Differential IP clustering for Vulnerability Detection using applied Machine Learning

Sebastian Bocquier
Division of Computing and Mathematics
University of Abertay Dundee

A thesis submitted for the degree of
Honours of Science
in
Ethical Hacking and Computer Security

April 16, 2017

This thesis is dedicated to
no one
for no special reason

Acknowledgements

I would like to take this opportunity to thank my project supervisor, Xavier Bellekens, who introduced me to the topic of Machine Learning and offered guidance from the beginning through to the completion of this project. Without whom this project would not have been possible. I would also like to thank the module leader, Colin McLean, who has given me continuous advice throughout my years of study at Abertay University.

Abstract

plenty of waffle, plenty of waffle, plenty of waffle, plenty of waffle, plenty of waffle, plenty of waffle, plenty of waffle, plenty of waffle.

Contents

1 Introduction **1**

1.1 Background 1

1.2 Current State of Machine Learning 2

1.3 Significance of study 4

1.4 Types of Machine learning 5

1.4.1 Classification 5

1.4.2 Regression 5

1.5 Types of Learning algorithms 6

1.5.1 Supervised 6

1.5.2 Semisupervised 6

1.5.3 Unsupervised 7

1.5.4 Reinforcement 7

1.6 Common Machine Learning Algorithms 7

1.6.1 Linear Regression 7

1.6.2 K-Nearest Neighbours 8

1.6.3 K-means Clustering 8

1.6.4 Dimensionality Reduction 8

1.6.5 Decision Trees 9

1.6.6 Random Forests 9

1.6.7 Ariticial Neural Networks 10

2 Methodology **11**

2.1 Design 11

2.1.1 Application Brief 12

2.2 Infrastructure 13

2.3 Proof of Concept Testing 13

A Code **15**
A.1 code1 15

List of Figures

1.1 Timeline of artificial intelligence for games 2
1.2 Illustration of machine learning types 6

2.1 Application Infrastructure flow Diagram 14

Chapter 1

Introduction

This section will give a brief background and history of Machine Learning before going in to depth with the definitions and types of algorithms used in modern Machine Learning systems. An analyses of the impact Machine Learning has in the field of computer security and how it can be applied directly to penetration testing and red teaming. Research aims and objectives will be provided as well as a statement to the structure of the rest entire thesis.

1.1 Background

The term machine learning was first defined by Arthur Samuel in the year 1959 as a "Field of study that gives computers the ability to learn without being explicitly programmed" and later as "a field of study that concentrates on induction algorithms and on other algorithms that can be said to 'learn'", Kohavi & Provost (1998). Machine learning has largely evolved from several subfields of artificial intelligence, specifically, computational learning and pattern recognition but now it stands on its own with its subfield Deep Learning being at the forefront of technology. Machine learning consists of the studying and construction of algorithms that can learn from and make predictions of data, Simon (2013). The early information available for machine learning was almost entirely theoretical due to the lack of processing power at the time. One of the early pioneers was Valiant (1984) whom developed a PAC learning framework and established the theory of a learnable algorithm, Munoz (2012). Modern machine learning algorithms use many calculations from statistics, information theory, theory of algorithms, probability and functional analyses, Munoz (2012).

