Anomaly Detection in Cellular Network Measurement Data With Bayesian Networks

Jure Kopitar
Faculty of Computer and Information Science, University of Ljubljana
jk1156@student.uni-lj.si

Abstract **- With increased availability of cellular network technologies, the number of connected devices is growing. Field measurements are performed in order to gain an insight into the network status. In this paper we attempted to construct an anomaly detection system that would allow automatic analysis of cellular network measurement data. We used Bayesian Network as our model and manually specified its topology using the expert knowledge. Potential anomalies were searched for by computing the *log-likelihood* of individual data samples. We analyzed the effects of using quantitative rules when computing *log-likelihood.* In hope of gaining a performance benefit, *log-likelihood* computation was implemented in C++. Furthermore, our anomaly detection system was tested on a real-life measurement dataset.**

Keywords

Log-likelihood, LTE, UMTS, Rcpp, OutlierMiner

I. Introduction

In recent years mobile carrier networks have evolved from a closed-off system providing only a single service to an enormous ecosystem offering a variety of services. With the enhancement of data throughput capabilities and decrease of latency, the usage of mobile networks became viable in various fields. One of key fields are self – driving cars that require large amounts of real-time information from the road and the other vehicles. Another field is the Internet of Things which consists of a large number of connected devices with low power consumption.

In order to satisfy the demands of the market, a new generation of cellular mobile communications is introduced every few years. 5G is the most recent mobile communication standard with the first terminal devices expected in 2019. Its key features are the following: considerably lower latency compared to previous generations, higher data throughput and usage of higher frequency spectrum combined with existing frequencies. The latter will allow creation of large number of cells, each with a small area coverage.

Cellular network operator is responsible for maintenance and management of a cellular network. Its primary task is assuring that such network performs properly. Quality assurance will become even more important with the introduction of mission critical applications. Real-time network monitoring allows an operator to respond to different kinds of faults faster. Many times, early detection can even prevent certain malfunctions.

With the increasing number of terminal devices as well as base stations the challenge of managing a cellular network grows larger. The condition of network devices can be assessed by measuring various network parameters and analyzing the results in real time. There are two main sources of network disturbances that can be measured:

- radio–channel disturbance: electromagnetic interference, high signal attenuation, excessive amount of users, weather effects
- IP or operator core network disturbance: network node overload, electromagnetic interference

The amount of data acquired through measurements is too large for a single technician to analyze visually. Therefore, computer-based techniques are employed in order to perform anomaly and fault detection.

In this article, we suggest a data-science based approach to anomaly detection in cellular network data. The ability to automatically detect anomalies is important because it allows faster issue resolving, relieves tech personnel and, in turn, allows an operator to provide better service to its users.

We analyze the data that was acquired by using a stationary measurement probe, connected to a base station of a cellular network operator. The probe performed measurements on two different generations of mobile communication standards, namely 3G (UMTS – Universal Mobile Communication System) and 4G (LTE – Long Term Evolution). See Appendix I for specification information. We explain how the technological differences between the standards are reflected in data. We propose a Bayesian network (BN) topology as a way to incorporate expert knowledge into machine learning models. We use the BN to represent normal behavior of the observed part of the cellular network. The BN allows us to detect anomalous samples by computing the log-likelihood on a per-sample basis. We explain our implementation of a semi-supervised anomaly detection model in R programming language.

It is common to have several measurement probes active in a cellular network, which can lead to large amounts of data requiring analysis. In order to speed up the analysis process we implement log-likelihood computation in C++ programming language and analyze the effect on performance. We also test how *OutlierMiner* algorithm[2] affects anomaly detection results.

The rest of the paper proceeds as follows: Section II gives a brief overview of related work on anomaly detection. Section III explains our approach to anomaly detection. Section IV describes the dataset. Implementation details are presented in Section V. Section VI explains our approach to testing. In Section VII, the results are presented.

II. Related Work

Extensive research has already been done on different approaches to anomaly detection. Ahmed et. al(2016) categorize and in turn analyze anomaly detection approaches in the context of Information and Communication Technology networks. They focus on the detection of network intrusions and research challenges associated with the lack of datasets used for intrusion detection.

