Self-Driven Computer Networks

Simon Klimaschka
Advisor: Fabien Geyer
Seminar Future Internet SS2018
Chair of Network Architectures and Services
Departments of Informatics, Technical University of Munich
Email: simon.klimaschka@tum.de

ABSTRACT

This paper provides an overview about the different kinds of machine learning algorithms that can be used for detecting anomalies in computer networks regarding how the different algorithms work and how they are beneficial for anomaly detection in computer networks. It also provides an overview about which paper uses which machine learning technique and how those techniques worked out. Furthermore, it gives an introduction to anomaly detection with graphs and what advantages graph-based anomaly detection has compared to machine learning based approaches. In the end, the paper presents the implementation of a supervised and an unsupervised machine learning algorithm for anomaly detection from the scikit module and evaluates the used algorithms in terms of overall detection performance and computing time.

Keywords

anomaly detection, machine learning, graph, computer, network, scikit

1. INTRODUCTION

One of the inventions, that brought humanity the most advantages is undoubtedly the internet. While in the beginning very few people used this new achievement, the e-mail traffic grew very fast in the first years. The web pages in these days were very lightweight and hacker attacks not very frequent. But in the past years, more and more services use the internet. Nowadays, nearly everyone owns a smartphone, uploads their data to the cloud, streams movies from services like Netflix or Amazon Prime or can look up nearly every information they want to. Companies also use the internet to be more connected and use it to connect their facilities. The Internet of Things (IoT) will enable devices like light bulbs, refrigerators or a door bell to be connected via the internet. This will result in billions of new devices in our networks. As can be see, the poor security in those devices can result in big attacks like the mirai attack [1]. With so many different devices and very diverse usages, it gets more attractive for hackers to intrude networks and devices. Possible attack scenarios are stealing classified data from company networks and blackmailing them, infiltrating military networks to gather data about upcoming missions, hacking the networks of banks and payment service providers to make money or even distributing malicious software through the whole internet like in the WannaCry attack [2] which affected railway services as well as hospitals. All these attack ideas underline, that we need systems, which can detect anomalies in networks, which often are attacks. To get an idea about how anomaly detection in computer networks works, we will observe two different ideas. In section 2, we will address the different machine learning approaches and how they can be beneficial for anomaly detection. Section 3 is dedicated to how graphs can be used to detect anomalies, as well as comparing graph-based approaches to machine learning approaches with an eye on their respective advantages. In the end, we will evaluate two implementations of a supervised and an unsupervised machine learning approach and compare them regarding their detection accuracy and computation time.

2. ANOMALY DETECTION IN COMPUTER NETWORKS USING MACHINE LEARNING

Anomaly detection plays an important role in computer networks nowadays, especially regarding security and stability. Agrawal and Agrawal [3] define "Anomaly detection [a]s the process of finding the patterns in a dataset whose behavior is not normal on(sic!) expected."
Spotting anomalies means spotting behaviours which do not occur very often and are most likely not wanted. As we want to focus on anomaly detection in computer networks in this paper, this could happen during an attack by hackers or due to malfunctioning hardware or software. As can be assumed, classifying packets is not trivial at all, many factors have to be considered.
Machine learning is very good at tasks like classifying high-dimensional data or reducing high-dimensional data to lower dimensions. To further survey machine learning for anomaly detection, we have to consider the different machine learning techniques at first, regarding how they work, for which cases they will work and if it will be beneficial for anomaly detection in computer networks:

- **Artificial Neural Networks:** Artificial Neural Networks (ANNs) can "approximate any given function" [9], which also includes anything non-mathematical, like image recognition or natural language processing. ANNs consist of so called neurons, which are connected to each other. The neuron is based on the way the neurons in the brains work, hence the similar names. In which manner they are connected depends on the style of the network.
  A feed forward network for example has an input layer, an output layer and an adjustable number of hidden layers. Every neuron in a layer is connected to the

