UNIVERSITY OF HASSELT

BACHELOR THESIS

Machine learning techniques for flow-based network intrusion detection systems

Author:
Axel FAES

Supervisor:
Prof. Dr. Peter QUAX
Prof. Dr. Wim LAMOTTE
Bram BONNE
Pieter ROBYNS

Bachelorproef voorgedragen tot het behalen van de graad van bachelor in de informatica/ICT/kennistechnologie

A thesis submitted in fulfillment of the requirements for the degree of Bachelor of Science

in the

Networks and Security
Computer Science

April, 2016

Declaration of Authorship

I, Axel FAES, declare that this thesis titled, "Machine learning techniques for flow-based network intrusion detection systems" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a bachelor degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: ______________________________

Date: ______________________________

UNIVERSITY OF HASSELT

Abstract

Wetenschappen
Computer Science

Bachelor of Science

Machine learning techniques for flow-based network intrusion detection systems

by Axel FAES

Acknowledgements

Contents

List of Figures

List of Tables

List of Abbreviations

IDS	**I**ntrusion **D**etection **S**ystem
IPS	**I**ntrusion **P**revention **S**ystem
IDPS	**I**ntrusion **D**etection (and) **P**revention **S**ystem
NIDS	**N**etwork (based) **I**ntrusion **D**etection **S**ystem
HIDS	**H**ost (based) **I**ntrusion **D**etection **S**ystem
DDOS	**D**tributed **D**enial **o**f **S**ervice
ML	**M**achine **L**earning
MSE	**M**inimum **S**quared **E**rror (function)

List of Symbols

$H_0(x)$	Hypothesis function
$J(\theta)$	Cost function

Nederlandstalige samenvatting

Chapter 1

Introduction

The internet is constantly growing and new network sevices arise constantly. Sensitive data is also increasingly being stored digitally. All these new services could contain security flaws which could leak private data, such as passwords or other sensitive data. This means that security flaws become more and more important. Considering this, it becomes more important to be able to detect and prevent attacks on network systems.

Intrusion detection systems are used for this purpose.

Chapter 2

Intrusion detection systems

An intrusion detection system is a system which tries to determine whether a system is under attack, to detect intrusions within a system. There are different types of intrusion detection systems or IDS. There are network-based intrusion detection systems and host-based intrusion detection systems. This thesis uses machine learning techniques to detect malicious network behaviour. As such only network-based intrusion detection systems are covered.

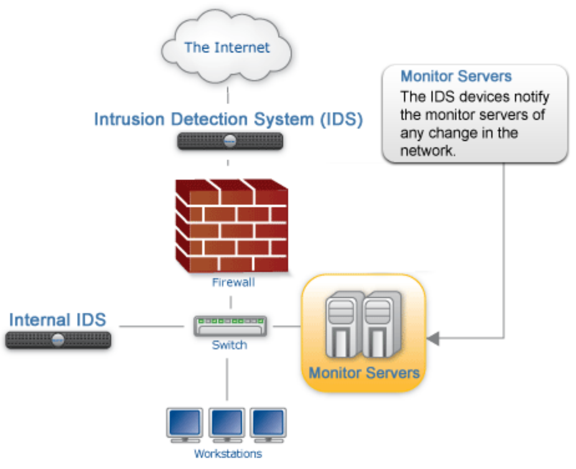
The Internet
Intrusion Detection System (IDS)
Monitor Servers
The IDS devices notify the monitor servers of any change in the network.
Firewall
Internal IDS
Switch
Monitor Servers
Workstations

FIGURE 2.1: An IDS can for example be placed within the network or just before the network.

2.0.1 Host-based Intrusion Detection Systems

Host-based intrusion detection systems are systems that monitor the device on which they are installed. The way they monitor the system can range from monitoring the state of the main system through log files, to monitoring program execution. In this way they can be quite indistinguishable from Anti-Virus programs.

2.0.2 Network-based Intrusion Detection Systems

Network-based intrusion detection systems are placed at certain points within a network in order to monitor traffic from and to devices within the network. The system can analyse the traffic using multiple techniques to determine whether the data is malicious. There are two different ways to analyse the network data. The analysis can be packet-based or flow-based.

Packet-based analysis uses the entire packet including the headers and payload. An intrusion detection system that uses packet-based analysis is called a packet-based network intrusion detection system. The advantage of this type of analysis is that there is a lot of data to work with. Every single byte of the packet could be used to determine whether the packet is malicious or not. The disadvantage is immediately obvious once we look at networks through which a lot of data passes, such as data centers. Analysing every byte is very work-intensive and near impossible to do in such environments. [1]

Flow-based analysis doesn't use individual packets but uses general data about network flows. An intrusion detection system that uses flow-based analysis is called a flow-based network intrusion detection system. A flow is defined as a single connection between the host and another device. A flow can be defined using a (source_IP, destination_IP, source_port, destination_port) tuple. However flowdata also contains other information such as the duration of the connection, the start time, the amount of bytes and/or packets within the flow. Flow data can even contain data such as the amount of SYN packets within the flow. This could be useful to detect SYN overflow attacks. However not every flow collector collects this data Since flow data is much more compact than all the individual packets, it is much more feasable for data centers to use flow-based intrusion detection systems.

2.0.3 Intrusion Prevention Systems

An intrusion prevention system or IPS/IDPS is an intrusion detection system that also has to ability to prevent attacks. An IDS does not necessarily need to be able to detect attacks at the exact moment they occur, although it is preferred. An IPS needs to be able to detect attacks real-time since it also needs to be able to prevent these attacks. For network attacks these prevention actions could be closing the connection, blocking an IP, limiting the data throughput.

2.1 Detection

There are mutliple different methods to detect intrusions. There are **Signatures based methods** and there are **Anomaly Based** methods. Both of these methods have their own strengths and weaknesses. [2]

2.1.1 Signature based methods

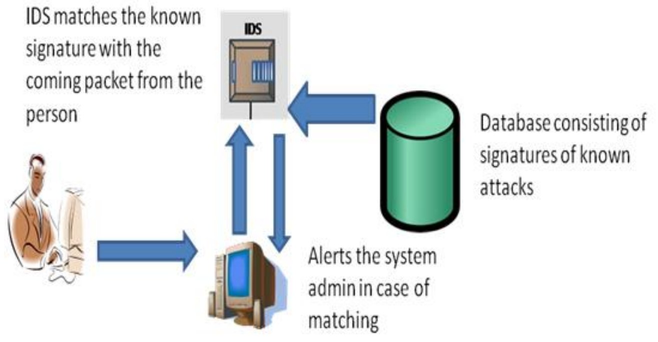
IDS matches the known signature with the coming packet from the person
IDS
Database consisting of signatures of known attacks
Alerts the system admin in case of matching

FIGURE 2.2: An Signature-based intrusion detection system.

Signature based methods compare so called "signatures" with an existing database of signatures. An packet or flow record is decomposed into features that together construct a signature. If the signature of an incoming flow or packet matches with a signature in the database, it is flagged as malicious. Signature-based methods have little overhead in both computation and preprocessing as it only tries to match incoming signatures to known signatures in the database. Because it only compares signatures, it is easy to deploy within a network. The system does not need to learn what the traffic within a network looks like.

Signature based methods are very effective against known attacks. New attacks cannot be detected unless the database is updated with new signatures. It is also for attackers to avoid being caught by signature based methods, only slight modification of the "signature" is required in order to bypass the exact matching. Updating the signature database requires a lot of technical effort, since new attacks are discovered all the time.

2.1.2 Anomaly based methods

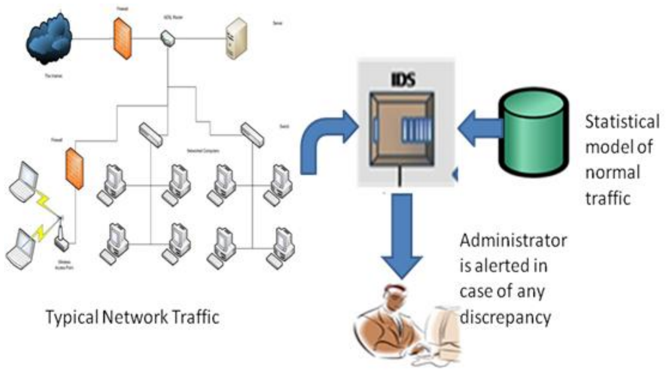
IDS
Statistical model of normal traffic
Administrator is alerted in case of any discrepancy
Typical Network Traffic

FIGURE 2.3: An Anomaly-based intrusion detection system.

Anomaly based methods, also called Behaviour based methods are methods in which the IDS tries to model the behaviour of network traffic. When an incoming packet deviates from this model, it is flagged as malicious and an alert is send. Because they use a statistical model of normal behaviour, they should be able to detect all deviations from this normal behaviour. As a result, new attacks that deviate to much from normal behaviour are detected aswell.

Since a model of the network traffic needs to be created, the system cannot be deployed into a network and be expected to work. The system needs to learn the behaviour of the network traffic. Problems, such as generating a lot of false positive alarms, can arise when training data includes mistakes, such as misclassifications.

Machine learning algorithms can be used as a anomaly based method. Machine learning techniques have the ability to learn from data and decide whether new data is malicous.

2.2 Existing IDS

2.2.1 Alienvault

2.2.2 SNORT

Chapter 3

Attack Classification

An intrusion detection system can use multiple methods to detect malicious behaviour. In order to make the IDS as effective as possible, the exact classifications of attacks that can be detected need to be known.

3.1 Classification

There are several types of attacks that can occur. Some of these attacks occur from within the network, other from outside the network. The exact classifications are not mutually exclusive. Some types of malware utilise network attacks. However it is important to make a distinction between these attacks. Every attack is identified by different characteristics. Knowing these characteristics is useful to be able to tweak the IDS to make identification more effective.

3.2 Network attacks

There are **Physical attacks**, these are attacks which attempt to destroy physical equipment and hardware. **Buffer overflows** are attacks that try to execute arbitritairy code or crash a process by overflowing a buffer on the targeted system. **Password attacks** attempts to break into a system by gaining the password that the system uses. The simplest password attacks are brute-force password crackers. **DDOS** attacks are attacks which attempt to make a network resource temporarily or permanently unavailable for the users of that resource. An attack could happen by flooding a system with TCP SYN packets. **Network scans** are information gathering attacks. They do not cause any damage by themselves but usually serve the purpose to gather information about a system that could be used in further attacks. Network traffic sniffing or port scans are examples of network scans. [3]

3.3 Malware

There are several types of malware. There are four distinct categories of malware. There are **botnets**, **viruses**, **trojan horses** and **worms**. Malware are actual programs that infect a system to execute a specific task. The task of the malware defines which category the malware belongs in.

Trojan horses are programs disguised as harmless applications but contain malicious code. **Worms** are programs that replicate themselves among a network. They can spread extremely fast. **Viruses** are similar to worms.

However they only replicate themselves on the infected host computer. Thus they require user interaction in order to be spread around a network. The virus can accomplish this by attaching itself to an email-attachment, embed itself within an executable, etc.

Botnets is malware that causes infected computers to become "slaves" to the master. An infected computer is controlled externally by the bot-master without the knowledge of the owner of the infected computer. The bot-master can use the distributed network of "slave" computer to perform other malicious tasks, such as performing an DDOS attack. [3]

3.4 Detection

An NIDS only monitors the network. As such not every attack can be detected by an NIDS. Only the attacks that actually use the network can be detected. Flow-based IDS have the additional constraint that they can only use flow data. This further limits the attacks that can be detected. The attacks that can be detected using a flow-based network intrusion detection systems are:

- DDOS
- Network scans
- Worms
- Botnets

Other attacks either do not use network communication, or they are not visible within the header information of network traffic. In order to detect other attacks, including **viruses**, **trojan horses** and **Buffer overflows**, other detection systems such as HIDS or Packet-based NIDS should be used.

