 you can find in the technical report [4] in section 2.1.

2.2 Interesting features

When talking about fields (features) that can be monitored in the IEC104-5 communication, there are several variants [1]:

1. Length of APDU
2. Type of the APDU (type)
3. Type of information object
4. Sequence of COA - Common Address of Application Service Data Unit
5. Sequence of COT - Cause of Transmission
6. Sequence of IOA - Information Obejct Address
7. Inter-packet interval

8. Direction of the packet (from master or to master)

Even if it seems that more feature model have, more accurate result will be, it is not always true. During implemented on the model, it was revealed that some subsets of mentioned features provides better result than when model is trained using all features From different tries, following subset of features is the seems to be most suitable:

1. Length of APDU
2. Type of the APDU (type)
3. Type of information object
4. Inter-packet interval
5. Direction of the packet

For more convenient work with data, relative timestamps are also included into parsed data set. For direction there are two columns: `src` and `dst`. These columns contain an integer starting from 0 that represent kind of identifier of the host. The reason why direction is not marked as from master and to master is because in IEC104-5 there are types of packets that both sides can send – packets for file transfer. So, to make features more robust, it was decided to implement direction using simple identifiers.

2.3 Anomaly Detection Technique

While talking about anomaly detection task, there are several techniques that can be used:

- Simple statistical models using Box Plot (IQR) or Three-sigma rule
- One-Class Support Vector Machine classification (OC-SVM)
- Time Series and ARIMA model
- One-class Neural Networks (OC-NN)

In this work I used OC-SVM to implement machine learning model. Reason for this chose is that SVM in general are one of the best classification models that have proven themselves throughout history.

Support vector machines (SVMs) are a set of related supervised learning methods used for classification and regression. They belong to a family of generalized linear classifiers. In another terms, Support Vector Machine (SVM) is a classification and regression prediction tool that uses machine learning theory to maximize predictive accuracy while automatically avoiding over-fit to the data [2].

One-Class SVM

For anomaly detection with SVM we distinguish specific variant of model called One-Class SVM (OC-SVM). In simple words, in OC-SVM we are dealing with only one class of data that we are interested in. In other words, we are interested in „normal“ data, everything else is considered as an anomaly.

Chapter 3

Model building

For building the model, scikit-learn[1] framework was used.

3.1 Communication Analyses

Before actual model building, communication in given data sets need to be analysed. There are two data sets with IEC104-5 communication:

1. *mega104-17-12-18.pcapng*: contains file transfers from master and to master with 37660 entries in parsed data set (after extracting features) during 2 days 19 hours 55 minutes.

2. *10122018-104Mega.pcapng*: contains spontaneous communication mostly to master (controlling station sends some values to the master) with 66377 entries in parsed data set during 4 hours and 53 minutes.

During analyses, some types of IEC104-5 packet were dropped because those they didn't contain any useful information for classification. Moreover, values of chosen fields in this packets were breaking the model.

As it was discussed previously, direction of the packets are encoded as an integer that defines some devices in the communication but don't specifies which devices acts like a master.

3.2 OC-SVM Parameters

As the firs parameter of the model is features that we would feed the model. This step we already discussed in section 2.2. Other parameters specific for OC-SVM are:

- kernel: specifies the function of the support vectors
- ν (nu): both a lower bound for the number of samples that are support vectors and an upper bound for the number of samples that are on the wrong side of the hyperplane. The default is 0.1. The nu parameter must be in the range [0,1] [2]
- γ (gamma): defines how much influence a single training example has. The larger gamma is, the closer other examples must be to be affected

[1] https://scikit-learn.org/stable/index.html

There are more parameters that can be specified for the model, but we would discuss only these three.

There are several variants for the kernel for SVM in skylearn: linear, polynomial, radial basis function (RBF) and sigmoid. Polynomial kernel is considered to be inefficient in practise, when RBF and sigmoid kernels considered to be preferable kernels for SVM. To choose RBF or sigmoid kernel, an experiment was made where several models with different parameters of ν where trained on the two data sets separately. Nu value was varies in range [0.015, 0.021] with step 0.001. For gamma parameters tree values are tested: `auto`, `scale` and 0.1

Models were trained on 2/3 of each data set and than tested on the rest 1/3 of corresponding data set. After training models on two data sets, key value for choose was accuracy (percent of correctly classified points) and percent of false-positive. After collecting the data to the data set, rows where false-positives was lower then 3% were filtered

kernel	gamma	nu	false_positive_1	false_positive_2
sigmoid	scale	0.015000	0.015852	0.017265
		0.016000	0.016728	0.018575
		0.017000	0.017604	0.019434
		0.018000	0.018878	0.020745
		0.019000	0.020073	0.021784
		0.020000	0.020870	0.022824
		0.021000	0.021985	0.024451
rbf	0.10	0.017000	0.023339	0.018711
		0.021000	0.023260	0.016632

As we can see, combination of sigmoid kernel with scale value for gamma and whole range of nu value [0.015, 0.021] while rbf kernel with gamma equals to 0.1 provides similar results only with nu equal to 0.021 or 0.017. But result of both variants are suitable for further validation.