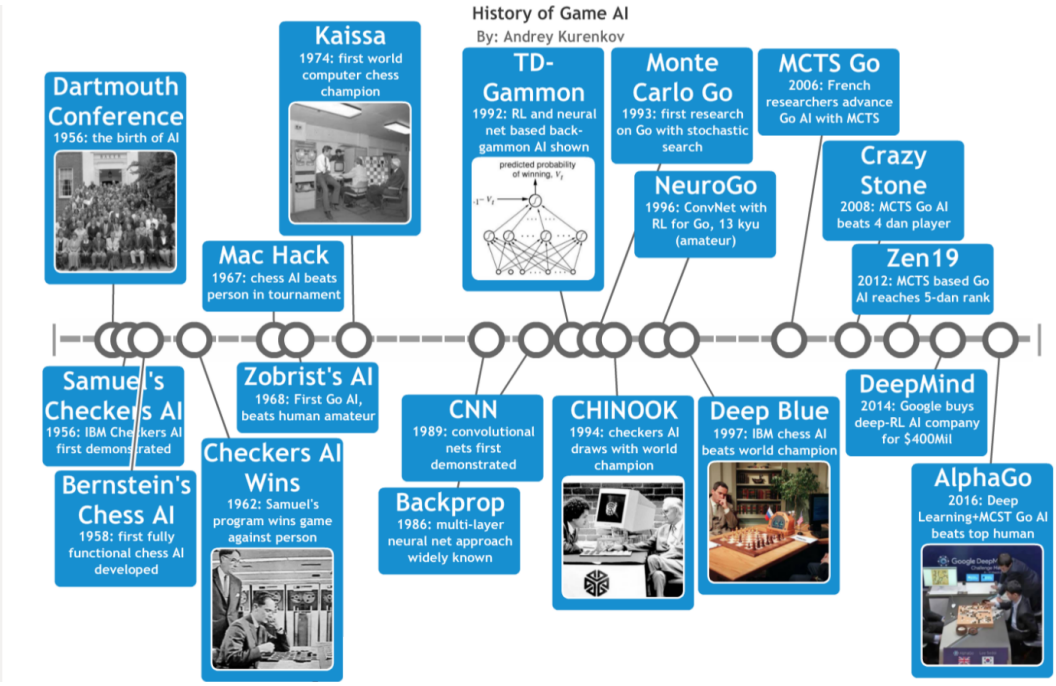

Figure 1.1: Timeline of artificial intelligence for games, Kurenkov (2016)

1.2 Current State of Machine Learning

In the recent years, machine learning accuracy and efficiency has increased dramatically which has caused the field to rapidly gain in popularity, especially in deep learning. One of the major reasons for this is due to the improvement in graphics processing units which in some computational tasks can outperform central processing units by an order of magnitude or more, Mattsson (2016). The rapidly growing field of deep learning could be made apparent by Googles recent self-driving car and LeNet image recognition system, however, examples of machine learning can be found in just about every field with a classification or regression problem. Some of which include: natural language processing using morphological analyses for linguistics (Black & Zernik (1994)), document classification and spam detection, better image recognition than humans (Karpathy (2011)) and beating a professional player at the complex Chinese game Go (DeepMind (2016)). Figure 1.1 illustrates the advancement of machine learning using the context of game artificial intelligence players.

The majority of projects over the past few years have focused on Deep Learning due to its performance and versatility with complex problems but also because of the media coverage with large scale projects by international corporations such as Google and Microsoft. The idea of deep learning has been around since the 1980s where a

Japanese scientist, Kunihiko Fukushima, proposed a hierarchical, multi-layered artificial neural network named Neocognitron, designed for handwritten character recognition. Neocognitron was recognised as the inspiration for convolutional neural networks, the most commonly used deep learning algorithm, LeCun et al. (2015). Deep learning attempts to model complex abstractions in data by using a multiple-level architecture most commonly compromising of artificial neural networks and non-linear transformations in its algorithms, Mattsson (2016). As such, the concepts of deep learning are extensions of regular machine learning algorithms.

In the past year a new topic has become of consider interest in the machine learning field, Automated Machine Learning, which can be considered to cover the tasks of algorithm selection, hyperparameter tuning, iterative modelling and model assessment, Mayo (2017). By the end of 2016 the python Auto-sklearn library was created based off of the scikit-learn library which encompasses these tasks, created by a team from the University of Freiburg it won the KDnuggets AutoML challenge, Matthias Feurer (2016). Using these examples, its possible to hypothesize that the future of machine learning will include automatic deep learning, however, these individual fields of machine learning are still in their infancy and far from being used together.