3.4.1 Distributed Denial of Service

A distributed denial of service can be detected by the amount of data that is being received. However, there are many different types of DDoS attacks. There are ICMP floods, SYN floods, etc. These attacks can be described in terms of traffic patterns. A traffic pattern is expressed in a couple features. These features include the number of flows and packets, the packet size, and the total bandwidth used during the traffic. For example UDP flooding can be characterised by a traffic pattern which contains a lot of packets. These patterns can be searched for during the detection phase. [4]

3.4.2 Network scans

There are three categories of network scans.

- Horizontal scans: a single port is scanned across many different devices.
- Vertical scan: several different ports are scanned on a single device
- Block scan: a combination of both a vertical and a horizontal scan.

Scans can also be described using traffic patterns. They are characterised with a high number of flow and a low number of packets. These can again be used to detect whether a vertical or horizontal scan occurs. [3] [4]

3.4.3 Worms

Worms exhibit different behaviour depending on their current state. There are two different states, a target discovery state and a transfer state. In the target discovery state, the worm explores the network to find vulnerabilities and a host to infect. During the transfer state, the worm actually transfers itself to the targeted host. The Sapphire/Slammer worm is an example of this type of behaviour. [5]

Since transfering of the worm itself happens within the payload data, a flow-based NIDS cannot detect this state. The target discovery state can be detected. Worms use techniques similar to network scans in order to find vulnerable hosts. So similar detection techniques can be used to detect worms. [6]

3.4.4 Botnets

Botnets usually consist out of a huge amount of infected slaves controlled by a central bot-master. Locating the individual infected slaves and isolating them is a difficult problem but is also insignificant due to the huge amount of remaining slaves. Detecting the bot-master and isolating that device is key to taking down a botnet. However indentifing botnet behaviour is a far more difficult problem than detection other types of malicious activities. [7] Malicious behaviour alone is not enough to detect botnets.

Botnets often use IRC channels in order to communicate between slaves and the bot-master. These can be indentified using flows since they often use specific ports. It is possible to use a method that does not require specific port numbers. This requires flows including extra information such as the number of packets for which the PUSH flag is set.

Chapter 4

Machine learning

4.1 What is machine learning

Machine learning is a subfield of Computer Science. It is a type of Artificial Intelligence which allows programs to learn and find patterns within data. [8]

Machine learning explores algorithms that can learn from and make predictions on data. These algorithms are called machine learning algorithms. A machine learning algorithm has to learn before it can be used to make predictions on data. **Learning** means that the algorithm has to be shown several examples of data and what the correct predictions for these examples would be. The amount of examples that have to be shown to the algorithm can be within the range of several thousands.

Once the machine learning algorithm has learned from the data, it can be used to make predictions on other data. For example, machine learning can be used in order to watch the heartrate of patients at a hospital. During the learning phase, the machine learning algorithm is shown the heartrate of a patient and the current time. After learning is done, the machine learning algorithm can predict what the heartrate of that patient should be based on the current time. This can be used to determine whether the patients heartrate is normal by comparing the predicted heartrate and the real heartrate.

There are two classes of machine learning algorithms. There is **supervised** learning and **unsupervised** learning. Supervised learning is trained using labeled data. Labeled data is data which consists of input data and the corresponding output data. Unsupervised learning uses unlabeled data. The data used to train machine learning algorithms is called a **training set**.

Supervised learning is the type of learning when the data presented to the learning algorithm is labeled. This means that the user of the learning algorithm can already make distinctions within the presented data. This is useful for regression or classification problems. Regression is explained in Section 4.2 and Classification is explained in Section 4.3.

Unsupervised learning is the type of learning when the data presented to the learning algorithm is unlabeled. The learning algorithm has to find structure in the given data.

A **training sample** is a data point in an available training set that is used in a predictive modeling task. For example, if machine learning is applied to spam filtering, the training set is a collection of emails, some of which are spam emails. A training sample would be a single email. Alternative names are a training example or training instance.

Machine learning is applied for predictive modeling. In predictive modeling, a particular process or event is modeled. Using a training set, the model tries to learn or approximate a particular function that, for example, lets us distinguish spam from non-spam email. This function is called the target function. The function the model makes to approximate the target function is called the **hypothesis**.

To model the particular process or event, one or more **features** are extracted. A feature is an individual property. All features combined define the model of the particular process of event. In the example of mails, a feature could be the textbody, or it could be the sender of the email. It could even be individual characters which appear in the email. Which features are chosen is completely dependent on which problem is being solved.

This chapter gives an introduction to the mathematical background of supervised machine learning. It starts with linear regression which is a simple category of machine learning algorithms. The principles behind lineair regression are the same as for other machine learning algorithms. Afterwards logistic regression is used to explain how classification works within machine learning. It begins by explaining what exactly the hypothesis is and how it can be constructed.

4.2 Lineair Regression

Lineair regression is a statistical approach to model the relationship between an "output" value y and one or more "input" values $x_1 ... x_n$. It belongs to the category of supervised learning. An example of this can be seen in Figure 4.1. The black dots represent the data to be modeled. The blue line is the model. This example only has one "input" value. This specific case of regression is called simple linear regression. [9].

4.2.1 Hypothesis

Machine learning relies heavily on a hypothesis. This is a function that transforms a given input to the machine learning algorithm into the required output. It is a function that tries to model the target function, it tries to give a function which fits the input data. When only one feature is considered, a hypothesis is a function of the form [10]:

$$H_0(x) = \theta_0 + \theta_1 * x \tag{4.1}$$

For example, we could use the grades of high school students to predict their chance of success at university. The $x - axis$ represents the grades (on a scale from 0 to 10) for students and the $y - axis$ is the chance of success. Given is input data (the black points) and from that data a hypothesis (the blue line) is constructed.

The hypothesis function can be generalised for N features. It generally has the form:

$$H_0(x) = \theta_0 x_0 + \theta_1 x_1 + ... + \theta_n x_n \tag{4.2}$$

x_0 always has the value 1. The θ values and the x values can be represented using vectors. Now the hypothesis function can be written in a more convenient way:

$$H_0(x) = \theta^T X \tag{4.3}$$

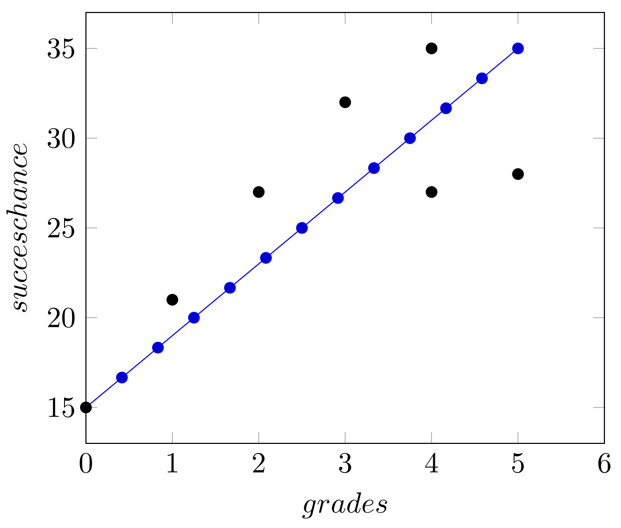

FIGURE 4.1: Lineair Regression

The hypothesis is the core of a machine learning algorithm. During the training process, the algorithm learns what the hypothesis looks like. More specifically, it tries to find correct values for the vector θ (as seen in equation 4.3. Once the hypothesis has been determined, the difficult work is done. The hypothesis forms the predictive model. The prediction itself is straightforward. A new datapoint is entered into the hypothesis equation and it returns the predicted value.

4.2.2 Cost function

In order to construct a good hypothesis function, good values of θ have to be chosen. This can be seen as a minimization problem. The difference between any output y and $H_0(x)$ has to be minimized. More concretely, the squared difference has to be minimized. This is the MSE, "Minimum Squared Error" function or also called the cost function. The notation $x^{(i)}$ and $y^{(i)}$ is to denote the ith training sample. With dataset size m, the MSE is [10]:

$$J(\theta) = \frac{\sum_{i=1}^{m} (H_0(x^{(i)}) - y^{(i)})^2}{2m} \tag{4.4}$$

Seeing this equation already makes it seem that the number of training samples is important. From statistics, it is known that the population of data points, in this case training samples, have to be large enough. This problem is addressed later in Section 4.4.

4.2.3 Gradient descent

To solve the minimization problem, the first step is to start with θ and keep changing the values to minimize $J(\theta)$, this is an iterative approach. Gradient descent is such an algorithm that can be used to find a solution to the minimization problem. Gradient descent is an algorithm that uses the gradient or derivative of a function to find a local minimum of that function.

In order to easily explain the gradient descent algorithm, a analogy to a ball rolling down a hill can be made. Let's say have a landscape such as in Figure 4.2. The horizontal location of the ball represents the values of θ, the height of the ball represents the MSE. We want to get the ball to the lowest possible point, in order to minimize the MSE. To do this, the ball rolls down the hill until it comes to rest at the lowest point. Every iteration of gradient descent will bring the ball closer to the lowest point. [11]

Ofcourse, when a ball rolls down a hill, it does not just roll down and immediately stop. The ball takes a path similar as described in Figure 4.2. When the ball finally comes to rest, it can be said that the algorithm converges to that horizontal position.

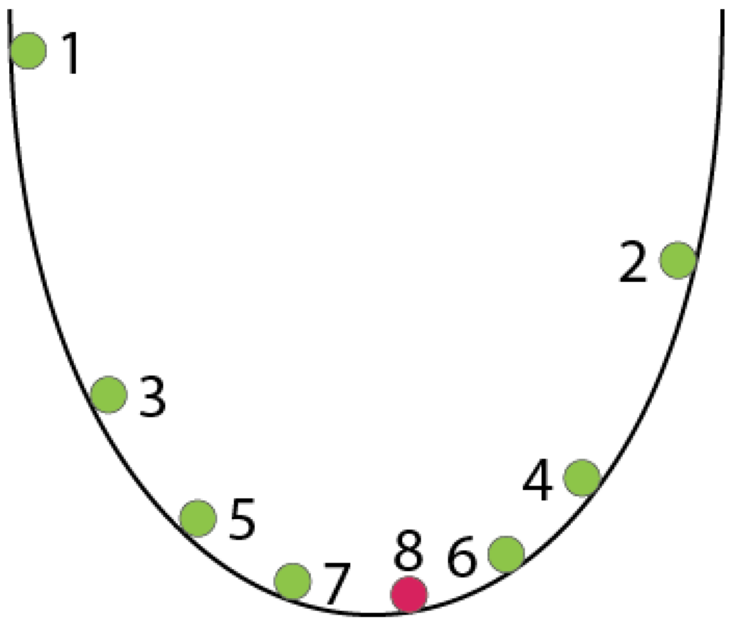

FIGURE 4.2: Iterations of gradient descent. [12]

A gradient descent algorithm does this using the following algorithm with a simultanious update for all values of θ [10]:

```
repeat until convergence {
```

$$\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta) \tag{4.5}$$

```
}
```

$$\frac{\partial}{\partial \theta_j} J(\theta) = \frac{\sum_{i=1}^{m} (H_0(x^{(i)}) - y^{(i)}) * x_j^{(i)}}{m} \tag{4.6}$$

The equation uses the derivative. A derivative indicates the rate of change, in this case, how steep the hill is. If the hill is very steep, then we move the ball a lot more than when the ground beneath the ball is almost flat.

α is the learning rate of the gradient descent. The value of α describes how fast the gradient descent algorithm approaches the local mimimum. If α is too small, the gradient descent can be very slow. In the other case, if α is too large, the gradient descent can overshoot the local mimimum. It may fail to converge and could even diverge.

To go back to the example of the rolling ball, having α too large, could be seen as having the ball roll down the hill extremely fast. Because of this, the ball could come to rest at another point in the landscape, it might have flown over the hill.

The value of α does not need to change during the gradient descent, since the closer the gradient descent gets to the local mimimum, the smaller the derivative becomes, and smaller steps will be taken. If θ_j is already a local minimum, the derivative is 0 and the gradient descent will not change the value of θ_j.