doi: 10.2313/NET-2018-11-1_06

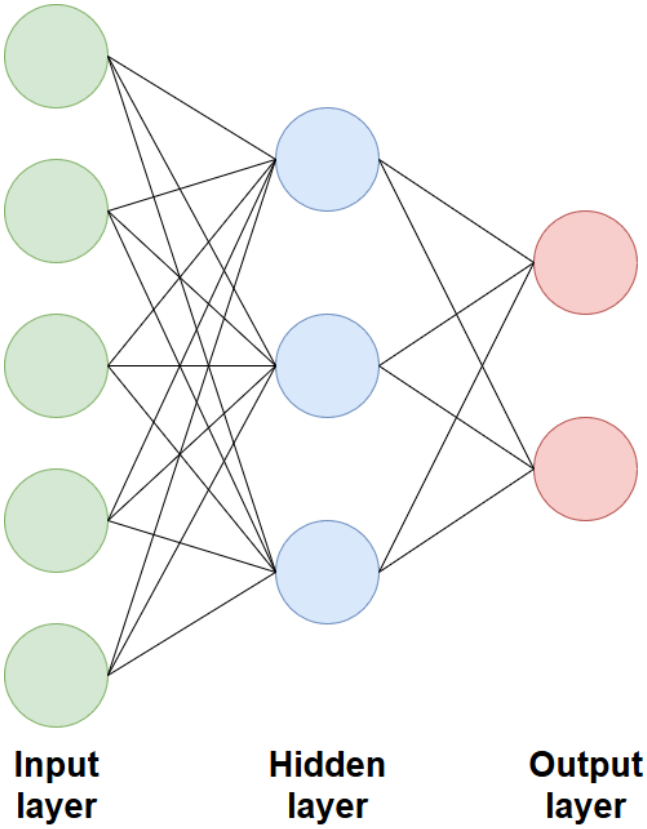

Figure 1: Neural Network

neurons of the anterior and posterior layer, as can be seen in figure 1. Every connection has a corresponding weight, which is altered depending on the usage of the network. Altering the weights of a network to a specific use-case is called training. When fitting an ANN, we use the properties of supervised training:

- In supervised training, a training data set has to consist of the input values and the expected output values. Regarding a specific input, the output of the network is calculated and compared to the expected output. Based on the rules of backpropagation the weights of the network are adapted. For anomaly detection, the ANN could be design with one output neuron and trained to output 1 for anomalies and 0 for normal packages. While the result is exactly what we want, the training process needs a lot of work done in advance for our use-case. Typical training data sets consist of hundreds of thousands of data samples which all need to be classified as normal or abnormal. This is not only a huge workload, different persons also classify the same packages differently, even the same persons tend to change their mind when classifying the same packages. [4, Section 1.2]. Furthermore, anomalies are per definition unusual and uncommon, which makes them even harder to spot. Lippmann and Cunningham [14] used a ANN with selected keywords as inputs. The ANN was able to detect 80% of all occurring anomalies with just about 1 false alarm per day. Palagiri and Chandrika [17] used a neural network trained with data from the DARPA 1999 training data set. While detecting all normal packages correctly, it could only identify 24% of all attacks, 76% were falsely classified as normal. The main problem they had was the low number of occurrences of attacks in the data set. When they trained the network for just one attack, they were able to detect all attacks with a false positive rate of 0%.
  This leads to the main problem with neural networks: They provide excellent results for very specific purposes, like a single attack, but get weaker the more attacks they should be able to recognize. They also struggle with detecting attacks they were not trained for. A solution for this could be creating an own neural network for every attack. This won't solve the new-attack problem, but will increase the rate of detected attacks, although it requires training many times over. Though the training time is relatively long, the detection itself can be done in quasi real-time.
- Unsupervised training, in contrary, doesn't need classified training data, it makes the classification internally by itself. An unsupervised learning technique is described in the subsection Clustering.