Görnitz et. al(2013) explain why semi-supervised anomaly detection approaches that are derived from supervised classifiers have issues detecting new types of anomalies. A generalization of the support vector data description is proposed which allows incorporation of labeled data into a semi-supervised anomaly detection approach. This approach is then applied in a network intrusion detection domain.

Deljac et. al(2014) feed error report and customer trouble call information into a BN, which determines the most probable location of a faulty network element.

Babbar et. al(2006) present a BN – based approach of incorporating expert knowledge into an anomaly detection system. A short introduction to BNs is provided. Two quantitative rules are introduced to help uncover anomalies. Refer to Section VII of this paper for results of using these rules on a cellular network dataset.

Scutari (2010) introduces the *bnlearn* package for Bayesian Networks in R, also used when researching for this article. Structure learning algorithms, network scores and conditional independence tests are explained. Several BN implementation examples are provided with explanation and R code.

III. Anomaly Detection

Anomaly can be defined in different ways depending on the application domain. For our domain, we found widely accepted definition by Hawkins (1980) suitable: *'An anomaly is an observation which deviates so much from other observations as to arouse suspicions that it was generated by a different mechanism.'*

Ahmed et. al(2016) categorize the anomalies as follows:

- **point** anomaly: when a particular data instance deviates from the normal pattern of the dataset
- **contextual** anomaly: when a particular data instance deviates from the normal pattern given a context
- **collective** anomaly: when similar data instances deviate from the normal pattern of an entire dataset.

Our goal in this article was to identify point anomalies given the measurement data. Identification of certain contextual anomalies would require a refined approach to performing cellular network measurements. Several measurement locations would be required in different environments over a longer period of time. The reason for this is that the number of users per base station fluctuates through time. An example would be a base station situated close to a highway. We can expect higher volume of users and therefore higher base station load during the rush hour. However, our model should be trained not to expect increased number of users at the time of rush hour during the weekend.

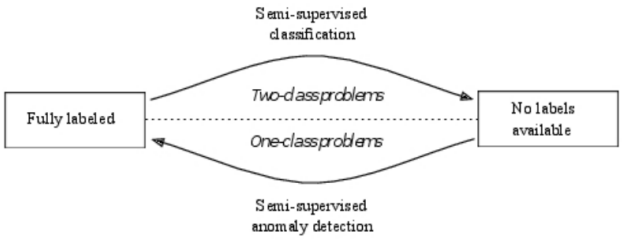

Fig. 1. Semi-supervised learning. [4]

A. Semi-Supervised approach

We will refer to the approach used for anomaly detection in this paper as *semi-supervised*. Other papers can use different terms when referring to this approach. Ahmed et. al(2016) for example refer to it as *classification*. Semi-supervised approach is based on the assumption that anomalous data instances are sparse compared to "normal" data instances. Thus, we are dealing with an imbalanced dataset often without labeled anomalies. When employing the *semi-supervised* approach we are modelling the *normal* behavior of a system. Therefore, model parameters are fitted using normal data. The advantage of such an approach over traditional supervised approach is the ability to work without a labeled dataset. Additionally, a model built using the supervised approach will struggle to detect new types of anomalies that were not present in learning phase. On the other hand, a semi-supervised model could produce a high false-positive rate

should a new kind of previously unseen normal data appear.

We employed the Bayesian Network as the model for *semi-supervised* anomaly detection because of the benefits of using expert knowledge in the modelling phase. Modelling with BNs is performed in two steps. In the first step, a directed acyclic graph topology which represents a BN is specified. In the second step, network parameters (probability tables) are fitted using the training data. There are several structure-learning algorithms available for BNs, which can specifiy BN topology based on a dataset.[7] By using the expert knowledge and understanding of the relations between observed attributes we were able to manually specify the topology of our BN (Section V).