The computer security industry due to its nature, has many examples of machine learning implementations such as intrusion detection systems using machine learning or deep learning for anomaly detection. One of which has been commercialised under the company name Deep Instinct and advertises zero-day detection using deep learning. However, the majority are very similar and are all blue team based security solutions. Researching into Red team or penetration testing tools using machine learning resulted in a disappointing lack of tools or ideas considering the vast amount available for Blue team.

The sole documented research found for red team was for an automatic penetration testing project named Auto Red Team (ART) framework by Lu, Song of Iowa State University in 2008 which used decision trees and hard programmed exploits. This meant the entire hard programmed exploit section had to be reconstructed for each use case, this would not be ideal but also extremely time consuming. Further analyses of the ART framework can be found in the literature review of "Auto Red Team: a network attack automation framework based on decision tree". There have also been tools and libraries created to test the security of software which use machine learning

models such as Deep Pwning. Deep Pwning is an open source metasploit plugin which allows the tricking of machine learning models. This field of research was named Adversarial Machine learning and the first paper of which was respectably named "Adversarial Machine Learning" and published by ACM in 2011.

There is an extensive amount branches to machine learning and unfortunately, too many to cover in this thesis due to time constraints. Therefore, this introduction will only detail the most popular types of machines learning algorithms.

1.3 Significance of study

The increased demand for penetration testers justifies the need for more effective and advanced tools to conduct their security assessments. The goal of a penetration test being to find vulnerabilities in a system using techniques similar to that of a malicious hacker. This means malicious hackers will continue to use the most advanced methods to gain access to critical systems and thus security teams must also continue to advance their toolset to be as effective and efficient as possible. Penetration tests can last a time scale of anything between one day to several months and more advanced machine learning tools would allow for a more efficient use of this time. The large variety of machine learning models used in the blue team results in a large variety of models required to test them using adversarial machine learning as well as advanced machine learning tools, specifically for the red team in order to bring each team to the same level. Having both red team and blue team on the same level is beneficial for the industry as a whole, providing competition between both sides and to continue to strive for improvement.

This project aims to help correct this unbalance by designing and developing a proof of concept red team tool using machine learning techniques.

The application must be able to be used as an aid to a security professional during a security assessment or capture the flag hacking events in order to be classed as a red teaming tool.

This application must also be scalable and versatile to be used in varying sized network environments.

The tool will be critically analysed and recommendations of related future work will be provided.

1.4 Types of Machine learning

As mentioned above there are many different machine learning algorithms available. Each algorithm can be classed based on the problem, required output and several factors of the data set, such as whether it includes labels and the amount of values it includes. The two primary problems machine learning algorithms provide solutions to, can be put in to two categories which are not mutually exclusive and can be combined in certain use cases. These are classification and regression problems. The following describes and states the differences of these types.

1.4.1 Classification

With the growth of big data, unstructured data is more prevalent than its structured counterpart. This is because, "while the amount of structured data has grown fast, the amount of unstructured data has grown much faster" Simon (2013). This creates the need for ever more efficient data analytics and is a typical classification problem for machine learning. There are many types of classification, simple types such as a linear classifier and then more complex types such as multiclass and structured classifiers. Classification is largely used in data mining and statistical analyses for these purposes. In short, classification is used when you require an input variable to be identified as part of a group or label, resulting in the full dataset being categorised. Classification can only take a finite set of values such as picking from 1 of N values. In classification, each incorrect answer is equally incorrect, compared to regression where incorrect answer can be varying levels of incorrect.

1.4.2 Regression

Regression problems are for when prediction of real continuous values is required, such as predicting stock market values and detecting the age of a person from a picture, Rossant (2014). Regression analyses involves predicting and estimating a response based on previous data and input variables. As mentioned above regression answers can have a varying level of inaccuracy as appose to binary correct or false predictions. This is due to regression using continuous values. Figure 1.2 provides a basic illustration example of these two problems.