- **Fuzzy Rules:** Fuzzy sets are an extension of normal sets. In a conventional set, an object either is or isn't a member of the set. For example, the number 4 is a member of the set 'even numbers' but 5 isn't. Fuzzy logic allows objects to be a member of a set in a certain percentage. In a conventional set, 29°C will be considered as normal, but 30°C already as hot. The difference is not that big, but the outcome is very different. In fuzzy logic, 29°C can be a member of warm with 60% and a member of hot with 40%. For anomaly detection in networks, fuzzy rules have the big downside, that the training cannot be done automatically like with neural networks. Though neural networks could help with for example feature reduction, the selection of features and the definition of the rules has to be done by the system administrator with his experience. This yields the risk, that the administrator forgets certain attacks or might even not be aware of them being existent. How well fuzzy logic detects anomalies is therefore dependent on the system administrator creating the rules. However, the construction of the if-then rules is easier than describing attacks in a normal fashion, as it mimics human thinking [7]. Fuzzy logic can also work with other data mining algorithms and extend them, providing more abstract and flexible methods for creating rules [10]. Thus, fuzzy logic can be powerful when set up correctly.
- **Bayesian Networks:** When working with a network we often know the statistical features between the variables in our network, but keeping an eye over all of them and how they all correlate is not an easy task. Bayesian networks can close this gap and provide an overview over the variables and their relations with a probabilistic graph model, represented by a direct

doi: 10.2313/NET-2018-11-1_06

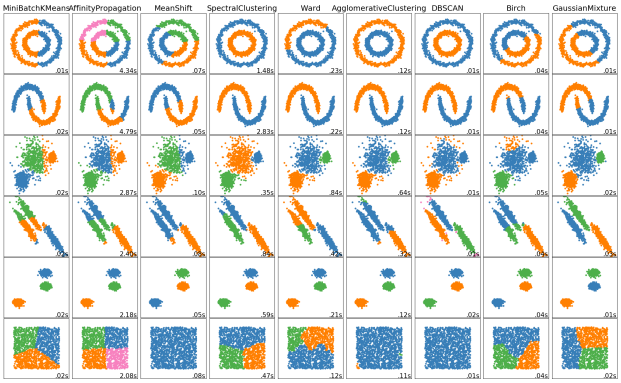

Figure 2: A comparison of the clustering algorithms in scikit-learn [20]

acyclic graph with each node being one system variable and every edge showing that one node is influenced by another. This system could answer questions like with which probability an attack is happening if the system variables have a certain setting [22]. The problem with this technique is that you need to know about the statistical dependencies of your networks variables, which is not given very often, especially not in new networks, as it requires experience in working with networks.

- **Clustering:** This technique doesn't need labeled data, it's an unsupervised algorithm. Clustering is able to discover structures and patterns in any-dimensional data. There exist various methods:
  - Connectivity models group the data points based on the distance between them
  - Distribution models assume, that groups are acquiescent to a statistical distribution
  - Density models use dense regions and connected regions to group data points
  - Graph models sets of connected nodes describe each cluster while every node holds an edge to at minimum another node in the set [5]

  The big advantage for clustering is it's ability to learn from arbitrary data and that it doesn't need a supervisor which defines possible attack scenarios. This freedom also comes with a downside, to get acceptable results we have to choose an appropriate algorithm. As can be seen in figure 2, depending on the organization of the data points, an algorithm could be better than another one. In terms of computing we also should consider, that some algorithms like DBSCAN don't scale very well as they are very memory demanding. Leung and Leckie [13] used the approach fpMAFIA described in their paper, a density- and grid-based high dimensional clustering algorithm. It has a high detection rate, but a rather high false positive rate compared to other approaches like k-nearest neighbour or support vector machines. How well clustering performs is dependent on the choice of the used algorithm and how the data is structured.

- **Ensemble learning:** Ensemble learning refers to the approach of combining several weak learning algorithms to improve their total performance [22]. As it depends on the choice of algorithms and their connection how this technique performs, we will not look any further into it.

- **Evolutionary/Genetic:** Genetic learning is inspired by nature and the principle of evolution. We start with organizing the problem structure in a chromosome like data structure. To solve the problem, the genetic algorithm uses the chromosomes of the best solutions and mixes them together to create new chromosomes. This leads, given some slight randomization, to a better solution every epoch. In computer networks, Khan [11] uses genetic algorithms to develop detection rules. Every chromosome consists of genes like the used service, if the user is logged in, uses a root shell or is a guest. Genetic algorithms have the advantage, that it searches for the best anomaly finding solutions from all directions, whilst needing no prior knowledge of the network. However, this can take a while, the computing power needed is comparatively high [8].