When using Bayesian Networks for classification, target variable is usually represented by a root node. The class of the target variable can be inferred by computing its probability given the evidence (i.e. the observed node values). In a semi-supervised approach to anomaly detection there is no node that would represent the target variable. Rather than that, all nodes in a BN represent the observable attributes of a system. The parameters (probability tables) of the BN are fitted in a way to represent the normal behavior of the system. To find out, how anomalous a sample is, the likelihood that this sample was generated by the BN is computed. Should the likelihood value be low, it can be assumed that the sample is anomalous.

A Bayesian Network can be understood as a compressed representation of the joint probability distribution of a system. The compression is achieved by the conditional independence assumptions. The edges in the BN represent the conditional dependence between attributes (nodes). Only the edges where conditional dependence is significant are kept, which reduces the size of conditional probability tables.

The likelihood of a sample is the joint probability of the observed (measured) evidence. Joint probability can only be computed if the evidence is complete, i.e. all the nodes of the network have been observed. Typically, the natural logarithm of the joint probability is computed, which is called the *log-likelihood* of a sample. The main reason for the usage of log-likelihood is to avoid the underflow of floating-point precision. Joint probability of a sample (1) is a product of prior and conditional probabilities. Therefore, it is often a sufficiently small number to cause floating point precision underflow. Log-likelihood (2) resolves this issue. Since the usage of logarithm replaces multiplication with summation, some systems can gain a performance benefit when computing log-likelihood. It should be considered that log-likelihood shifts the probability interval from **[0,1]** to **(-∞, 0]** where 0 indicates a certain event.

$$P(X_1, X_2, \dots X_n) = \prod_{i=1}^{n} P(X_i | parents(X_i)) \quad (1)$$

$$LogLik(X_1, X_2, \dots X_n) = \sum_{i=1}^{n} \ln(X_i | parents(X_i)) \quad (2)$$

X_i - the value of an ith node (represents an attribute)

parents(X_i) – the values of all nodes that are parents to the ith node

n – the number of nodes in a Bayesian Network

IV. Dataset

Our dataset consisted of two tables containing raw measurement data with corresponding timestamps. Each column in a table represents a time series of observed values for the given attribute. Both tables contain parameters related to the quality of radio channel, radio access type, measurement probe geolocation and SNMP (Simple Network Management Protocol) interface configuration parameters. In addition, the first table contains download speed measurement data, whereas the second table contains data acquired through the usage of ICMP (Internet Control Message Protocol). The recorded parameters are *round trip time*, with corresponding *traceroute* and *packet size* information (see Appendix II).

Semi-supervised approach to anomaly detection requires dataset which represents the normal system behavior. This is why we conceived our measurement approach in a way that reduces the number of possible sources of anomalies. Therefore, the measurement probe was stationary and connected to a base station of a single operator and the packet size for ICMP was set to a fixed size. Only two radio access technologies were used, namely LTE and UMTS. Radio access technology directly refers to the generation of mobile communications.

The differences between LTE and UMTS can be observed from the acquired data. LTE being a more recent standard allows for lower average *round trip time* of **25.5ms** compared to UMTS **75.9ms**. LTE offers significantly higher download speeds (**19.1Mbps** mean, **65.2Mbps** peak) compared to UMTS (**10.6Mbps** mean, **12.4Mbps** peak), but suffers from a higher download speed variance. A single UMTS frequency band and three different LTE bands were used during data acquisition.

Data was acquired from 7.11.2018 until 8.12.2018. Measurements were performed with a period of one minute plus time of measurement. The results were

averaged when several were available at the same time. Timestamps were synchronized between both tables.

We selected a subset of six attributes (see Appendix II). There are two reasons why we assumed that the selected subset was appropriate:

- each attribute is a representative of its domain
- the interactions between the attributes in the subset were understood

V. Implementation Details

A. Bayesian Network

As mentioned in the introduction, we used the *bnlearn* package for Bayesian Networks.[7] We specified the BN topology manually (Fig. 2), and compared it to a learned network topology (Fig. 3). We performed the comparison by computing Bayesian Information Criterion (BIC) network score on a testing dataset. BIC score is equivalent to minimum description length and penalizes complex networks. Higher BIC value is better.[6][7] Our network scored **-19459** compared to **-16186** scored by network topology learned by hill-climbing greedy search algorithm. Although learned network scored better than manually specified network, this was not completely reflected in the results (see Section VII). It is worth noting that if the number of attributes should be too large for a manual BN topology specification, only a subset of network edges can be specified (whitelisted). A structure learning algorithm can then be used to search for additional edges.