4.2.4 Normal equation

There is another method that could be used in place of gradient descent, a normal equation. This is a method to solve for θ analytically. But this method becomes slow when there are a lot of features. X is a m X n matrix constructed by putting all training samples (vectors of features) together. The notation $x^{(i)}$ and $y^{(i)}$ is used to denote the ith training sample. With dataset size is denoted by m.

$$\theta = (X^T X)^{-1} X^T y \tag{4.7}$$

But what happens when $(X^T X)$ is non-invertible, or singular? This means there are redundant features or more features than training samples. There are mathematical models, such as the pseudo-inverse to still compute a correct result. There are still other methods that could replace gradient descent, such as conjugate gradient, BFGS and L-BFGS. Thinking about matrices and non-invertibility can help to be able to understand how features relate to eachother.

4.2.5 Feature scaling

The main idea behind feature scaling is to make sure that the different features are on a different scale. The reason behind this is to optimize the gradient descent algorithm. The gradient descent will converge much more quickly with feature scaling. With feature scaling, the partial derivative will be larger and because the features are in scale, the values of θ will also be similar in scale.

When one feature is very large and another is very small, the values of θ need to be small for the first feature and larger for the second, since the cost function needs to be minimized. This means that the values of θ need to change more and thus, requires more iterations for the gradient descent algorithm. The range that should approximately be used is $-1 < x < 1$.

Mean normalization could be used. This replaces each x_i with $\frac{x_i - \mu_i}{s_i}$, where μ_i is the average value of all x_i values and s_i is the standard deviation.

4.3 Classification with logistic regression

In a classification model, the machine learning algorithm tries to sort data into different classes. The simplest version is binary classification. For example, "is the email spam or not". In lineair regression, the hypothesis can output values other than the classes that exist. However there is a method, logistic regression, which constrains the hypothesis to the available classes.

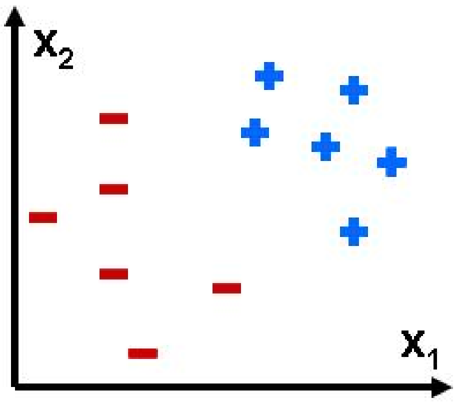

FIGURE 4.3: Binary classification. [13]

Figure 4.3 shows a simple binary classification example. x_1 and x_2 are features. Some points belong to the minus class, others belong to the plus class. A logistic regression or classification algorithm will learn from training data what data from the minus class and data from the plus class looks like. Afterwards, the classifier can be used to predict whether an input sample belongs to the minus class or the plus class.

4.3.1 Logistic regression

In logistic regression, a hypothesis is needed which outputs whether an input datapoint belongs to a class or not. It can also be explained as, "what is the chance that the input belongs to a class"? Now the hypothesis needs to output a percentage, the chance that the input belongs to a certain class.

Then a decision boundary is used. This decides whether the input belongs to the class or not. Usually the decision boundary is 0.5. If the hypothesis outputs a value higher than the decision boundary, it can be said that the input belongs to the class for which it is being tested. Logistic regresion uses the Sigmoid or logistic function for the hypothesis:

$$g(z) = \frac{1}{1 + e^{-z}} \tag{4.8}$$

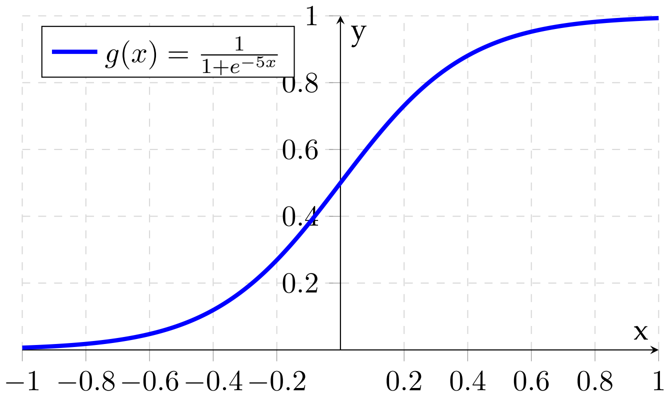

FIGURE 4.4: Sigmoid function

Using this function, the hypothesis can be written as:

$$H_0(x) = g(\theta^T X) = \frac{1}{1 + e^{\theta^T X}} \tag{4.9}$$

4.3.2 Cost function

The cost function from lineair regression cannot simply be applied to logistic regression. For classification it is important that data that belongs to the class has an output closer to 1, and that data that does not belong to the class has an output closer to 0. To do this a different cost function should be used. The notation $x^{(i)}$ and $y^{(i)}$ is used to denote the ith training sample. With dataset size is denoted by m. A helper function $Cost$ is used. This helps to see the parallel with the cost function for linear regression, as seen in Section 4.2.2.

$$J(\theta) = \frac{\sum_{i=1}^{m}(Cost(H_\theta(x^{(i)}), y^{(i)}))}{m} \tag{4.10}$$

$$Cost(H_\theta(x^{(i)}), y^{(i)}) = \begin{cases} -log(H_\theta(x^{(i)})), & \text{if } y^{(i)} = 1 \\ -log(1 - H_\theta(x^{(i)})), & \text{if } y^{(i)} = 0 \end{cases} \tag{4.11}$$

The given cost function accomplishes exactly what is wanted. If the expected output is 1 (belongs to a class), then the cost becomes greater as the output goes to 0 and analogous if the expected output is 0. Gradient descent for logistic regression is exactly the same as for lineair regression except with a different hypothesis.

4.3.3 Multi-class classification

The above hypothesis and cost function are for binary classification. To compute multi-class classification, the One-vs-all algorithm could be used. This algortihm splits the multiples classes into two groups. One group contains one class, the other group contains all other classes. Using these two new groups, the algorithms for binary classification can be used. In other words, for each class i a different logistic regression classifier $H_\theta(x)$ is trained to predict the probability that $y = 1$. On an input x, the most probable class is chosen.

Another method called One-vs-One could be used. This method compares all pairs of classes. It checks, "does the input belong rather to class A as compared to class B". It does this for all pairs and decides which class most likely contains the input.

The One-vs-One method uses $n_classes * (n_classes - 1)/2$ different logistic regression classifier $H_\theta(x)$. Each of these classifiers is trained to fit a pair of classes. [14]

To provide an example, Figure 4.5 can be used. On the left, binary classification is shown. The hypothesis and cost function as seen in the above sections can be used to do the classification. On the right, multi-class classification is used. There are three classes: triangles, squares and crosses.

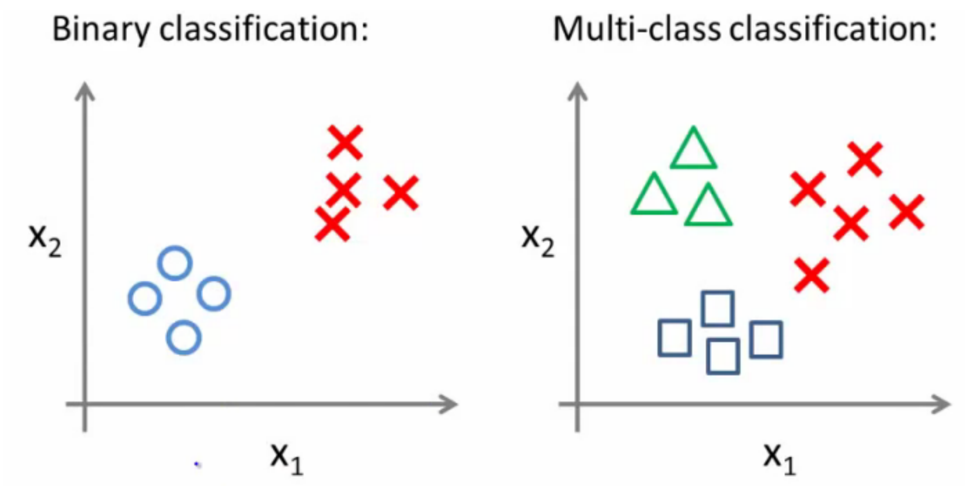

FIGURE 4.5: On the left, binary classification is shown. On the right, three different classes are used. [15]

One-vs-all classification will have to use three classifiers to predict which class a given input sample belongs to. The first classifier predicts whether the input samples belongs to the triangles or to another class. The second classifier predicts whether the input samples belongs to the squares or to another class and analogous for the third classifier. This method can quite quickly find the result. If the first classifier is used first and it is found that the input sample belongs to the triangles class, the prediction is done. [15]

One-vs-One classification will have to use six classifiers. Each classifier is trained to predict whether an input sample belongs to the triangles or the squares, to the triangles or the crosses, and so on. If the classifiers predict that the input samples belongs rather to the triangle class than to the square class and rather to the triangle class than to the crosses class, it can be predicted that the input sample belongs to the triangle class. Because the One-vs-One classification has to use more classifiers than One-vs-All, it is also slower. However, because it compares all pairs it is also more accurate. [15]

4.4 Overfitting

When the data is not modeled correctly and the model is too precise for the training data, a problem called overfitting occurs. Models that are overfitted have high variance, there are too many possible hypotheses. In other words, if there are too many features, the hypothesis may fit the training set very well but fails too correctly predict new examples. The hypothesis becomes very complex. In a similar way, underfitting may occur.

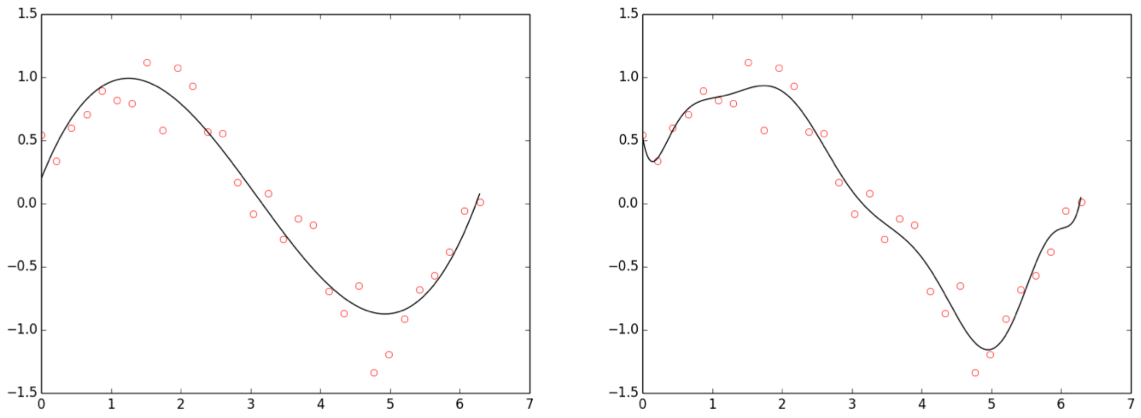

FIGURE 4.6: The points are generated by a sin function with Gaussian noise. Left: good fit (polynomial of degree 3), Right: overfit (polynomial of degree 10). [16]

There are methods to avoid overfitting. It is possible to reduce the amount of features. This can be done manually or by using a model selection algorithm which is explained in Section 6.1.2. Another option is regularisation. This method keeps all features but manages the values of the parameters of θ. Regularisation works well when there are a lot of features that are all contribute to be able to predict y.

Regularisation is a method to resolve overfitting. When a hypothesis overfits, the hypothesis is too complex. This means that the hypothesis fits the training data very well and does not generalise well as seen in Figure 4.6. It fits the training data too good and as such the hypothesis cannot properly make predictions on other data.