- **Support Vector Machines:** Support Vector Machines (SVMs) are able to classify data points into two different classes. The two classes are divided by a hyperplane, which is outlined by a set of support vectors which have to be members of the training input. In contrast to clustering with the k-nearest neighbours approach, SVMs can handle big dimensions, because an upper bound is set on the margin between the different classes. SVMs have the crucial advantages, that their real-time performance can be better than neural networks and they are easily scalable, in the size of the data points as well as for dimensionality [15]. Mukkamala et al. [15] concluded, that the time to train a SVM is significantly shorter than for neural networks (17.77s and 18min) while delivering the same accuracy in detecting anomalies. The downside with using SVMs is that you can only make binary decisions like is anomalous or not. This could become a problem, if the attacks are very different and we would need to divide into different kinds of anomalies to detect them correctly [15].

- **Self-Organized Maps:** Kohonens Self-Organizing Maps [12] are able to convert many-dimensional data into a two-dimensional map using the internal representation of a map. The method a self-organizing map (SOM) is trained, is related to competitive learning. All points of the map are initialized randomly, then the training data is presented to the map. The map-point, which is fitting the best to the data-point, is pulled towards the data-point. All surrounding map-points are pulled towards the data-point too, but the further they are apart, the less they are pulled. As the training progresses, the points are pulled a little bit less. The SOM creates an internal representation of the given data in the process. In our use-case, the SOM classifies all packet classes, without any prior definitions, just by the input data [21]. A restraint for SOMs is that it only considers the bulk of a traffic class, not the exception that occur seldom, which are no anomalies, but would be classified as one by the SOM [19].

doi: 10.2313/NET-2018-11-1_06

2.1 Conclusion

Anomaly detection with machine learning plays it's advantages if we already have collected a lot of data from the network. This, for example, could be a recording of all packages that are passing the network. The more training data we have, the more accurate our model gets. The training itself can be very costly and tedious, but when trained, it delivers very accurate results. In the last two years hybrid approaches were mainly used, for example combining PSO and K-means with fuzzy logic [10] or combining Naive Bayes and CF-KNN to gain better results than with just one algorithm [16]. A comparison between different implementations in papers can be seen in Table 1.

3. GRAPH BASED ANOMALY DETECTION

3.1 Overview

In section 2 we focused on anomaly detection in data sets which are based on multi-dimensional data points. If we consider, that the relationship between such data points can be important, we should rethink our model. For example, in big networks, the structure of the network and the dependencies are absolutely essential for detecting anomalies. An easy but powerful representation for such cases are graphs. They are described by nodes which are connected with each other by edges. Obviously, our old techniques for detecting anomalies are not compatible with graph-based data, so we need to develop new ideas.

3.2 Graphs and anomaly detection

Before we can discuss how graph based anomaly detection can be beneficial for computer networks, we have to figure out how anomaly detection in graphs works in general.
Akoglu et al. [4] differentiate between anomaly detection in static graphs and in dynamic graphs.
In static graphs, used methods are further divided into structure-based (searching for frequent patterns in the graphs and subgraphs) and community-based (searching for nodes that belong together as a community) methods. For the former, the following approaches exist:

- Feature-based approach: Extracts structural graph-centric features from the graph layout and combines them with other features already known from other sources. This remodels the problem to an anomaly detection problem with multi-dimensional data points, which we already covered in section 2.
- Promixity-based approach: The main goal of this approach is to highlight the closeness of nodes in the graph, as objects that are very close to each other often belong to the same class (e.g if one person is involved in crimes, his or her acquaintances are likely to also be involved).