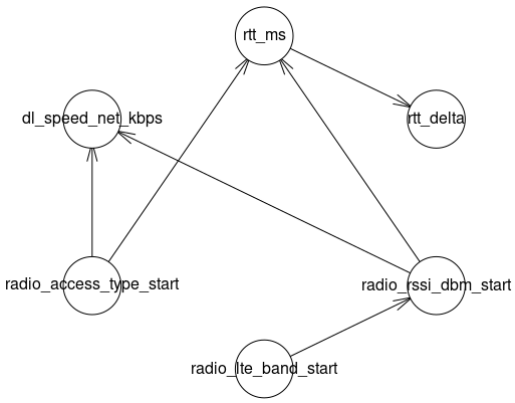

Fig. 2. Manually specified Bayesian Network topology.

We split our dataset into training and testing set and discretized the continuous attributes. Known anomalies that were present in the training set were removed in order to provide normal data for the BN to fit its parameters on. The removed anomalies were long packet delays caused by the usage of public IP network which were unavoidable in our measurement scenario. BN parameters (i.e. probabilities) were fitted using *Bayesian parameter estimation* for the given manual BN topology and training set.[7]

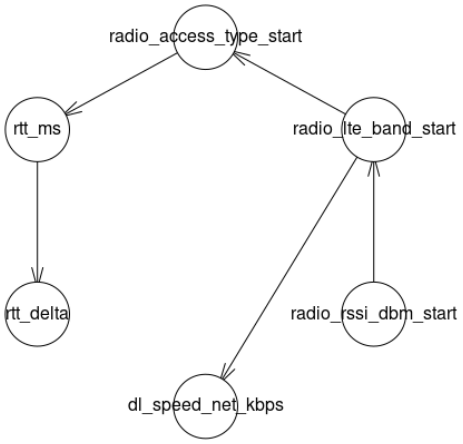

Fig. 3. Learned Bayesian Network topology. Hill-climbing greedy algorithm was used.

B. Log-likelihood Computation

Once the BN parameters were fitted, log-likelihood could be computed for each test data sample. *Bnlearn* package provides a function which computes log-likelihood of a sample. In real-life scenario there are large amounts of samples expected to arrive from several measurement nodes. Therefore the function for log-likelihood computation is required to process a large number of samples in a time period. Given such scenario, the log-likelihood function from *bnlearn* failed to provide a sufficient data throughput on a per-sample basis.

This is why we resorted to implementation of our own log-likelihood function in C++. The function grants us a performance benefit as well as independence from R should our anomaly detection system reach actual implementation. Additionally, our implementation of log-likelihood function allows us to implement the rules provided by the *OutlierMiner* algorithm.

We implemented the log-likelihood function using *Rcpp* package which enables code compilation and C++ function calls directly from an R script. We provided an initialization function which accepts a BN with fitted parameters from the *bnlearn* package as an argument. Therefore, only small changes to existing R code were required. Each of the BN probability tables is stored in a class which also stores the names of the node's parents. For each value in probability table, its logarithm is computed and in turn stored in a hash table. The initialization function supports natural logarithm and logarithm with base 10.

Function for log-likelihood computation accepts an R data frame containing the observed evidence as an argument. The function iterates through the nodes. For each node it generates a key from the evidence which allows it to access the probability from the node's hash table. The

resulting log-likelihood is the sum of probabilities over all nodes (see (2)). Log-likelihood function can be parallelized should there be a need to compute log-likelihood for a BN with a large number of nodes. At the time of writing the function supports complete evidence only. Missing evidence values can be inferred using the BN in the same way as a BN would be used for classification.