A hypothesis is complex when the values of θ are too large. Regularisation tries to solve this problem. When regularisation is used, the chosen values of θ become lower. For lineair regression, this can be done in the cost function by redefining the cost function as:

$$J(\theta) = \frac{\sum_{i=1}^{m}(H_0(x^{(i)}) - y^{(i)})^2 + \lambda * \sum_{j=1}^{m}(\theta_j^2)}{2m} \tag{4.12}$$

The only change to the cost function is that a penalty is added for large values of θ. When values of θ become larger, the cost function also becomes larger. When this cost function is minimized lower values for θ will be chosen. Only θ_1 till θ_m should be regularised. λ is called the regularisation parameter. Because of this small modification, the hypothesis that is constructed will be simpler.

Chapter 5

Algorithms

This chapter gives an introduction to different machine learning algorithms. It starts with two popular supervised learning algorithms, Support Vector Machines and K-Nearest Neighbors. Afterwards it explains how neural networks work. Finally some unsupervised techniques such as clustering and anomaly detection are discussed.

5.1 Support Vector Machines

Support vector machines or SVMs is a supervised learning algorithm which offers an alternative view on logistic regression. Support vector machines try to find a model which devides the 2 classes exactly with the same amount of margin on either side as shown in Figure 5.1. Samples on the margin are called the support vectors.

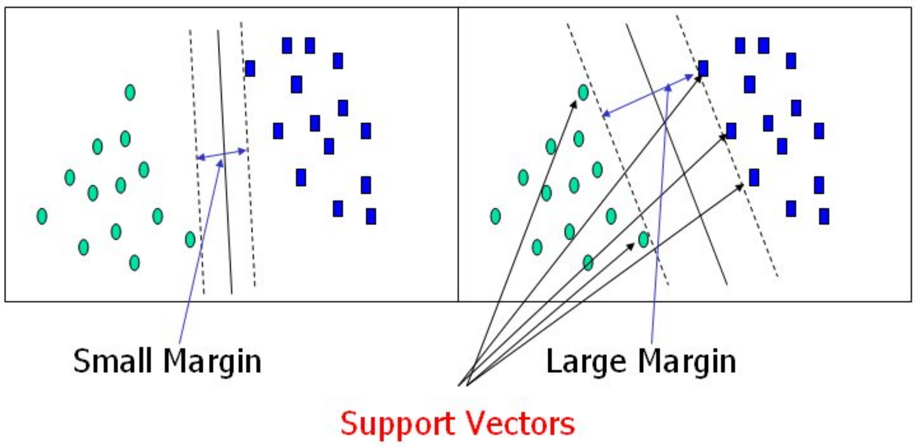

FIGURE 5.1: Support vector machines with their support vectors. [17]

In order to adapt Support Vector Machines to be able to fit non-lineair classifiers, some adjustments need to be done. This can be done with kernels. A kernel is a similarity function. The function compares two inputs and computes their similarity. Normally features are extracted from data and then fed into a machine learning algorithm. Kernels offer an alternative. The kernel should be a function to compare input data. The kernel, along with labeled data is then used to construct features. Using no kernel is called a lineair kernel. The basic type of kernel algorithms are called Gaussian kernels. The formula for Gaussian kernels is:

$$K(x,y) = \exp\left(\frac{-||x-y||^2}{2\sigma^2}\right) \tag{5.1}$$

5.2 K-Nearest Neighbors

The K-Nearest Neighbors or KNN algorithm is an algorithm which computes the classification by looking at the classes of the K-Nearest neighbors of the inputted data. The K-Nearest Neighbors is an Instance-based algorithm. [18] The chosen class is the class most common among its K-Nearest neighbors, this can be seen in Figure 5.2. K is typically a small number. When K is equal to 1, the assigned class is the same as the class of the closest sample.

When training the algorithm, the input data and classes are stored. There are mutliple methods to compute the distance between data. Euclidean distance can be used for continous data. For discrete variables another metric can be used, such as the overlap metric or Hamming distance.

A major drawback of the KNN algorithm is the weakness to skewed data. Since the class is chosen based on the most popular nearest class, these popular classes may dominate the prediction. This can be overcome by taking the distance between the input data and the neighbors into account.

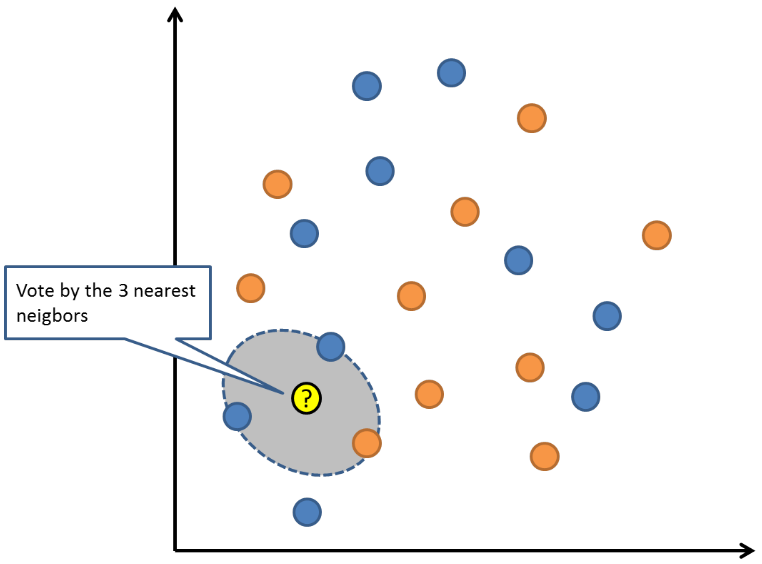

FIGURE 5.2: K-Nearest Neighbors. [19]

5.3 Clustering

Clustering is a machine learning concept using unsupervised learning. Unsupervised learning does not have labels with the training set. An unsupervised machine learning algorithm tries to find structure within the given training set. Clustering is the first type of unsupervised learning. It tries to cluster the training samples into different clusters.

5.3.1 K-means Algorithm

The K-means algorithm is a simple clustering algorithm. K is the number of clusters that is going to be used. The algorithm first randomly places the K clusters in the space (from the training set). This can be done by randomly chosing K training samples. Then it repeats the following steps until the cluster centers remain stationary. it iterates over all training samples, and assigns them to the closest cluster. Next the cluster centers are moved to the center of the total cluster.

The cluster centroids will be addressed using the symbol μ with μ_k the cluster centroid of cluster k. $c^{(i)}$ is the index of the cluster to which training sample $x^{(i)}$ has been assigned. $\mu_{c^{(i)}}$ is the cluster centroid to which training sample $x^{(i)}$ has been assigned. The cost function can be described as:

$$J(c,\mu) = \frac{1}{m}\sum_{i=1}^{m}(||x^{(i)} - \mu^{(i)}||)^2 \tag{5.2}$$

The final clusters that are found by K-means are dependent on the random placement of the K clusters in the beginning. It could lead to suboptimal clustering. This can be fixed by running K-means a number of times and after each iteration check the value of the cost function to find the most efficient run of K-means.

There is a similar problem with determining the amount of clusters. The same method could be used to determine the correct number of clusters. However, manually determining the number of clusters could be more time efficient.

5.4 Neural networks

Neural networks are a useful alternative to logistic regression if the amount of features becomes too large. The origin of neural networks are algorithms which try to mimic the brain. There is a hypothesis, the "one learning algorithm" hypothesis, that shows that the brain can learn very different things, such as sound, touch, etc. by using a single algorithm.

A neural network is created of neurons, which are called a logistic unit. Each neuron receives input wires, and has an output wire, which computes a value using the sigmoid (logistic) hypothesis, the activation function. A neural network is a group of "neurons" that are connected together. This can be grouped into a layered approach as seen in Figure 5.3. The first layer is called the input layer, the final layer is called the output layer which outputs a $H_\theta(x)$ which is class. All layers inbetween these layers are called hidden layers.

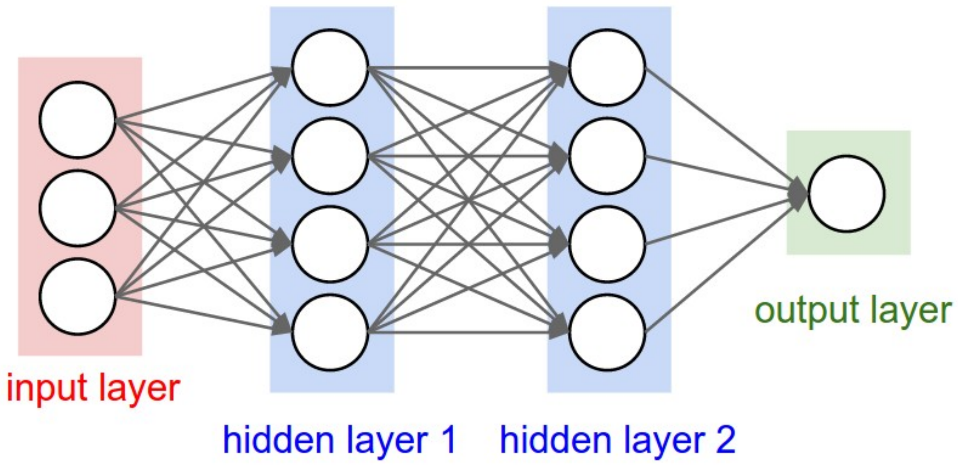

FIGURE 5.3: A neural network showing the different layers.[20]

Each logistic unit is denoted by a_i^j. j is the layer and i is the position in that layer. θ^j is a matrix of weights or parameters controlling function mapping from layer j to layer $j+1$. s_l is the number of units within a layer. The number of layers is denoted by L.

The neural network works similar to logistic regression, except that it performs logistic regression from layer to layer. The parameters θ_j required are learned by itself. The architecture of a neural network refers to the way the units are connected to eachother. Neural networks can be used for multi-class classification. Hereby there are mutliple units in the output layer and each unit represents a different class. K will denote the amount of output units.

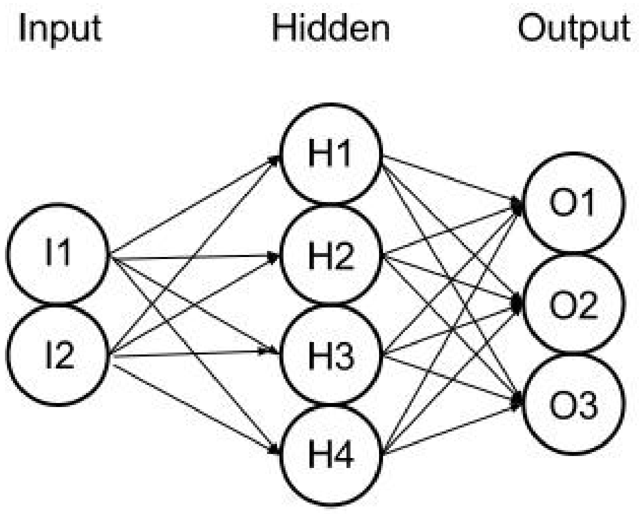

FIGURE 5.4: A neural network capable of multi-class classification. [21]

Data flows using the principle of forward propagation. The data passes through the first layers, move to the second layers and so on, until it arrives at the final layer. Mathematically this can be described as:

$$\begin{aligned} a^{(1)} &= x \\ z^{(2)} &= \theta^{(1)} * a^{(1)} \\ a^{(1)} &= g(z^{(2)}) \end{aligned}$$

5.4.1 Cost function and backpropagation

The cost function for neural networks is a generalisation of the cost function for logistic regression. The cost function accounts for the different layers, units and the number of output units. To minimize the cost function, the same methods such as gradient descent can be used. However, the problem is how to compute the partial derivative of the cost function. Using backpropagation, it is possible to compute this.