For the latter, the main idea is trying to find communities, like a group of friends or servers that work closely together, which are densely connected. Detecting anomalies is then basically just finding connections, that are not in between communities.
Static graphs can also have attributes, for example interests of the users of a social network or purposes of a server (http, mail, ftp, ...) for a computer network. The anomaly detection in those graphs are for the most part the same, so the main idea is still spotting structures or communities. For the latter, outliers in a community can now also be spotted, like if there is one smoker in a community of fitness enthusiasts. A new method in attributed static graphs are relational learning based methods, which can detect much more complicated relationships between nodes to group them into normal and anomalous nodes.
In dynamic graphs, we focus on detecting anomalies in dynamic or time-evolving (meaning a sequence) graphs. There exist four ways:

- Feature-based events: In graphs that evolve, consecutive graphs should have similar properties like diameter or degree distribution. This approach considers a time step as anomalous, if the properties change too much. It is very crucial to choose the right properties that will be observed, as they change depending on the problem.
- Decomposition-based events: This approach converts the time-series of graphs into matrices or tensors, which then are interpreted with chosen eigenvectors, eigenvalues or similar indicators.
- Community- or Clustering-based events: Here, the focus is not on monitoring the network as a whole but detecting communities or clusters and watching the structural change in them.
- Window-based events: This approach works with time windows, e.g. a predefined number of previous graphs is taken into account to decide whether the current graph is anomalous or not. They represent the largest category of anomaly detection methods in dynamic graphs.

This is just a very shallow outline of how graph-based anomaly detection works. For more details, look into [4].

3.3 Graph based anomaly detection for computer networks

Computer networks can be easily represented with graphs, as their structure is very similar, with computers and servers acting as nodes and connections between them being edges. Those edges might be weighted with the current load or data flow. We assume, that the weight of edges changes when the computer network is under attack. The two big problems we face are on the one hand, that watching just single nodes and edges will miss all dependencies in the graph, which would destroy one big advantage, graph-based anomaly detection has. On the other hand, big networks with a lot of edges get very expensive to monitor in parallel.
In conclusion, graph based anomaly detection has advantages when having a big structured network and the knowledge, how the structure looks like exactly. This approach is also very powerful when we deal with newly set up networks for which we do not have any previous data.

4. COMPARISON OF THE IMPLEMENTATIONS OF TWO MACHINE LEARNING APPROACHES

doi: 10.2313/NET-2018-11-1_06

Table 1: Comparison of machine learning algorithms used for anomaly detection

Reference	Use-case	Machine learning method	Result
[14]	Detecting attacks with keywords and DARPA Intrusion Detection Data Base [6]	Neural Network	80% of attacks detected with one false alarm per day
[17]	Detecting attacks with DARPA Intrusion Detection Data Base [6]	Neural Network and SOM	24% of all attacks correctly predicted, when trained with one attack 100% prediction rate
[7]	Detecting attacks with own data set	Fuzzy networks	Detected nine TCP port scans and four ping scans in three weeks
[10]	Detecting attacks with generated dataset from CCNx software	Fuzzy anomaly detection based on hybridization of PSO and K-means clustering algorithms over Content-Centric Networks	Detection rate of 100% with a false positive rate of 1,847%
[5]	Detecting attacks with KDD Cup 1999 Data Set	Evolutionary Computations	Detection rate of 94,19% with false positive rate of 5,81%
[13]	Detecting attacks with KDD Cup 1999 Data Set	Cluster detection using fpMAFIA	Performance value of 0,867
[11]	Detecting attacks with KDD Cup 1999 Data Set	Genetic algorithms	Reliability of 94,64% to detect an attack and 1% difference of detecting attacks between training and test data
[15]	Detecting attacks with 1998 DARPA Intrusion Detection Data Set	Support Vector Machine	Detection rate of 99,5% while needing just 17s training time (compared to 18min for neural network)
[19]	Detecting attacks with MIT Lincoln Labratory DARPA Intrusion Detection Evaluation Project data set	Self organizing maps	1,19% false positive rate for DNS, 1,16% false positive rate for HTTP

4.1 The used data set

The data set used for testing the following implementations is the KDD Cup 1999 Data Set. It was used for the Third International Knowledge Discovery and Data Mining Tools Competition to build a network intrusion detector which should be able to detect attacks. The data contains nine weeks of TCP dumps simulating a U.S. Air Force LAN which was under attack to generate the anomalous data. The whole data set is labeled with not just normal or anomalous, it also contains information about the type of the attack. The attacks can be classified into four different categories:

- Denial of service attacks
- Unauthorized access from a remote machine like guessing a password
- Unauthorized access to local superuser privileges
- Surveillance or probing activities like port scanning

The test data also includes attacks which did not occur in the training set to create a more genuine simulation, as in the real world new attacks are discovered all the time.