Code is available at:

https://github.com/nezezime/bnAnomaly

C. OutlierMiner Rules

The likelihood (joint probability) of a sample is a product of conditional and prior probabilities of a BN. A sufficiently high number of low probability values contributes to low likelihood of a sample which is therefore treated as anomalous. Babbar et. al(2006) explain that: *"In data mining terminology, prior and conditional probability are referred to as support and confidence respectively."* Should both confidence and support be low for the given attribute values, we are most likely dealing with a "noise event". The *OutlierMiner* algorithm which they suggest incorporates two quantitative rules which remove the terms with low prior and conditional probability from the likelihood computation.

The rules are defined using three parameters: *minsupp*, *minconf* and *maxconf. Minsupp* is computed for each root node in *Rcpp* initialization function. *Minconf* and *maxconf* are user-defined. For each root node and its child, the log-likelihood function assesses the quantitative rules. If both rules turn out to be false, the conditional probability of the child node for the given pair is omitted from the log-likelihood computation.[2]

VI. Testing Methodology

Since there was no testing dataset with labeled anomalies at our disposal, we opted to take a heuristic approach to validation of the anomaly detection model. There were two testing datasets at our disposal. The first dataset was an extension of the training set without the anomalous samples removed. We will refer to this dataset as *testing dataset.* The second dataset contained only LTE samples. Some samples were normal data, while the others were measurements recorded while one of the antennas of the measurement probe was covered with aluminum foil. The probe has two antennas, the second antenna was not covered and was orthogonal to the first antenna. No other parameters were altered during the course of the experiment. Time when the antenna was covered was recorded. We will refer to this dataset as *radio dataset.*

Since aluminum is a reflective material for electromagnetic radiation, the electromagnetic waves emitted by the base station were reflected away from the antenna. This resulted in a decreased received signal strength (*radio_rssi_dbm_start* attribute). Thus a dataset with known anomalous data that originated from radio-channel was at our disposal.

Radio dataset also served as a baseline for the tuning of our anomaly detection model. An anomalous sample is a piece of data for which the BN outputs a sufficiently low log-likelihood. Should the threshold be set too high, normal data would be classified as anomalous. We used the radio-channel anomaly dataset to set the threshold as low as possible while keeping the radio channel anomaly detected. Afterward we analyzed the results for the *testing dataset* using the threshold from *radio_dataset*.

In Fig. 4 it can be observed that at 00:16 time *radio_rssi_dbm_start* increases. The increase is consistent