The resulting class is known in the final layer. From here, the algorithm can find the error. δ_j^l will be the symbol used for the error of node j in layer l. For an example with 4 layers:

$$\delta_j^{(4)} = a_j^{(4)} - y_j \tag{5.3}$$

$$\delta^{(3)} = (\theta^{(3)})^T \delta^{(4)} * g'(z^{(3)}) \tag{5.4}$$

The algorithm starts by setting a parameter $\Delta_{ij}^{(l)}$ to 0 for all i, j, l. Then, it iterates from $i = 0$ to m, with m the number of training samples. Each iteration, $a^{(1)}$ is set to $x^{(i)}$. Forward propagation is computed for $a^{(l)}$ for all $l = 2, 3, ..., L$. Using $y^{(i)}$, $\delta^{(L)}$ can be computed. Then the different $\delta^{(L-1)}$ to $\delta^{(2)}$ are computed. Finally, $\Delta_{ij}^{(l)}$ is incremented by $a_j^{(l)}\delta^{(l+2)}$ for each l. After the iteration is done, the final value for the partial derivative can be calculated: $\frac{\partial}{\partial \theta_i j^{(l)}} J(\theta)$. This can be done by dividing $\Delta_{ij}^{(l)}$ by the amount of training samples and adding $\lambda \theta_i j^{(i)}$.

5.4.2 Using a neural network

A neural network should have as many input units as the dimension of features. The number of output units is equal to the number of classes. Default, there should be either 1 hidden layer or if there are more, all layers should have the same number of hidden units. The neural network should be trained by first assigning random weights to the values of θ. Afterwards, forward propagation should be used to get $H_\theta(x^{(i)}$. Next the cost function should be computed. Backwards propagation is used to compute the partial derivatives. The result of backwards propagation can be checked by numerical methods to compute the gradient. Finally gradient descent or other advanced optimization methods with backpropagation should be used to try to minimize the cost function as a function to the parameters θ.

5.5 Dimensionality reduction

Dimensionality reduction is the process of reducing the amount of features used in machine learning algorithms. This can be used to increase the accuracy and the performace of machine learning algorithms. One form is to do data compression. For example, transform 3D data into 2D data and eliminating a feature or dimension. It can also be used to reduce dimensions to be able to efficiently visualise data.

5.5.1 Principle Component Analysis

Principle Component Analysis is a way to do dimensionality reduction. The algorithm is formed as a minimalisation problem. When given N-dimensional data and N-1 dimensional data is prefered. The algorithm tries to find the correct N-1 dimensional value so that the projection is the closest to the original data.

Before this algorithm should be run, the features of the data should be scaled, so all features are on a similar scale. This can be done by using mean normalization. In order to reduce the dimension from n to k the covariance matrix should be computed:

$$\Sigma = \frac{1}{m}\sum_{i=1}^{n}((x^{(i)})(x^{(i)})^T) \tag{5.5}$$

From this matrix, the eigenvectors need to be computed using singular value decomposition. From these values, only the first k values are going to be used and be multiplied with the training data.

PCA can be used to speed up the time it takes for other learning algorithms to learn. By using PCA, the amount of features or the amount of training samples is reduced which reduces the running time of the training, but the compressed data still retains the same information as the uncompressed data.

5.6 Anomaly detection

Anomaly detection is also a form of unsupervised learning. The algorithm learns what normal behaviour looks like through the training set and then tries to predict if a given input data belongs to the normal behaviour or is abnormal for any reason.

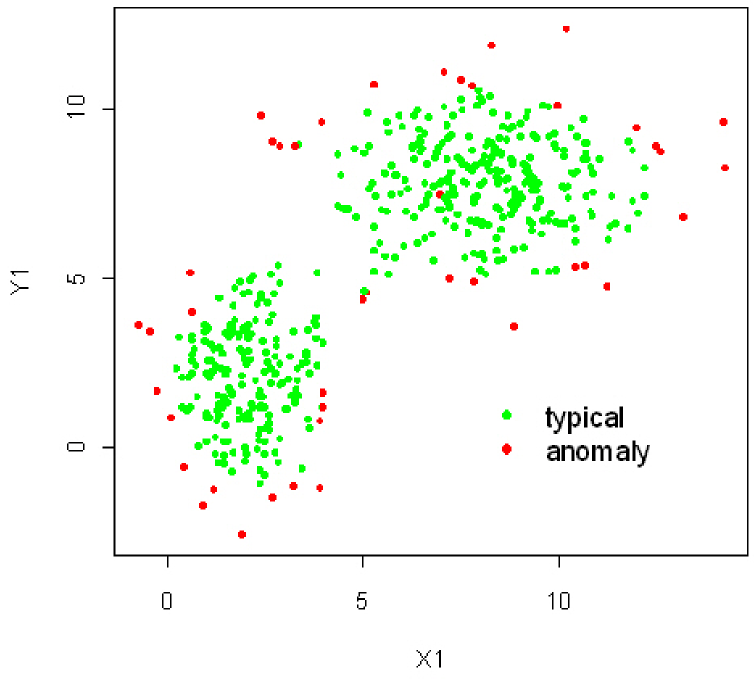

FIGURE 5.5: Anomaly detection. [22]

Anomaly detection algorithms make heavy use of (Gaussian) Normal distribution:

$$x \sim N(\mu, \sigma^2) \tag{5.6}$$

Hereby is μ the mean parameter and σ is the standard deviation. Now the probability of x being an anomaly can be calculated as:

$$p(x) = \prod_{j=1}^{n} (\frac{1}{\sqrt{2\pi}\sigma_j} exp(-\frac{(x_j - \mu_j)^2}{2\sigma_j^2})) \tag{5.7}$$

5.7 Other algorithms

There are also other, more specific and advanced categories of machine learning algorithms. There are decision tree algorithms, for example, Classification and Regression Tree (CART), Conditional Decision Trees, etc. There are Bayesian Algorithms which explicitly apply Bayes' Theorem for problems such as classification and regression. These algorithms include Naive Bayes, Gaussian Naive Bayes, Multinomial Naive Bayes, etc. [18]

Association Rule Learning Algorithms are methods that extract rules that best explain observed relationships between variables in data. Apriori algorithm and Eclat algorithm are examples of such algorithms. Deep Learning Algorithms are a modern modification to Artificial Neural Networks that exploit abundant cheap computation. Deep learning networks are very deep and complex neural networks. Deep Boltzmann Machine (DBM), Deep Belief Networks (DBN) and Convolutional Neural Network (CNN) are examples of deap learning algorithms. [18]

Ensemble Algorithms such as Bootstrapped Aggregation (Bagging) are models that combine multiple weaker models and try to combine the predictions made by these models. Yet there are still many more algorithms. [18]

A lot of algorithms are specifically constructed for a specific sub-field of machine learning, for example computer vision, natural language processing, etc. Even within the categories of algorithms that were discussed, regression, regularization, instance-based, clustering, neural networks and dimensionality reduction, there are a lot of different variants. However, these are considered to be to advanced and outside the scope of this thesis.

Chapter 6

Validation an algorithm

6.1 Machine learning evaluation

Machine learning diagnostics are tests that can be run to get to know what is and isn't working with a machine learning algorithm. They also provide guidance as to how performance could be improved.

6.1.1 Evaluating the hypothesis

The most obvious way to test whether the hypothesis is correct is by dividing the training set into two sets. The first set which should have approximatly 70% of the samples of the original training set will be used to train the learning algorithm. The other set is used to check whether the output of the learning algorithm is correct. After the algorithm has done its learning with the training set, this check set is thrown into the algorithm. Aftwards the output of the algorithm is compared to the labels of the check set. If the output isn't correct, a test error can be computed. These test errors can be used to compute global error value, which could be calculated by, for example, a mean squared error. This value can be used ito evaluate the hypothesis. [23]

6.1.2 Model selection algorithm

To get back to the problem of overfitting, there is another method next to regularisation called model selection. Model selection uses the same principle as mentioned above except it divides the training set into three new sets. A new training set, a cross validation set and a test set.

The training set is used to actually train the algorithm. The cross validation set is used to compare different algorithms or different models. Different models can be tested and compared to eachother by comparing the cross validation error, the error between the prediction and the actual output from the cross validation set. The testing set is used purely to test the algorithm once the cross validation has been done. It is considered good practice to use seperate sets for cross validation and testing. [24]

6.1.3 Diagnosing bias vs variance

High variance means that there is an overfitting problem. High bias on the other hand, is the opposite problem. It means that the hypothesis does not fit the training set at all. There is a relation between the amount of training samples and the training error. With more training data, the training error goes down.

With a low model complexity, there is a high error for the training set and a high error on the test set. This is a signal there is a underfitting problem or a high bias problem. It does not help to increase the amount of training samples since the model just is not complex enough.

With a high complexity model, the training set error is very low, but the test set error is very high. This is a signal there is an overfitting problem or a variance problem. This can be seen in Figure 6.1. The complexity of the model can be adjusted by changing the amount of features that are being used. [10]

6.1.4 Learning curves

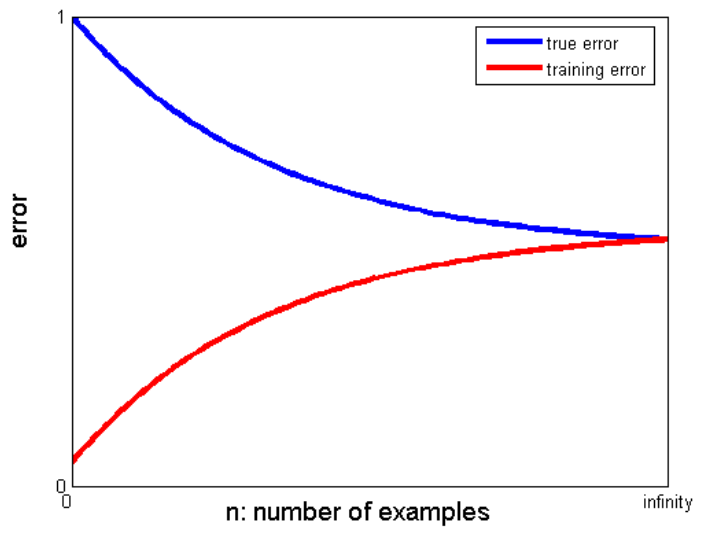

FIGURE 6.1: The effect of the amount of training samples on the accuracy.

In Figure 6.1 the general trend of training error and test (or true) error is shown. This figure shows the example for high bias and no matter how many training samples there are, the functions are already converged to the same error amount.

In contrary to high bias, high variance can be solved by using extra training samples. In that case there is a gap inbetween the true error line and the training error line and it is likely that they still converge to the same point. With a bigger training set, they converge more and more. These curves are called learning curves.

6.2 Error analysis

When trying to use machine learning for any purpose, such as intrusion detection systems, there are several considerations to be made. Another important step is the approach used to find the correct algorithm. The recommended approach is to start with a simple algorithm and test it with cross validation data. Afterwards, learning curves could be plotted to decide if more or less data, more or less features, ... are likely to help. Finally, error analysis can be done by maually examining the samples on which the algorithm made mistakes. This could help to spot any systematic trends in the type of samples on which the algorithm is making mistakes.

The error analysis mostly consists of manual or brute force work. The samples on which the algorithm is wrong need to be categorized based on which features could help it categorize correctly and on which class it belongs to. Calculating statistics, such as the accuracy of the algorithm can also help with error analysis. It is possible to brute force this approach by automatically trying a lot of different combinations of features. This is less efficient than manually tweaking the algorithm.

Another issue to account for are skewed classes. Skewed classes are classes that are underrepresented in training data. For example, in binary classification, when trying to classify a flow as malicious or not, the training set might only provide 0.5% malicious data. Knowing this, having an accuracy of 99% does not seem that great.