Chapter 4

Validation and tests

	kernel	rbf		sigmoid	
	gamma	0.1	scale	0.1	scale
nu	interval				
0.017	5	1.398489	21.420100	21.527331	9.512613
	7	1.473942	21.749738	23.096061	11.205972
	10	2.039134	23.496930	24.322280	11.923516
0.021	5	1.263915	21.281265	21.588278	11.281149
	7	1.406228	21.574253	22.051360	11.850376
	10	2.042682	22.643155	23.518404	13.655174

Table 4.1: Mean percent of not detected randomly generated anomalies

	kernel	rbf		sigmoid	
	gamma	0.1	scale	0.1	scale
nu	interval				
0.017	5	9.375000	29.577465	30.000000	29.746835
	7	7.142857	30.000000	30.000000	29.641694
	10	7.058824	30.000000	30.000000	29.764652
0.021	5	12.500000	30.000000	30.000000	29.852744
	7	12.000000	29.569191	30.000000	29.817708
	10	7.017544	29.644998	30.000000	29.725759

Table 4.2: Max percent of not detected randomly generated anomalies

For validating trained model build in chapter 3 following steps are made for validation:

1. Original data sets is split into intervals of 5, 7 and 10 minutes
2. Model trained with rbf and sigmoid kernel are used both with nu $\in \{0.017, 0.021\}$ and gamma $\in \{0.1, \text{scale}\}$
3. Pseudo-random subset of intervals of each type (5, 7, 10 minutes) are take
4. For chosen subset of intervals anomalies were created using algorithm described in Appendix A

5. Model with given parameters (kernel, nu, gamma) were used for prediction on interval without anomalies and then with anomalies

Key metric of validation is difference between real percent of anomalies and percent of anomalies in prediction. Mean difference of not detected anomalies for same type of interval (5, 7 or 10 minutes) is shown in the Table 4.1. Also, important perspective on not detected anomalies is what is the maximum value of not detected anomalies Table 4.2.

Lets look at an example of out from validation

```
interval_index = "1"
interval_time = "5"
data_set_num = "1"

interval = get_interval(interval_index, interval_time, data_set_num)
r = create_anomalies(interval)
anoms = pd.read_csv(r["frame"])
orig = pd.read_csv(interval)

print(f'Number of generated anomalies: {len(r["indexes"])}')
print(predict("1", 0.017, interval, gamma=0.1))   # Prediction WITHOUT anomalies
print(predict("1", 0.017, r["frame"], gamma=0.1)) # Prediction WITH anomalies
```

Listing 4.1: Example of perdition result

The output of this code is following:

```
Indexes of generated anomalies:
[ 5  5 12 18 20 23 24 25 26 27 28 29 30 31 32 33 34 35 40] (count 19)

------------------------------
Datset: data/csv/intervals/model-1-5min/frame-1-1.csv
Total number of samples: 67
Normal: 66 (98.51%)
Anomalies: 1 (1.49%)
------------------------------
    asdu_len  io_type  type_id  src  dst  interval  anomaly
21        40        2       36    1    0  6.203463       -1

------------------------------
Datset: data/csv/anomalies/frame-1-1.csv
Total number of samples: 67
Normal: 48 (71.64%)
Anomalies: 19 (28.36%)
------------------------------
    asdu_len  io_type  type_id  src  dst  interval  anomaly
5         38        2       36    1    0  2.600935       -1
12        39        2       36    1    0  4.802479       -1
18        35        2       36    1    0  0.800157       -1
20        31        2       36    1    0  1.864086       -1
...
34        25        2       36    0    1  5.402437       -1
35        25       32       36    0    1  0.600301       -1
40        32        2       36    1    0  3.000017       -1
```

In the code, we get interval for specified values, creating the anomalies with function `create_anomalies(interval)` and run `predic` function on this original interval and on anomalies. Also we can look at indexes on which anomalies were created. In the output of each run of `predict` function we see the size of given data frame, how many items are classified as normal entries and how much are as anomalies. Also, Pandas DataFrame object is printed with entries that are classified as anomalies. See more complex example in file *test-output.txt*.