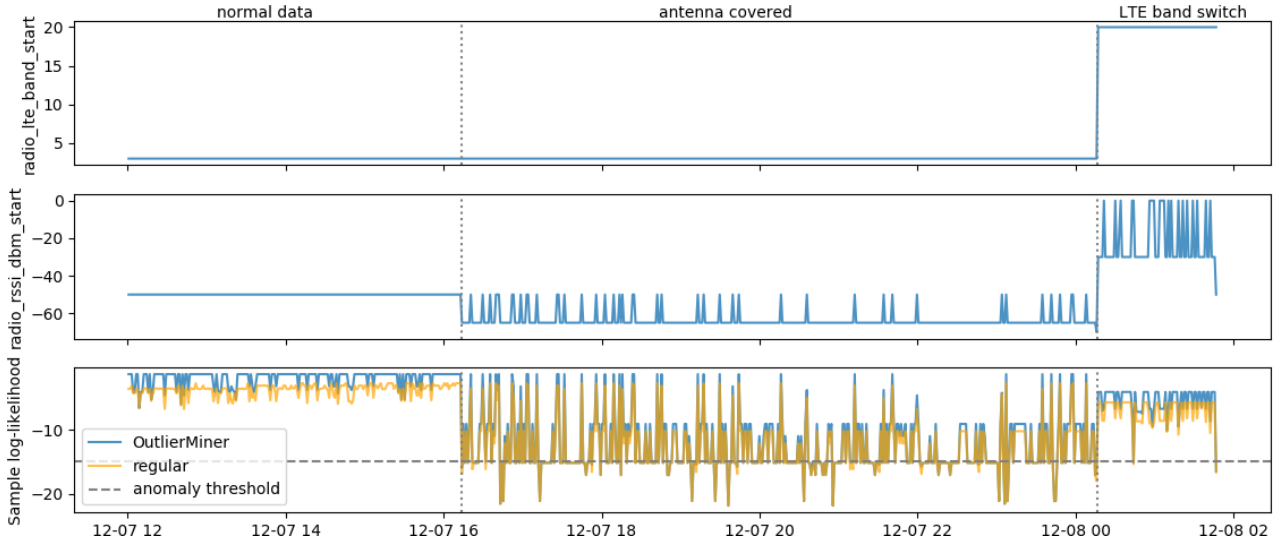

Fig. 4. *radio dataset log-likelihood* results for manual BN topology with and without *OutlierMiner* rules. *Radio_rssi_dbm_start*, the received signal strength decreases on December 7th 16:13, the time when measurement probe antenna was covered. LTE band switch occurs on December 8th 00:16.

Table I
Testing results for *radio dataset*

BN Topology	With OutlierMiner (*minconf* = 0.1, *maxconf* = 0.8)	*log-likelihood*				Anomaly threshold log-likelihood
		Normal segment		Radio anomaly segment (covered antenna)		
		mean	variance	mean [a]	variance	
manual	yes	-1.93	1.34	-13.23	11.85	-15
	no	-3.73	0.83	-13.72	8.63	-15
learned	yes	-3.16	0.83	-10.41	1.41	-10.5
	no	-3.16	0.83	-10.45	1.41	-10.5

[a] *radio_rssi_dbm_start* values sometimes jump to normal levels in radio anomaly segment (Fig. 4). The samples with normal *radio_rssi_dbm_start* values were excluded from the computation of the mean.

with measurement probe switching the LTE frequency band from 3 to 20 (*radio_lte_band_start* attribute). With the increased received signal strength most of the remaining data instances were classified as normal. The first antenna of the measurement probe was still covered up at this point. We assume that the reason for the signal strength increase was the orthogonal polarization of the second antenna which must have been consistent with the polarization of the signal at band 20. However, we cannot explain why the measurement probe swithced LTE band precisely at the given time. Therefore the corresponding data was not taken into account at the time of the analysis.

VII. Results

In this section we compare manual and learned BN topologies on a *testing* and *radio dataset*. Furthermore, the effect of using the *OutlierMiner* rules is analyzed for each BN topology. The effect that *log-likelihood* computation in C++ has on the performance is also analyzed.

As described in Section VI, *log-likelihood* threshold for anomalous samples was determined according to *radio_dataset*. In Table I, thresholds are recorded for each BN topology. We also analyzed the properties of *log-likelihood* time series for normal and radio anomaly data