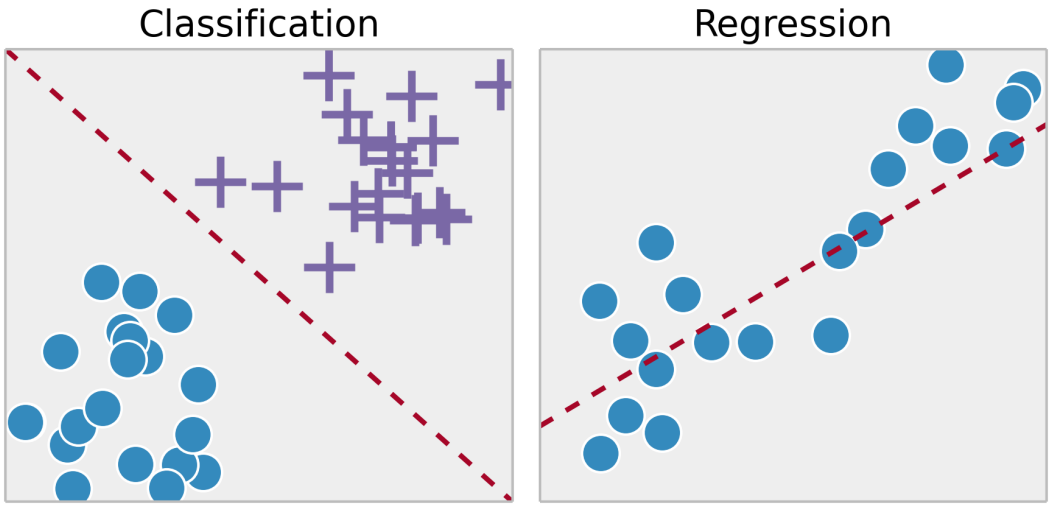

Figure 1.2: Illustration of machine learning types, Rossant (2014)

1.5 Types of Learning algorithms

The most commonly found learning methods for algorithms are supervised and unsupervised followed by semi supervised and reinforcement learning which are also fairy common. These are explained briefly as follows.

1.5.1 Supervised

In a supervised training model, the entire data-set used for training is pre-labelled so that the algorithm can use pattern analyses to predict given values after training. A use case scenario for supervised learning includes the above stock market example where current values with labels are given and the model is used to predict future labels.

1.5.2 Semisupervised

Due to the expensive nature of labelled data in most cases, semisupervised learning uses a split of labelled and unlabelled data to train the model. This is most often split unevenly with the large majority of training data being unlabelled. This model is often used in cases where the cost of labelled data-sets is simply too expensive.

1.5.3 Unsupervised

The opposite of supervised learning, where the training data-set has no labels and the model must attempt to determine the correct answer itself. This is often used as a method to determine a structure in the data given. These models can identify segments of similar attributes such as clustering. Unsupervised learning is the primary method used for this thesiss methodology.

1.5.4 Reinforcement

Reinforcement learning works similarly to heuristic algorithms in the sense that every possibility is attempted and assigned a score in which the iteration with the highest score is used as the model output. Whilst training this model the algorithm uses the highest scores iteration to modify its calculations for greater accuracy and efficiency. This model is often used in robotics as well as game design for path navigation calculation and computer player AI (artificial intelligence). The following is an example use case: A chess player AI calculates each possible move it can make using a reinforcement model for each of its turns. The model assigns a score to each move it could possible make at that point in time and weights them based on pieces acquired, future strategy prospects and defence risk. Similar models have been used for AIs mentioned in Figure 1.1.

1.6 Common Machine Learning Algorithms

As mentioned above, due to the amount of machine learning algorithms available and project time constraints, only a few of the most common algorithms will be described below.

1.6.1 Linear Regression

Linear regression predicts real values based off continuous data of two variables. It does this by using a best fit or regression line over the existing data extending in predictions. If the data does not indicate any positive or negative trends then using linear regression will likely not be a very useful model. The trend or direction of data can be calculated using correlation coefficient as follows, where $(x_2, y_1), (x_n, y_n)$ is the observed data.