Chapter 5

Result

From validation part of the model chapter 4, we see that interesting thing about suitable parameters of the model. Even when model with sigmoid kernel and the lowest nu value (0.015) seems to be better then rbf kernel with higher nu value from amount of false-positives predictions side, in practice for detection purposes, model with rbf kernel is significantly better for both two types of communication. From length of time interval side, each type of interval (5, 7, and 10 minutes) can be used and would provide similar results.

Chapter 6

Summary

In this work we looked and machine learning algorithm One-Class Support Vector Machine (OC-SVM). As an example for analyses we used IEC104-5 protocol that is based on standard TCP/IP stack. For analyses we had two data sets with different type of communication. From set of available kernel with related parameters for OC-SVM we took two of them: rbf and sigmoid kernels. Decision was made based on amount of false-positives values in each data set. Trained models were tested on several intervals with different subset of values for nu and gamma parameters. After testing we take rbf kernel with nu equals to 0.017 and gamma equals to 0.1 because this combination of kernel and parameters gives minimal mean percent of not detected randomly generated anomalies.

Bibliography

[1] Burgetová, I. and Matoušek, P. *Statistical Methods for Anomaly Detection in Industrial Communication.* 2021. 59 p. Available at: https://www.fit.vut.cz/research/publication/12502.

[2] Jakkul, V. *Tutorial on Support Vector Machine (SVM)* [online]. 2011. Available at: https://course.ccs.neu.edu/cs5100f11/resources/jakkula.pdf.

[3] Lin, C.-Y. and Nadjm Tehrani, S. *Understanding IEC-60870-5-104 Traffic Patterns in SCADA Networks* [online]. 2018. Available at: https://www.diva-portal.org/smash/get/diva2:1287470/FULLTEXT02.pdf.

[4] Matoušek, P. *Description and analysis of IEC 104 Protocol.* 2017. 38 p. Available at: https://www.fit.vut.cz/research/publication/11570.

Appendix A

Generating of anomalies

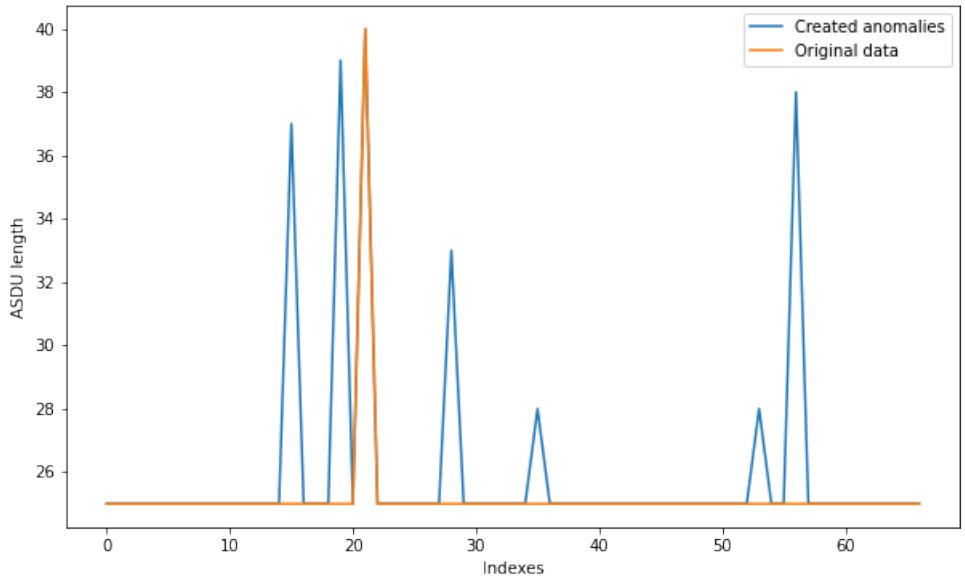

Figure A.1: Anomalies created by the algorithm

For testing purposes, function for creating pseudo-random anomalies were created. Function uses is following algorithm:

1. Interval for given parameters is loaded to `pandas.DataFrame` object
2. Pseudo-random subset of indexes is generated all available indexes with `numpy.random.randint()` function
3. Pseudo-random values for generated indexes from range of minimal value of `asdu_len` column to max
4. Original values on generated indexes are replaced with generated values
5. One pseudo-random index is generated

6. From this index value of `src` and `dst` values for next 10 elements are swapped
7. Updated data frame is store

On the Figure A.1 you can see the result of this algorithm