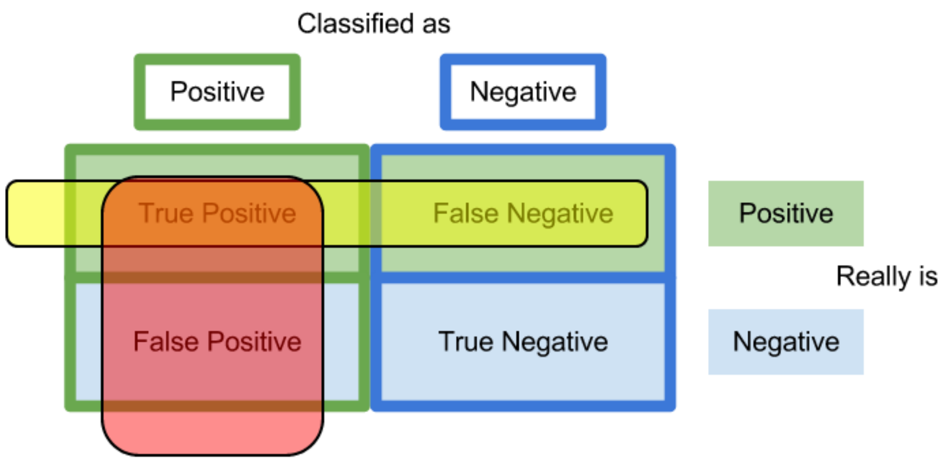

FIGURE 6.2: Precision and recall. [25]

A different metric is required to evaluate machine learning algorithms that are trained using skewed data. This metric is the precision and recall method. We classify a result as true positive, true negative, false positive and false negative. False positive and false negative respectively mean that the predicted value is falsely classified as positive and negative. These are the errors. True positive and true negative are correctly predicted values. **Precision** is defined as the fraction of predicted malicious flows that were actually malicious:

$$\frac{truepositive}{truepositive + falsepositive}$$

Recall is defined as the fraction of predicted positives and the actual amount of positives:

$$\frac{truepositive}{truepositive + falsenegative}$$

There is always a tradeoff to be made between precision and recall. As an effect of a higher precision, there will be a lower recall. Similarly, a higher recall means a lower precision.

Algorithms with a different precision and recall can be compared to eachother. This can be done using an F-score:

$$2\frac{PR}{P + R}$$

The F-score is the harmonic mean of precision and recall. For Table 6.1, this means that Algorithm 1 is the most effective. In contrast, using for example the average of both precision and recall would make Algorithm 3 the most effective.

The main reason to use the harmonic mean is because the average is taken of ratios (percentages), and in that case the harmonic mean is more appropriate than the (arithmetic) mean. [26]

TABLE 6.1: Example of precision and recall of certain algorithms.

	Precision (P)	**Recall (R)**
Algorithm 1	0.5	0.4
Algorithm 2	0.7	0.1
Algorithm 3	0.02	1.0

Looking at the values of the precision and the recall in Table 6.1, it can already intuitively be seen that Algorithm 1 is the best algorithm. For algorithm 3, the precision is 0.02, which means that there were a lot more false positives than true positives. With a recall of 1.0, there are almost no false negatives. This means that almost every prediction was a positive. That is not good at all.

Algorithm 2 has similar problem but the other way around. Only Algorithm 1 seems to have some kind of balance between precision and recall which makes it seem as the best algorithm. This confirms that the F-score is a better metric than the average.

TABLE 6.2: Example of average and F-score of certain algorithms.

	Average	**F-score**
Algorithm 1	0.45	0.444
Algorithm 2	0.4	0.175
Algorithm 3	0.51	0.039

The data that is being fed into a machine learning algorithm is also important to consider. Sometimes a lot of data can be useful. First, the assumption is made that the features are chosen correctly and sufficiently. Lots of data is useful when a machine learning algorithm is used with a lot of parameters such as a neural network with a lot of hidden units. This means that the algorithm has low bias.

6.2.1 Different Algorithms

If the number of features is large relative to the number of training samples, then using a lineair kernel Support Vector Machine or logistic regression is preferred. A Gaussian kernel is preferred when there are more training samples than features, but the difference is rather small. Otherwise, new features should be added. Neural networks are almost always effective but they may be slower to train.

Chapter 7

IP Flows

Flows are aggregated from all packet data that travels through the network. Flow exporters are programs which collect network packets and aggregate them into flow records. A flow is not the same as a TCP connection. A flow can be any communication between two devices with any protocol. Flows are defined using a (source_IP, destination_IP, protocol) tuple. This is why flows are also called IP Flows.

Since flow data does not contain any payload information, intrusion detection systems that use flow data cannot detect malicious behaviour embedded within payload data. [3]

7.1 How to use flow-data

The following attributes are available with flow-data:

- Source IP
- Destination IP
- Protocol name
- Source port
- Destination port
- Starting time of the flow
- Duration of the flow
- Amount of packets in the flow
- Amount of bytes in the flow

However, should a flow exporter be implemented, some additional features can be generated from packet data. 7

- Amount of TCP SYN within the flow
- Source and Destination Type of Service
- Payload size

These data can be used within the machine learning algorithms. However some variables have undesirable effects on the accuracy of the algorithm. Some care should be taken when training the machine learning algorithms with the additional data. Not all data, both training data as predictive data, will have the additional features.

Most machine learning libraries use numeral data instead of string data. All string data has been hashed in order to be able to use it in machine learning algorithms. The probability on a collision is low enough to be able to ignored.

7.1.1 IP addresses

Flow data can contain multiple forms of IP addresses. Both IPv4 and IPv6 data can be found. For some protocols the flow data can also contain the MAC-addresses instead of the IP-address. These addresses are hashed, so they become numeral, discrete data and are then fed into the machine learning algorithm.

Using the IP, it is possible to find the country or region of origin. Tests where the country of origin was fed into the machine learning algorithms, have been done. Results however showed, that the accuracy of the IDS became lower.

TABLE 7.1: The effects of using IP country-of-origin on accuracy of IDS.

With Country-of-origin	Accuracy
Yes	96.16%
No	98.57%

7.1.2 Ports and protocol name

Both the source and destination port are discrete data. They are usually received in decimal form, however some data-sets might use them in hexadecimal data or refer to ports as "ssh port" instead of "22". Port data, in decimal form, can be directly fed into the machine learning algorithm.

The protocol name can simply be converted to a standard string in lower case, in order to avoid errors by lower and uppercase forms of the same name (for example "tcp" and "TCP"). This string can than be hashed into a discrete value.

7.1.3 Timing

7.1.4 Size

The amount of packets used in the flow and the amount of bytes are both discrete data. They are always received in decimal form. They can immediately be fed into the machine learning algorithm.

Chapter 8

Machine learning for an IDS

8.1 Using ML for an IDS

An intrusion detection system has to detect whether some data it receives is either malicious or regular web traffic. This can be seen as a classification problem which means an machine learning algorithm for classification could be used. It needs to be determined whether data is either normal network traffic or malicious behaviour.

Some parameters have to be chosen that will be feed into the machine learning algorithm.

8.2 Disadvantages of using ML for an IDS

8.2.1 Problems

As said before, machine learning for an intrusion detection system is a classification problem. More precisely, it can be said that intrusion detection systems have to detect abnormal behaviour in a network with mostly normal behaviour. There are several problems that can be encountered when using machine learning techniques.

The first problem is the ability to detect new attacks. A machine learning algorithm compares incoming data with a model that it has created internally. An new type of malicious behaviour might appear to be closer to normal network traffic as compared to the model of known attacks.

Another problem is the diversity of network traffic. The notion of "normal network traffic" is difficult to actually define. The bandwidth, duration of connections, origin of IP addresses, applications used can vary enormously through time. This makes it quite difficult for machine learning algorithms to distinguish between "normal network traffic" and malicious behaviour.[27]

8.2.2 Solutions

There are several solutions that can be used in order to make machine learning algorithms more effective for intrusion detection systems. One option is to chance the way the classification problem is defined. Instead of defining the classes, "normal" and "malicious", there might be different classes for different types of malicious behaviour. In the same way, different classes can be defined for different types normal traffic.

8.3 Advantages of using ML for an IDS

Chapter 9

Implementation

- Discuss important decisions
- Talk about the data sets
- Cegeka
- CTU datasets
- Own generation + inline placement

9.1 Structure

9.2 Class diagram

9.3 Logging

9.4 Graphing

Chapter 10

Evaluation

Chapter 11

Future work

11.1 Prevention

This chapter will only be done if this is made in the thesis

11.1.1 Real-time detection

11.1.2 Data limiting

11.1.3 Connection closing

Chapter 12

Conclusion

Bibliography

[1] H. Alaidaros, M. Mahmuddin, and A. Al Mazari, "An overview of flow-based and packet-based intrusion detection performance in high speed networks", in *Proceedings of the International Arab Conference on Information Technology*, 2011.

[2] M. J. N. Jayveer Singh, "A survey on machine learning techniques for intrusion detection systems", *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 2, no. 11, 2013.

[3] A. Sperotto, G. Schaffrath, R. Sadre, C. Morariu, A. Pras, and B. Stiller, "An overview of ip flow-based intrusion detection.", *IEEE Communications Surveys and Tutorials*, vol. 12, no. 3, pp. 343–356, 2010. [Online]. Available: http://dblp.uni-trier.de/db/journals/comsur/comsur12.html#SperottoSSMPS10.

[4] A.-S. Kim, H.-J. Kong, S.-C. Hong, S.-H. Chung, and J. W. Hong, "A flow-based method for abnormal network traffic detection", in *Network operations and management symposium, 2004. NOMS 2004. IEEE/IFIP*, IEEE, vol. 1, 2004, pp. 599–612.

[5] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver, "Inside the slammer worm", *IEEE Security & Privacy*, no. 4, pp. 33–39, 2003.

[6] Y. Abuadlla, G. Kvascev, S. Gajin, and Z. Jovanovic, "Flow-based anomaly intrusion detection system using two neural network stages", *Computer Science and Information Systems*, vol. 11, no. 2, pp. 601–622, 2014.

[7] Z. Zhu, G. Lu, Y. Chen, Z. J. Fu, P. Roberts, and K. Han, "Botnet research survey", in *Computer Software and Applications, 2008. COMPSAC'08. 32nd Annual IEEE International*, IEEE, 2008, pp. 967–972.

[8] *Coursera machine learning stanford university*, https://www.coursera.org/learn/machine-learning, Accessed: 2016-02-09.

[9] *scikit-learn linear regression*, http://scikit-learn.org/stable/modules/linear_model.html, Accessed: 2016-02-15.

[10] *Stanford cs229 machine learning*, http://cs229.stanford.edu/materials.html, Accessed: 2016-04-18.

[11] *Intuitive machine learning : Gradient descent simplified*, http://ucanalytics.com/blogs/intuitive-machine-learning-gradient-descent-simplified/, Accessed: 2016-04-20.

[12] *Stackoverflow gradient descent convergence how to decide convergence?*, http://stackoverflow.com/questions/17289082/gradient-descent-convergence-how-to-decide-convergence, Accessed: 2016-04-18.

[13] *Cis 520 machine learning - 2015*, https://alliance.seas.upenn.edu/~cis520/dynamic/2014/wiki/index.php?n=Lectures.Classification, Accessed: 2016-04-22.

[14] C. M. Bishop, "Pattern recognition and machine learning. springer".

[15] *Logistic regression*, http://www.holehouse.org/mlclass/06_Logistic_Regression.html, Accessed: 2016-04-22.

[16] *Test-Driven Data Analysis in defence of xml: Exporting and analysing apple health data*, http://www.tdda.info/, Accessed: 2016-04-18.

[17] *DTreg svm - support vector machines*, https://www.dtreg.com/solution/view/20, Accessed: 2016-04-18.

[18] *A tour of machine learning algorithms*, http://machinelearningmastery.com/a-tour-of-machine-learning-algorithms/, Accessed: 2016-03-11.

[19] *Ricky Ho predictive analytics: Neuralnet, bayesian, svm, knn*, http://horicky.blogspot.be/2012/06/predictive-analytics-neuralnet-bayesian.html, Accessed: 2016-04-18.

[20] *Stackoverflow what is a 'layer' in a neural network*, http://stackoverflow.com/questions/35345191/what-is-a-layer-in-a-neural-network, Accessed: 2016-04-18.

[21] *Openstar cnx neural network implementation*, https://cnx.org/contents/6CAkQax3@1/Neural-Networks, Accessed: 2016-04-18.

[22] *R interface to oracle data mining*, http://www.analyticbridge.com/group/rprojectandotherfreesoftwaretools/forum/topics/r-interface-to-oracle-data, Accessed: 2016-04-18.

[23] A. Ng, *Coursera machine learning, evaluating a hypothesis*, https://www.coursera.org/learn/machine-learning/lecture/yfbJY/evaluating-a-hypothesis, Accessed: 2016-04-22.