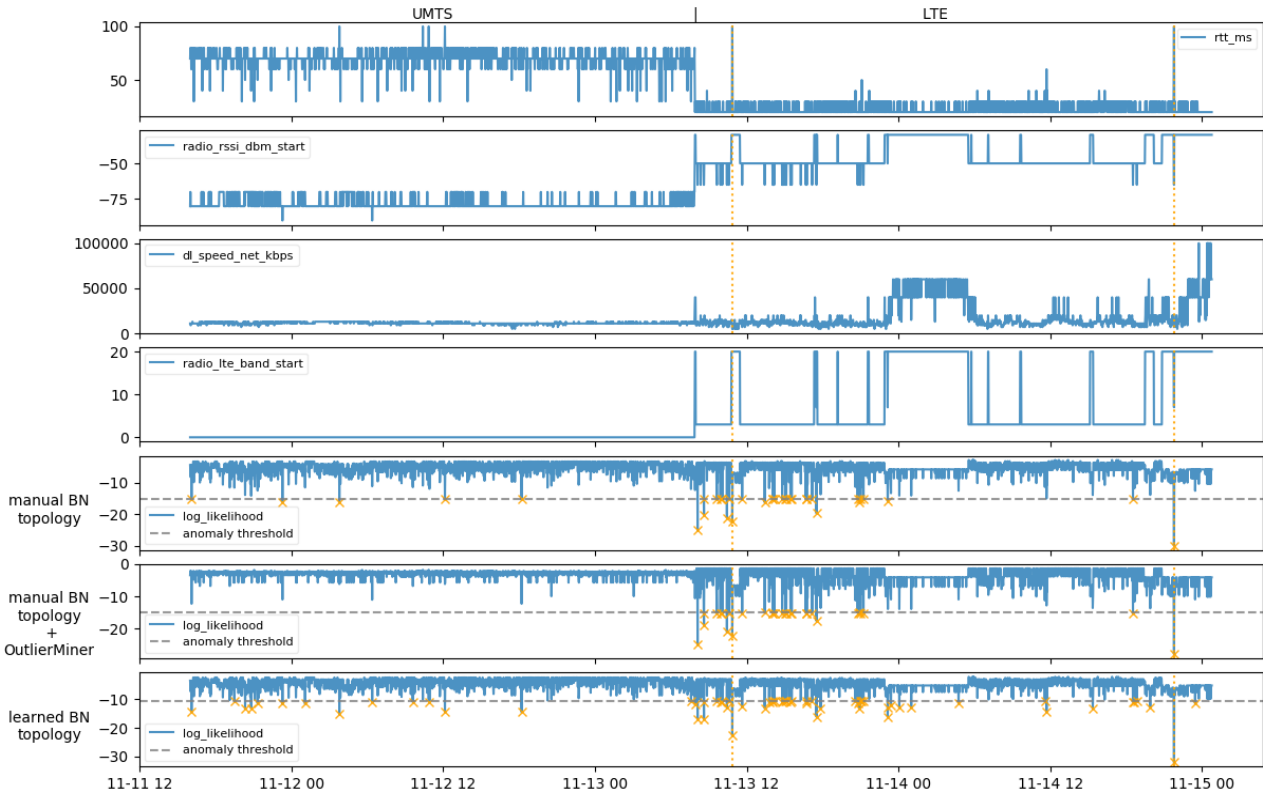

Fig. 5. Results for *testing dataset*. Detected anomalies are marked with orange crosses. Attribute *rtt_delta* is omitted, *radio_access_type_start* is marked on top of the figure. Log-likelihood for learned BN topology combined with *OutlierMiner* rules is omitted because it is visually indistinguishable from learned BN topology alone. Orange dotted vertical lines mark prominent point anomalies.

segment separately. Manual BN topology had a significantly higher variance in radio anomaly segment compared to learned BN topology. The difference between mean values of both segments is larger for manual BN topology (**9.99** compared to **7.29**).

Results in Table I imply that the usage of *OutlierMiner* rules had no significant impact on log-likelihood when using learned BN topology. Using the rules does not affect log-likelihood mean and variance and there was no visual effect either. On the other hand, *OutlierMiner* rules had a significant impact on manual BN topology. Log-likelihood mean value is increased to **-1.93** for normal data segment, which could be anticipated. Using the rules can lead to exclusion of certain terms from log-likelihood computation which effectively increases its final value. *OutlierMiner* rules also increase the separation between the means of normal and radio anomaly segment from **9.99** to **11.3**.

Table II and Fig. 5 represent results for *testing dataset.* There were no known anomalies in the UMTS segment. Both BN topologies successfully detected two prominent anomalies in the LTE segment marked in Fig. 5. Using the *OutlierMiner* rules had no significant impact on the learned BN topology results. According to Table II, using the *OutlierMiner* rules with manual BN topology does affect log-likelihood. UMTS segment variance is decreased whereas LTE segment variance is increased when using the rules. Additionally, UMTS segment mean is increased by **42.0%** whereas LTE segment mean is increased by **27.9%**. The combination of manual BN topology and *OutlierMiner* rules is also the least sensitive with the lowest number of detected anomalies.