4.2 Programming language and machine learning library

The most important goal for these implementations were fast prototyping whilst performance could be neglected. The tests should provide an overview about the different approaches and how they perform related to each other, so it's not important that the implementation itself is the fastest. For this reason, Python was chosen as a programming language. It's rather easy to code in Python, providing a lot of built-in features that make preprocessing a lot easier while still providing decent performance. There also exist a lot of libraries for Python that fit our needs.
The chosen machine learning library was scikit-learn [18], as it's available for free and provides packages for classification, regression, clustering or dimensionality reduction and different approaches for preprocessing data.

4.3 Preprocessing

The data from the KDD Cup 1999 Data Set is organized linewise. Every line represents one connection with features like the duration, the used protocol, sent bytes or if it attempted to gain superuser rights. The last element is not a feature, it's the classification of the connection. They are 'normal' in most cases, but also contain attacks like 'buffer_overflow', 'guess_passwd' or 'rootkit'. Not only the classification itself is a string, but also the second, third and fourth feature are no numeric values but a string. Machine learning algorithms can only work with numeric values, so we have to convert all strings to numeric values. At first, all string valued features were taken out of the training data and collected in one array each. From the preprocessing package, we used the LabelEncoder, which provides a normalization of labels. It can also convert hashable and comparable strings into normalized numeric values by hashing them at first and them normalizing them. Every features LabelEncoder instance was fit to the data of the repective feature and then transformed into a numeric value which can be used for machine

doi: 10.2313/NET-2018-11-1_06

learning. After the process, the features were add back into the data set.

4.4 The testing system

The code was tested on a laptop from ASUS, with a Intel Core i5-7200U CPU and 24 GB of memory. The data set was saved on a SSD, the code was also run on a SSD.

4.5 A supervised learning approach

For the supervised learning approach, a standard neural network should be evaluated. From the skicit-learn package, the MLPClassifier was chosen. It implements a multi layer perceptron (MLP) using backpropagation for training, which describes the approach we made in section 2 for the supervised network. The MLPClassifier has some parameters which can be altered, like the use of which activation function for the hidden layers, the used solver, the size of the hidden layers or if it should implement early stopping. Early stopping refers to the technique of reserving 10% of the training data for validation purposes and stopping the training if the results of the validation data set did not improve for more than two epochs.
To find the best parameters for anomaly detection, the algorithm was fed with the training data from the KDD Cup 1999 Data Set and validated with it's testing data set. The results for the different parameters can be found in table 2. One can easily see, that more neurons does not mean better accuracy, but introducing a second layer resulted in just a fifth false alarms. Sgd performed not as well as adam, while being a lot better in recognizing the anomalies, it generates a lot of false alarms. The activation functions didn't have a big impact on the results, just the identity function generates worse results. It took about 4,5 minutes to run on the testing system. The following parameters were tested:

- **Solver:** The solver for weight optimization, choosing from lbfgs, which wasn't used as it's for smaller data sets, sgd, the stochastic gradient and adam, a stochastic-gradient optimizer which works good for bigger data sets.
- **Hidden Layer Size:** The hidden layers are denoted with parentheses and commas, where every part represents the number of neurons in a layer, e.g. (30,10,5) are 30 neurons in the first hidden layer, 10 neurons in the second and 5 in the third.
- **Activation Function:** The activation function used for the neurons of the hidden layer, with the following possibilities: Identity (f(x) = x), Logistic (logistic sigmoid function), Tanh (the hyberbolic tan function) and relu (the rectified linear unit function)
- **Correct Predictions:** How many out of the 311.030 connections were predicted correctly
- **False negatives:** How many percent of the wrong predictions were classified as normal, but are an attack
- **False positives:** How many percent of the wrong predictions were classified as attack, but are a normal connection