[24] *Model selection*, https://en.wikipedia.org/wiki/Model_selection, Accessed: 2016-04-22.

[25] *Cornell cs4670*, http://www.cs.cornell.edu/courses/cs4670/2015sp/lectures/lec34_datasets_web.pdf, Accessed: 2016-04-18.

[26] *Which "mean" to use and when?*, http://stats.stackexchange.com/questions/23117/which-mean-to-use-and-when, Accessed: 2016-04-22.

[27] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection", in *Proceedings of the 2010 IEEE Symposium on Security and Privacy*, ser. SP '10, Washington, DC, USA: IEEE Computer Society, 2010, pp. 305–316, ISBN: 978-0-7695-4035-1. DOI: 10.1109/SP.2010.25. [Online]. Available: http://dx.doi.org/10.1109/SP.2010.25.

Appendix A

Meetings

A.1 Meeting 1: 09 Feb 2016

aanwezigen: Peter Quax, Bram Bonne, Pieter Robyns, Axel Faes

Dit is de eerste bijeenkomst met de begeleiders en promotor. Er is dus geen rapportering mogelijk van een vorige bijeenkomst. Tijdens de bijeenkomst is beslist om een *intruder detection system* te bestuderen en te implementeren.

De actiepunten die gedaan moeten worden:

- Beslissen voor wie het systeem gemaakt moet worden. Gaat dit voor end users zijn, of voor grote data centers. Hieraan hangt vast welke data (packets of netflow) gebruikt moet worden.
- Bekijken hoe machine learning algoritmes gebruikt kunnen worden in een *intruder detection system*.
- Bekijken wat netflow is.
- Er moet gekeken worden naar de manier waarop anomalies gegenereerd gaan worden om het systeem te testen/trainen.

Volgende afspraken zijn gemaakt:

- Er is gevraagd om te zorgen dat het systeem ook op correcte wijze informatie kan weergeven aan gebruikers. Tijdens het semester moet bekeken worden hoe deze weergave moet gebeuren.
- Libraries gebruiken indien mogelijk, om te vermijden dat het wiel opnieuw uitgevonden word.
- Er is de mogelijkheid geboden om aan de thesis te werken op het EDM.
- Er is afgesproken om *Overleaf* te gebruiken om de thesis in te schrijven.
- Een ruwe planning voor het werk moet gemaakt worden tegen 12 Feb.
- Een wekelijkse meeting is vastgelegd. Dit om 10:00 elke vrijdag.
- Begin mei moet een eerst draft van de thesis klaar zijn en eind mei moet de finale draft af zijn.
- Er moet een vulgariserende tekst gemaakt worden en een postersessie gegeven worden (op 29 juni).

A.2 Meeting 2: 12 Feb 2016

aanwezigen: Bram Bonne, Pieter Robyns, Axel Faes

Dit is de tweede bijeenkomst met mijn begeleider. Netflow bevat op zichzelf niet zoveel informatie, maar het is toch handig om te kijken welke bevindingen gemaakt kunnen worden met deze data. Mogelijks kan er, indien gevonden wordt dat netflow alleen niet genoeg informatie bevat, ook gebruikt gemaakt worden van packet data.

Er is de mogelijkheid besproken om eventueel meerdere machine learning algoritmes te implementeren en te bekijken in welke situaties welke algoritmes beter werken.

De actiepunten die gedaan zijn:

- *Beslissen voor wie het systeem gemaakt moet worden.*: Dit gaat gedaan worden voor data centers
- Er zijn verschillende classificaties van machine learning algorithmes gevonden die gebruikt kunnen worden.
- Verschillende grote data sets van netflow en packets met sporen van anomalies zijn gevonden. Alsook programma's om verkeer te genereren.

Volgende actiepunten zijn besproken:

- Verder uitwerken van welke machine learning algoritmes gebruikt kunnen worden
- Bekijken netflow v9

A.3 Meeting 3: 19 Feb 2016

aanwezigen: Bram Bonne, Pieter Robyns, Axel Faes

Professor Quax is aan het bekijken ofdat ik (gelabelde) netflow data kan verkrijgen van Cegeka. Dit zou heel handig zijn om mijn implementatie te testen op real world data.

Voorlopig moet ik enkel focussen op een passive intrusion detection systeem, geen preventie en niet direct inline in het netwerkverkeer. Ook de visualisatie moet later bekeken worden, de gebruiker is een netwerkadministrator. Er is tevens besproken dat python zelf mogelijks te traag is om packet sniffing op een goede snelheid uit te voeren. Hiervoor zou ik wireshark kunnen gebruiken (of de command line versie). Er is besproken om eventueel zelf datasets te genereren door malware te runnen op een VM of aparte machine.

De datastructuur voor de machine learning algoritmes is bekeken. Ik moet eens bekijken hoe de timestamps van de flowdata gebruikt kunnen worden. Om de effectiviteit (van de machine learning algoritmes) mogelijks te verhogen ga ik eens bekijken of ip-adressen ingedeeld kunnen worden in country-of-origin of iets dergelijks. Dit zou de machine learning algoritmes de mogelijkheid bieden om ook op deze parameter te bekijken of data malicious is of niet.

De actiepunten die gedaan zijn:

- Er is al een basis implementatie uitgewerkt voor het IDS
- De netflow structuur is bekeken en er is een datastructuur opgestelt die gefeed kan worden aan verschillende machine learning algoritmes.
- Progressie in de machine learning cursus: chapter 3 van de 18.

Volgende actiepunten zijn besproken:

- Beginnen aan de thesis: het schrijven van een hoofdstuk over machine learning en over hoe deze algoritmes toegepast kunnen worden op een intrusion detection systeem.
- Verder werken in de machine learning cursus.
- Ik moet eens bekijken ofdat ik een programma vind om pcap files om te zetten naar netflow. Anders moet ik dit zelf schrijven.

Ik heb ook een korte planning gemaakt van hoe de thesis eruit zou zien:

- Inleiding:
  - wat is een IDS
  - Waarom is er gekozen voor dit type IDS (host vs netwerk)
  - Waarom voor data centers
  - Waarom netflow
  - Waarom machine learning
- Wat is machine learning
- Hoe passen we machine learning toe op IDE en wat zijn de voor/-nadelen
- Welke machine learning algortimes zijn wel/niet gebruikt
- Wat zijn de voor/nadelen van netflow
- Hoe met combinatie netflow/packets (Als dit gedaan zou worden)
- Welke data sets zijn gebruikt
- Wat zijn de bevindingen
- Hoe kan visualisatie/feedback gebeuren (richting admin en richting automatische preventie)
- Conclusie

A.4 Meeting 4: 26 Feb 2016

aanwezigen: Bram Bonne, Axel Faes

Deze week is voornamelijk besteed aan de implementatie. Er is een netflow exporter geschreven. Er is bekeken ofdat timestamps gebruikt kunnen worden en ofdat ip-adressen opgedeeld kunnen worden per land. Er is besloten dat dit zeer weinig effect heeft op de accuraatheid van de machine learning algoritmes.

Momenteel zijn Support vector machines en K-nearest Neighbor Classifier algoritmes bekeken. Het K-nearest Neighbor Classifier algoritme is zeer efficient (98%).

In een later stadium kan bekeken worden om eventueel verdere analyse te doen op de data die malicious gevonden is, eventueel door pakketten te analyseren, of nogmaals door machine learning technieken. Er kan ook eens bekeken worden om een VM op te zetten, en daarin malware te runnen en dit verkeer te monitoren. Herbij zouden eigen datasets gegenereerd kunnen worden.

De machine learning cursus is gevolgd tot hoofstuk 7. De cursus zou normaal af moeten zijn binnen 2 weken.

De actiepunten die gedaan zijn:

- Er is al een netflow exporter geschreven
- Er zijn experimenten uitgevoerd m.b.t de datastructuur die meegegeven wordt aan de machine learning cursus.
- Progressie in de machine learning cursus: chapter 7 van de 18.
- Er is begonnen aan de thesis.
- Het zou interessant zijn om eens te kijken ofdat ip-addressen opgedeeld kunnen worden in subnets.

Volgende actiepunten zijn besproken:

- Focussen op de thesis
- Verder werken in de machine learning cursus.

A.5 Meeting 5: 04 Mar 2016

aanwezigen: Bram Bonne, Axel Faes

Deze week is voornamelijk besteed aan de thesis en aan het leren van de machine learning cursus. De machine learning cursus is gevolgd tot hoofstuk 10. De cursus zou tegen volgende meeting af moeten zijn. De algemene structuur van de thesistekst is nagekeken. Het hoofdstuk over "Attack classification" moet uitgebreid worden met een algemene uitleg over hoe aanvallen gedetecteerd kunnen worden. Het hoofdstuk over de gebruikte datasets moet samengevoegd worden met het hoofdstuk dat de implementatie beschrijft. Het hoofdstuk over voor/nadelen van machine learning voor intrusion detection systemen moet verwerkt worden in het algemene hoofdstuk over machine learning.

De actiepunten die gedaan zijn:

- Verder werken aan ML cursus
- Schrijven aan thesis.

Volgende actiepunten zijn besproken:

- Verder werken aan ML cursus
- Schrijven aan thesis.

A.6 Tussentijdse presentatie: 08 Mar 2016

aanwezigen: Maarten Wijnants, Peter Quax, Wim Lamotte, Jori Liesenborgs, Wouter vanmontfort, Pieter Robyns, Robin Marx, Bram Bonne, Axel Faes

Er is een tussentijdse presentatie geweest waarbij ik mijn huidige progressie moest tonen en een planning moest geven. De presentatie zelf is goed verlopen. Na de presentatie heb ik verschillende vragen gekregen.

Veel vragen die gesteld waren, waren bedoeld om te kijken of we het nut/doel van de bachelorthesis kennen en hoe we de invulling correct doen. Ook een belangrijk aspect is hoe het valideren van de correctheid van de experimenten die gedaan zijn/worden zal gebeuren.

Een opmerking was dat ik ook bestaande Intrusion detection systemen moet bekijken en ofdat deze machine learning gebruiken. Dan is ook belangrijk waarom ze het wel of niet gebruiken.

Er was verwacht dat ik al iets verder stond met de Machine learning cursus. Hierdoor kon ik niet altijd op de volledige diepgang de gestelde vragen beantwoorden. Ik begreep ook niet altijd de onderliggende vraag waardoor ik te oppervlakkig antwoorde. Qua machine learning algoritmes moest ik goed opletten voor overfitting en uitleggen hoe ik hiermee omga.

Er zijn ook vragen gesteld m.b.t mijn geplande extra om het intrusion detection systeem real-time te maken. Normaal wordt een flow pas doorgegeven als deze volledig afgesloten is, een mogelijke piste zou zijn om flows al te bekijken ook al zijn ze nog niet afgesloten. Ook het runnen van een VM met malware erop om zelf data-sets te generen is bevestigd dat een goed idee zou zijn. Als ik hiervoor infrastructuur nodig heb moet ik dit vragen.

Ik moet opletten met aanvallen die maar zeer weinig netwerktraffiek genereren. Ook moet ik goed beschrijven welke aanvallen wel of niet gedetecteert kunnen worden en uitleggen waarom. Dit staat momenteel al beschreven in mijn thesistekst. Ik zou ook mogelijkheden kunnen uitleggen die ervoor zouden kunnen zorgen dat ik toch alle (of een groot deel) van de aanvallen zou kunnen detecteren. Dit zou bv kunnen door toch packet-data te gaan bekijken.

Een algemene opmerking die gegeven was, was dat de presentatie visueler mocht zijn. Figuren en afbeeldingen zijn aangenamer om te tonen aan een publiek. Bij de postersessie moet er ook goed opgelet worden dat ik van persoon tot persoon bekijk hoe diep ik de materie uit mijn bachelorthesis kan uitleggen.