We analyzed the performance impact of log-likelihood computation in C++ using the *rbenchmark* R package. The execution time was recorded for *testing dataset* which consisted of 3885 samples. Log-likelihood was computed 200 times for each individual sample. Manual BN topology was used. Benchmarking was performed on a *Fedora Workstation* 28 system with Intel Core i7 3610QM processor, 8GB DDR3 RAM, with R version 3.5.1 and gcc 8.2.1. Given the measured execution time, we computed the number of samples per second each function is able to process:

- LogLik, *bnlearn*[7]: **1100** samples/s, standard deviation: 48 samples/s
- log-likelihood, our implementation in R: **1667** samples/s, standard deviation: 24 samples/s
- log-likelihood, our implementation in C++: **7268** samples/s, standard deviation: 189 samples/s
- log-likelihood with *OutlierMiner* rules, our implementation in C++: **7224** samples/s, standard deviation: 200 samples/s

According to the results above, we can expect a **6.6** times larger data throughput when using C++ log-likelihood implementation instead of *bnlearn.*

VIII. Conclusion

In this paper we introduced a *semi-supervised* approach to anomaly detection. We implemented an anomaly detection system using *bnlearn* R package and compared the results of manual and learned network topology. We implemented *log-likelihood* computation in C++ and analyzed the performance benefits. We also analyzed how the usage of *OutlierMiner* rules can impact an anomaly detection system based on our implementation. We have shown that using the *OutlierMiner* rules can help reduce sensitivity of an anomaly detection system.

Appendix I: Specifications

Specifications for different mobile communication standards can be accessed at:

http://www.3gpp.org/specifications

Older standards are available (GSM, EDGE, UMTS) as well as different LTE releases and the emerging 5G specifications.

Appendix II: Description of Attributes

radio_access_type_start – mobile communications standard used to access the network (LTE or UMTS).

Table II
Testing results for *testing dataset*

BN Topology	With OutlierMiner (*minconf* = 0.1, *maxconf* = 0.8)	*log-likelihood*				Number of detected anomalies
		UMTS segment		LTE segment		
		mean	variance	mean	variance	
manual	yes	-2.95	0.95	-4.00	7.75	29
	no	-5.09	3.11	-5.55	5.35	35
learned	yes	-4.26	3.08	-4.93	4.41	62
	no	-4.26	3.08	-4.93	4.40	62

radio_rssi_dbm_start – Received Signal Strength Indicator. A measurement of power of the received radio signal in *dBm* (decibel-milliwatts).

radio_lte_band_start – frequency band which the measurement probe used to attach itself to the base station.

rtt_ms – Round Trip Time in milliseconds. The time elapsed between dispatching an ICMP echo request message and receiving an ICMP echo response message.

rtt_delta – the difference between Round Trip Times for two different IP network endpoints in milliseconds. Both ICMP request messages were dispatched at the same time.

dl_speed_net_kbps – download speed in kilobits per second.

Acknowledgment

We are grateful to prof. Urban Sedlar and Janez Sterle from Faculty of Electrical Engineering in Ljubljana for providing the data and the assistance while the experiments were performed. Their assistance proved to be invaluable when we were trying to understand the attributes and different events in data. Additionally, we would like to thank prof. Marco Scutari from University of Oxford, Department of Statistics for explaining how *bnlearn* package can be used to compute log-likelihood.

References

[1] M. Ahmed, A. N. Mahmood, J. Hu (2015) A Survey of Network Anomaly Detection Techniques [Online] Available: https://www.sciencedirect.com/science/article/pii/S1084804515002891

[2] S. Babbar, S. Chawla (2006) On Bayesian Network and Outlier Detection [Online] Available: https://www.cse.iitb.ac.in/~comad/2010/ResearchTrack/paper 17.pdf

[3] Ž. Deljac, M. Randić, G. Krčelić (2015) Early detecion of network element outages based on customer trouble calls [Online] Available: https://www.sciencedirect.com/science/article/pii/S016792361500041X

[4] N. Görnitz, M. Kloft, K. Rieck, U. Brefeld (2013) Toward Supervised Anomaly Detection [Online] Available: https://dl.acm.org/citation.cfm?id=2512545

[5] D. Hawkins “Identification of Outliers (monographs on statistics and applied probability),” 1st ed. Netherlands: Springer, 1980

[6] Z. Liu, B. Malone, C. Yuan (2012) Empirical evaluation of scoring functions for Bayesian network model selection [Online] Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3439716/

[7] M. Scutari (2010) Learning Bayesian Networks with the *bnlearn* R package [Online] Available: https://arxiv.org/pdf/0908.3817.pdf