4.6 An unsupervised learning approach

For the unsupervised learning approach, the k-means algorithm from the scikit-learn package was chosen. Other algorithms like DBSCAN or Birch, which would also make sense for our use-case and can handle high dimensional data were not able to run on the previously describe testing system as they were too memory demanding.
K-means tries to cluster the data into any desired number of clusters of equal variance. The goal of the algorithm is to choose the centroids of the clusters in such a way, that the distance to all points belonging to it will be as small as possible. This raises a few problems when having elongated clusters or other irregular shapes, because the algorithm works best for convex blobs of points. For high dimensions, the euclidian distance between points become more and more irrelevant, which is also known as the curse of dimensionality.
K-means doesn't have much parameters that can be changed or would be useful changing. The number of clusters for our example is fixed, as we only want to distinguish between normal and anomalous. We could also take the number of different attacks as the number of clusters and it could be working better, but in a normal environment we don't know how many attacks occur in our training sample. If we analyzed the training samples the major advantage that we don't have to look into the data would be completely gone. The programmed algorithm ends with 60.593 wrong predictions. It took about 4,5 minutes on the testing system.

4.7 Comparison of the two approaches

The supervised algorithm outperformed the unsupervised algorithm in nearly every setting. This was expectable, the supervised algorithm knows how to trim it's network to get the best results while the unsupervised algorithm can only depend on the structural organization of the data points when detecting anomalies. We also have to consider the fact, that in a real world environment, we would have to classify the training data before being able to use the supervised algorithm. While the best setup for the supervised variant had about 92% correct predictions (23000 wrong predictions), the unsupervised variant is just 37.000 worse and works without classifying the data beforehand. Nevertheless, with feature reduction or other related techniques, the results of the unsupervised algorithm can be enhanced.

5. CONCLUSION

The paper provides a survey about the different types of anomaly detection existent for computer networks. At first, the different machine learning techniques were explained and evaluated in the terms of their usability in anomaly detection for computer networks. Then we examined graph based anomaly detection in terms of how it works and what benefits it has for anomaly detection in computer networks, especially in comparison to machine learning based approaches. To round this off, we conclude the survey with an overview of machine learning methods used in different papers with their respective outcomes. In the end, a example data set was taken and two algorithms were evaluated, a supervised machine learning algorithm and a unsupervised one. Furthermore the supervised algorithm was tested with different parameters. Nearly all settings outperformed the unsupervised algorithm. Generally, there are a lot of different algorithms that can be used for anomaly detection, but there

doi: 10.2313/NET-2018-11-1_06

Table 2: Result of the supervised learning implementation

Solver	Hidden Layer Size	Activation Function	Correct Predictions	False Negatives	False Positives
adam	(30)	relu	92.33%	95.22%	4.78%
adam	(15)	relu	85.09%	97.78%	2.22%
adam	(45)	relu	91.96%	95.88%	4.12%
adam	(30,10)	relu	92.11%	98.85%	1.15%
adam	(30,20,5)	relu	91.88%	98.82%	1.18%
adam	(80,55,30,10,5)	relu	92.00%	98.88%	1.12%
adam	(30)	identity	81.81%	97.82%	2.18%
adam	(30,20,5)	identity	89.48%	96.71%	3.29%
adam	(30)	logistic	91.89%	96.43%	3.57%
adam	(30,20,5)	logistic	91.94%	97.99%	2.01%
adam	(30)	tanh	91.91%	96.40%	3.60%
adam	(30,20,5)	tanh	91.73%	96.52%	3.48%
sgd	(30)	tanh	90.91%	96.00%	4.00%
sgd	(30,20,5)	tanh	81.39%	98.40%	1.60%

is no best algorithm for all cases. When selecting one, we always have to think about the structure of the data and what our goal is. Newer papers also try to combine different algorithms to use their best properties and combine them with for example fuzzy logic. For future work, one could look into how the different algorithms could be combined the best and which difficulties occur.

6. REFERENCES

[1] The mirai botnet: all about the latest malware ddos attack type. `https://www.corero.com/resources/ddos-attack-types/mirai-botnet-ddos-attack.html`. Accessed: 31.03.2018.