$r = \frac{1}{n-1}\Sigma \left(\frac{x-\bar{x}}{s_x}\right)\left(\frac{y-\bar{y}}{s_y}\right)$

A value that is close to 1 would indicate positive correlation where -1 would indicate negative correlation. A normalised covariance calculation may also be used instead.

1.6.2 K-Nearest Neighbours

K-nearest neighbours (or KNN) is a widely used computationally expensive supervised learning model for data classification but can also be used for regression problems. The value k refers to the distance is which to weigh the number of class nodes inside. Several distance functions can be used such as Euclidean, Manhattan and Minkowski with the most common being Euclidean. Euclidian distance is the straight-line distance between the two nodes. On a two-dimensional plane it is measured using the following formula where the coordinates are $\mathbf{p} = (p1,\ p2)$ and $\mathbf{q} = (q1,\ q2)$.

$d(p, 1) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2}$

For e.g. if k = 1 then the current node will be assigned to the same class as its nearest node, where as if k = 5 then the class with the largest number of nodes in a distance of 5 will be selected to represent the current node. KNN is not resistant to bias in data-sets and thus data must be normalised before inputted into the model. KNN models are not to be confused with K-means clustering models as they share a very loose relationship being that KNN can be used to add data into pre-existing K-means clusters known as a nearest centroid classifier.

1.6.3 K-means Clustering

K-means Clustering is an unsupervised model in which the number of clusters is specified in advance. K being the number of clusters the model will output for the given data. The value for K can be assigned manually or the optimal value can be calculated by using either the elbow method or the gap statistic, the latter of which is used and explained in the methodology section of this thesis with a practical use of K-means. The K-means algorithm itself is also known as Lloyd's algorithm and uses iterative refinement to determine the best clustering.

1.6.4 Dimensionality Reduction

Dimensionality reduction algorithms are exactly what the name suggests. With the dramatic increase in variable types and raw data size being captured leading to the invention of the term Big Data, datasets have become very large. These large datasets have become the main bottleneck for machine learning performance,

especially in computationally expensive models such as KNNs and K-means. There needs to be a way to identify the significance of each variable in the data-set in order to give weights to their values for use in calculations. Principle Component Analyses (PCA) can be used for this purpose as well as for variance maximization. The PCA algorithm attempts to detect correlation between each of these variable types and uses this information to detect the vector directions of maximum variance in high-dimensional data. The algorithm then projects these maximum variance vectors into a smaller dimension whilst attempting to keep the most amount of information and greatest variance as possible. If the measurement scales of the dataset variables are not equal, then the data should be normalised prior to PCA due to the variance maximization functions of the algorithm.

1.6.5 Decision Trees

Decision trees are a classification model that uses supervised learning to calculate either categorical or continuous dependent variables. Decision trees can also be designed manually as was traditionally done. As the name suggests, the model works by splitting the data at each branch of the tree by asking questions and making decisions on each layer. The concept of decision trees is simple however they can be combined together, or include other machine learning models within its layers greatly increasing the complexity of this model.

1.6.6 Random Forests

Random decision forest is one of these complications where the model contains several decision trees built during the training stage and can also be used with unsupervised learning. These random decision forests can be used for classification or regression by simply using the mode of the results from the forest or the mean in the case of a prediction problem. They creation of this model differs from regular decision trees as the training algorithm applies bootstrap aggregating, commonly referred to as bagging, followed by feature bagging resulting in a random decision forest. Bagging increases the stability and accuracy of decision trees by reducing variance and avoiding the overfitting of data. It does this by generating additional sets data based on the given data-set which results in a larger data-set overall reducing the variance but increasing the predictivity of the model. Bagging can also be used on several other machine learning algorithms. Feature bagging also known as the random subspace method, selects random samples from the entire dataset in order to reduce the

correlation between each tree in the forest. This is done to produce varied models for each tree as appose to them being very similar if they were all trained on the entire data-set. Random forests can be complicated once more by adding a neural network within its layers, or even a deep learning model such as recurrent neural networks.