A.7 Meeting 6: 11 Mar 2016

aanwezigen: Bram Bonne, Axel Faes

Deze week is voornamelijk besteed aan de thesis en het afmaken van de machine learning cursus. De machine learning cursus is volgens planning afgemaakt. De thesistekst moet ingestuurd worden zodat deze al nagekeken kan worden. Komende weken zal gespendeerd worden aan de implementatie en het uitzoeken hoe de flowdata gebruikt kan worden in de machine learning algoritmes.
De actiepunten die gedaan zijn:

- Afmaken Machine learning cursus
- Schrijven aan thesis.

Volgende actiepunten zijn besproken:

- Verdere implementatie algoritmes
- Bekijken hoe de flowdata gefeed kan worden aan de machine learning algoritmes

A.8 Meeting 7: 18 Mar 2016

aanwezigen: Bram Bonne, Pieter Robyns, Axel Faes

Deze week is voornamelijk besteed aan implementatie. Er zijn verschillende algoritmes geïmplementeerd. Deze algoritmes zijn K-Nearest Neighbors, K-Means, Lineair Kernel Support Vector Machines met One vs All classification, Lineair Kernel Support Vector Machines met One vs One classification en Gaussian Kernel Support Vector Machines.

Er zijn verschillende experimenten uitgevoerd. De poort nummers zijn geïmplementeerd in 2 varianten. De eerste variant splitst de poorten in categoriën (veel gebruikte poort nummers en niet-veel gebruikte poort nummers) als een binaire feature. De andere variant maakt van elk poort nummer een nieuwe binaire feature (die stelt of de poort gebruikt is of niet).

Deze variant geeft echter heel veel features, dit is enorm inefficient.

De gestelde feedback was om te kijken hoe goed het werkt als poort nummers als een continue feature voorgesteld wordt en om de categoriën op te spliten in >1024 en <1024. De IP-data can opgesplitst worden in aparte features voor IPv6, IPv4 en MAC. Deze data kan mogelijks voorgesteld worden als continue data. Er is voorgesteld om ook andere algoritmes te implementeren. Om gebruik te kunnen maken van datasets van Cegeka moet ik een presentatie maken en een afspraak maken met professor Quax.

Er is feedback gegeven op de thesistekst. Ik moet als eerste goed letten op spelfouten. De inleiding moet algemener uitgelegd worden. Ook niet gebruikte concepter zoals Signature-based IDS moet dieper uitgelegd worden. De attack classification maakt al teveel assumpties over wat er gedetecteerd kan worden. Dit zou eerst algemener uitgelegd moeten worden. Het machine learning hoofstuk bevat in het algemeen te weinig high-level beschrijvingen.

De actiepunten die gedaan zijn:

- Begin van hoe flowdata gebruikt kan worden
- Implementatie van K-Nearest Neighbors
- Implementatie van K-Means
- Implementatie van Lineair Kernel Support Vector Machines met One vs All classification
- Implementatie van Lineair Kernel Support Vector Machines met One vs One classification
- Implementatie van Gaussian Kernel Support Vector Machines.

Volgende actiepunten zijn besproken:

- Herschrijven en verwerken van feedback op de thesistekst
- Verdere implementatie: andere algoritmes
- Verdere implementatie: Ports indelen in >1024 en <1024
- Verdere implementatie: Ports indelen als continue data
- Verdere implementatie: IP indelen als continue data
- Verdere implementatie: Starttime instellen als unix time
- Maken presentatie voor Cegeka data set

A.9 Meeting 8: 24 Mar 2016

aanwezigen: Bram Bonne, Pieter Robyns, Axel Faes

Deze week is voornamelijk besteed aan implementatie. Er zijn verschillende experimenten uitgevoerd. De poort nummers zijn geïmplementeerd zodanig dat er een indeling is tussen normale poorten (<1024) en speciale poorten (>1024). Er is ook gekeken ofdat poort data niet continu voorgesteld kan worden. Dit geeft echter slechtere resultaten dan te werken met de binaire indeling.

De IP-data kan opgesplitst worden in aparte features voor IPv6, IPv4 en MAC. Deze data is voorgesteld als continue data. Het is moeilijk om IP-addressen zelf te gaan onderverdelen in categorieen of subnets. Het indelen als continue data geeft dan ook de beste accuraatheid. De starttijd is ook geïmplementeerd om te voegen als feature. Dit gebeurt door te bekijken in welke dag van de week/ uur van de dag en minuut van het uur de data gegenereerd word. Echter had deze feature geen goede invloed op de accuraatheid

Er zijn enkele algoritmes verder geïmplementeerd zoals een Decision Tree learner en een Naive Bayes algoritmes. Beide zijn supervised learning technieken. De Naive Bayes had nog goede accuraatheid, de Decision Tree Learner gaf slechtere resultaten.

De actiepunten die gedaan zijn:

- Verdere implementatie: andere algoritmes
- Verdere implementatie: Ports indelen in >1024 en <1024
- Verdere implementatie: Ports indelen als continue data
- Verdere implementatie: IP indelen als continue data
- Verdere implementatie: Starttime instellen als unix time

Volgende actiepunten zijn besproken:

- Herschrijven en verwerken van feedback op de thesistekst
- Testen van de implementatie
- Bekijken van Unsupervised learning algoritmes

A.10 Meeting 9: 01 April 2016

aanwezigen: Bram Bonne, Peter Quax, Axel Faes

Deze week is voornamelijk besteed aan implementatie. Er is een nieuwe dataset gevonden. Deze dataset is afkomstig van de universiteit van Twente. Er was een honeypot opgesteld op het netwerk, alle data die hiermee gevangen is, is geclassificeerd en een dataset mee gemaakt. De dataset bevat voornamelijk externe aanvallen.

Er is ook gefocused op het trachten te gebruiken van unsupervised learning algoritmes. Echter heeft dit weinig opgebracht. Er kan wel op een accurate manier onderscheidt gemaakt worden tussen malicious en niet malicious data, maar het is moeilijk om vervolgens af te leiden over welk type malicious data het gaat.

De features die gebruikt worden in de machine learning algoritmes zijn nog eens overlopen. Professor Quax kwam met het idee om IP-adressen eventueel op te delen in origine (zoals land). Dit kan gebeuren via services zoals WhoIs.

De EDM dataet kan afgehaald worden bij het kantoor van professor Quax. Er moet ook zo snel mogelijk een meeting georganiseerd worden met Cegeka.
De actiepunten die gedaan zijn:

- Herschrijven en verwerken van feedback op de thesistekst
- Testen van de implementatie
- Bekijken van Unsupervised learning algoritmes

Volgende actiepunten zijn besproken:

- Verwerken data EDM
- Implementatie van WhoIs als feature
- Maken presentatie voor Cegeka data set

A.11 Meeting Cegeka: 06 April 2016

aanwezigen: Peter Quax, Axel Faes, Cegeka

Er is een meeting geweest met Cegeka om de mogelijkheid te bespreken om data te kunnen gebruiken die afkomstig is van Cegeka, alsook om informatie te krijgen van de hudige IDS' die gebruikt worden.

In het begin is een korte presentatie gegeven waarin de eigenschappen van het te ontwikkelen systeem uitgelegd worden. Er wordt ook beschreven wat de algemene onderzoeksvragen zijn en hoe data nu gebruikt kan worden in machine learning algoritmes.

Een eerste vraag die gesteld is, is welke classificatie van 'onverwachte traffiek' er momenteel gebeurt in datacenter/hosting context. Cegeka werkt heel gelaagd. Voor de routers staat een DDoS protection systeem. Na de router staan zowel firewalls als SIEM devices. Voor grotere klanten is er extra beveiliging voorzien in de vorm van afgestelde IPS systemen en meer gedetailleerde threat detection. Zoveel mogelijk data wordt verwerkt, zowel flows als meer gedetailleerd. De systemen werken voornamelijk met een signature database en verwerken de data voornamelijk automatisch. De systemen moeten wel manueel afgesteld en onderhouden worden.

De classificatie gebeurt heel gedetailleerd door deze verschillende lagen. Er werd wel gesteld dat het veel interessanter is om outbound verkeer na te kijken in vergelijking met binnengaand verkeer. Zaken zoals port-scans zijn interessant om te weten maar gebeuren heel veel en kunnen al goed gedetecteerd worden.

Er was veel interesse naar een intrusion detectie systeem dat op basis van netflow en machine learning technieken werkt. Er accuraatheid van rond de 70-80 procent zou gezien worden als een goede accuraatheid. Ze hebben ook liever false positives dan false negatives. Teveel alerts genereren is heel vervelend, maar het is belangrijker dat voldoende anomalieen gedetecteerd worden.

Er is gevraagd of het mogelijk is om data te verkrijgen. Dit was zeker mogelijk. De netflow en corresponderende logs van 3 dagen wordt geleverd. De logs worden zowel in binair als text formaat geleverd. Het binair formaat kan ingelezen door een programma dat de logs visualiseerd. Dit programma wordt ook meegeleverd. Mogelijks zou ook output van het DDoS systeem geleverd kunnen worden. Om te bekijken ofdat een flow gezien is als malicious of niet moet dit bekeken worden of deze flow voorkomt in de logs.

A.12 Meeting 10: 12 April 2016

aanwezigen: Bram Bonne, Axel Faes

Afgelopen week, woensdag 6 april, is de meeting geweest met Cegeka. Van deze meeting is een verslag gemaakt. Op maandag 4 april is de dataset van het EDM verkregen. Deze dataset bevat netflow data van het EDM netwerk van 18 februari tot 24 maart. Elke dag is netflow beschikbaar die voorgekomen is tussen 10u tot 24u. Deze zijn per 15 minuten gelogd in een file.

Er is kort gewerkt aan het verwerken van de data van het EDM. Dit gebeurt door de data te laten verwerken door verschillende algoritmes. De data waarvan vervolgens gesteld kan worden dat deze daadwerkelijk malicious is, zal doorgegeven worden aan professor Quax. Op het moment zijn er nog niet genoeg testen uitgevoerd om iets te kunnen zeggen over de data.

Er is geïmplementeerd dat de country-of-origin van een IP ook gebruikt kan worden als feature voor de machine learning algoritmes. Er is gevonden dat dit voornamelijk werkt voor data die van buitenaf komt, zoals port scans. Er is ook kort al geprobeerd om visualisaties te maken van de machine learning algoritmes. Zulke visualisaties zouden interessant kunnen zijn om ook in de thesistekst te gebruiken. Tegen volgende bijeenkomst moet voornamelijk gewerkt worden aan de thesistekst. De actiepunten die gedaan zijn:

- Meeting Cegeka
- Implementatie van WhoIs als feature
- Implementatie van kleine visualisatie van algoritmes
- Verkrijgen data van EDM

Volgende actiepunten zijn besproken:

- Herschrijven en verwerken van feedback op de thesistekst

- Visualisaties van machine learning algoritmes zijn interessant voor thesistekst.

A.13 Meeting 11: 18 April 2016

aanwezigen: Bram Bonne, Axel Faes

Ik heb mijn huidige vooruitgang laten zien van de bachelorproef tekst. Hierbij is een korte herschikking gekomen van de hoofdstukken. Ik moet het "Implementatie" hoofdstuk opsplitsen naar een hoofstuk "Implementatie" en een hoofdstuk "Evaluatie". In "Implementatie" moet mijn implementatie zelf beschreven staan, in "Evaluatie" moeten de resultaten beschreven worden.

Verder is kort uitgelegd wat de algemene structuur is van het machine learning hoofdstuk en welke aanpassingen er gebeurd zijn. Het hoofdstuk is opgesplitst in meerdere delen zodanig dat de uitleg, de algoritmes en validatie in aparte hoofdstukken uitgelegd wordt. Het hoofdstuk "Preventie" moet ik aanpassen naar iets gelijkaardigs aan "Future work". Het hoofdstuk "Visualisatie" moet samengevoegd worden met "Implementatie". De actiepunten die gedaan zijn:

- Verwerken feedback op machine learning hoofdstukken.

Volgende actiepunten zijn besproken:

- Herschrijven en verwerken van feedback op de thesistekst
- Halen data bij Cegeka