[2] Wannacry ransomware attack summary. `https://www.dataprotectionreport.com/2017/05/wannacry-ransomware-attack-summary/`. Accessed: 31.03.2018.

[3] S. Agrawal and J. Agrawal. Survey on anomaly detection using data mining techniques. *Procedia Computer Science*, 60:708 – 713, 2015. Knowledge-Based and Intelligent Information Engineering Systems 19th Annual Conference, KES-2015, Singapore, September 2015 Proceedings.

[4] L. Akoglu, H. Tong, and D. Koutra. Graph based anomaly detection and description: a survey. *Data Mining and Knowledge Discovery*, 29(3):626–688, 2015.

[5] A. L. Buczak and E. Guven. A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications Surveys Tutorials*, 18(2):1153–1176, Secondquarter 2016.

[6] R. K. Cunningham, R. P. Lippmann, D. J. Fried, S. L. Garfinkel, I. Graf, K. R. Kendall, S. E. Webster, D. Wyschogrod, and M. A. Zissman. Evaluating intrusion detection systems without attacking your friends: The 1998 darpa intrusion detection evaluation. Technical report, MASSACHUSETTS INST OF TECH LEXINGTON LINCOLN LAB, 1999.

[7] J. E. Dickerson and J. A. Dickerson. Fuzzy network profiling for intrusion detection. In *Fuzzy Information Processing Society, 2000. NAFIPS. 19th International Conference of the North American*, pages 301–306. IEEE, 2000.

[8] P. Garcia-Teodoro, J. Diaz-Verdejo, G. Maciá-Fernández, and E. Vázquez. Anomaly-based network intrusion detection: Techniques, systems and challenges. *computers & security*, 28(1-2):18–28, 2009.

[9] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359 – 366, 1989.

[10] A. Karami and M. Guerrero-Zapata. A fuzzy anomaly detection system based on hybrid pso-kmeans algorithm in content-centric networks. *Neurocomputing*, 149:1253 – 1269, 2015.

[11] M. S. A. Khan. Rule based network intrusion detection using genetic algorithm. *International Journal of Computer Applications*, 18(8):26–29, 2011.

[12] T. Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, Sep 1990.

[13] K. Leung and C. Leckie. Unsupervised anomaly detection in network intrusion detection using clusters. In *Proceedings of the Twenty-eighth Australasian conference on Computer Science-Volume 38*, pages 333–342. Australian Computer Society, Inc., 2005.

[14] R. P. Lippmann and R. K. Cunningham. Improving intrusion detection performance using keyword selection and neural networks. *Computer networks*, 34(4):597–603, 2000.

[15] S. Mukkamala, G. Janoski, and A. Sung. Intrusion detection using neural networks and support vector machines. In *Neural Networks, 2002. IJCNN'02. Proceedings of the 2002 International Joint Conference on*, volume 2, pages 1702–1707. IEEE, 2002.

[16] H. H. Pajouh, G. Dastghaibyfard, and S. Hashemi. Two-tier network anomaly detection model: a machine learning approach. *Journal of Intelligent Information Systems*, 48(1):61–74, 2017.

[17] C. Palagiri. Network-based intrusion detection using neural networks. *epartment of Computer Science Rensselaer Polytechnic Institute Troy, New York*, pages 12180–3590, 2002.

[18] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and

E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[19] M. Ramadas, S. Ostermann, and B. Tjaden. Detecting anomalous network traffic with self-organizing maps. In *International Workshop on Recent Advances in Intrusion Detection*, pages 36–54. Springer, 2003.

[20] scikit-learn developers. *A comparison of the clustering algorithms in scikit-learn, available http://scikit-learn.org/stable/modules/clustering.htmlclustering (accessed on 31.03.2018).*

[21] T. Shon and J. Moon. A hybrid machine learning approach to network anomaly detection. *Information Sciences*, 177(18):3799–3821, 2007.

[22] C.-F. Tsai, Y.-F. Hsu, C.-Y. Lin, and W.-Y. Lin. Intrusion detection by machine learning: A review. *Expert Systems with Applications*, 36(10):11994–12000, 2009.

doi: 10.2313/NET-2018-11-1_06