1.6.7 Aritificial Neural Networks

Artificial neural networks (or ANNs) are commonly used as the bases for most deep learning algorithms such as convolutional and recurrent networks and are loosely based on the way a human brain functions. They can be used with several types of learning methods. Neural networks consist of interconnected neurons on a layered scheme in which every neuron has a function and its output can be seen by its connected neurons to then use it in their own functions. Each neural connection has an assigned weight for its output which is calculated during the training stage of the model. The layers consist of three types, an input, an output and hidden layers sandwiched between the two. The number of hidden layers in a neural network is what defines whether it is a deep learning model or not. Models with more than one hidden layer are referred to as deep learning models. How the layers are connected and the way data travels between the layers defines the type of neural network. For e.g. a recurrent neural network allows for sequences to be used for input and output unlike convolutional which can only use fixed values for these. The Google subsidiary DeepMind uses recurrent neural networks in its algorithms to create its AlphaGo AI mentioned previously.

Chapter 2

Methodology

This chapter contains an in-depth analysis of the design and implementation stages carried out during this project. Initially, the design and goals of the application will be enumerated. Subsequently an analysis of the applications infrastructure will be presented followed by an in-depth detail of the modules and submodules within the applications programming. A test case scenario will be defined and executed to provide a proof of concept. The efficiency and accuracy among other factors based on the test scenario will be analysed during the discussion chapter of this thesis.

2.1 Design

The application designs primary goal was to be able to detect vulnerable machines on a large-scale network infrastructure regardless of topology or host types by using machine learning techniques and automated tool outputs. However, there are several requirements the application must adhere to for it to be a viable tool during a security assessment. The application created for this thesis was done so strictly on a proof of concept bases.

The application was designed with the following requirements in mind:

Text progress output with multiple verbosity settings allowing for an experienced tester to understand what the application is doing at any point in time during execution. This is critical as tools used during an assessment on live networks must not hinder or damage the network or its hosts in any way as to disrupt an organisations business.

Several input type parameters for which the tester can utilise based on the current information known about the network. Such as only using one type of scan file and manually selecting the clustering model.

Several output options including visually in the form of graphs and to a dot type file to be used with other industry applications and reports.

Manual overriding of variables via parameters in order to allow for the application to be scripted and modified by the tester. This will increase the efficiency of using the tool and provide advanced customisation of the algorithms within the application.

Highly versatile with working conditions and configurability. The programming of the application to be highly documented allowing a tester to fix and modify the application code to suit the operations needs. By using a primarily interpreted language as appose to compiled one would allow for this, as well as making the application portable without extra code. For these purposes, the Python programming language was chosen. With the majority of modern tools and scripts used by penetration testers haven been written in Python due to its versatility, reliability and portability, it further enforces this choice.

2.1.1 Application Brief

The application requires several parameters to run and has three different global modes; manual, assisted and automatic. These modes can either be run with Nmap, Nessus or both inputs with the majority of the use case scenarios requiring both. The application will then parse these inputs, process the data in several ways and cluster the information based on feature similarities. Output includes the full details of each cluster within the clustering. If using both inputs the application will combine the data from each and subtract large similarity clusters. By doing this, the application will determine the most unique hosts within the topology and display them in a new clustering. The unique hosts will, based on probability, be the most vulnerable on the network and should be prioritised by a tester during the manual assessment. This is due to the model prioritising the vulnerabilities each scanner detects then appends them to the pre-existing host set, thus rendering that host more unique than the others.

The difference between modes and parameters will be explained within the infrastructure analyses.

2.2 Infrastructure

- Link to figure
- Modes and parameters. Usage examples.
- Explain nmap cluster library from blackhat and others.
- Explain infrastructure.
- followed by an in-depth detail of the modules and submodules within the applications programming.
- Explain cluster subclass.
- Distance matrix and covariance coefficient matrix
- provide a description of the mathematical concepts used such as
- Gap statistic- http://web.stanford.edu/ hastie/Papers/gap.pdf
- The elbow method
- Create a Mathematical Model

2.3 Proof of Concept Testing

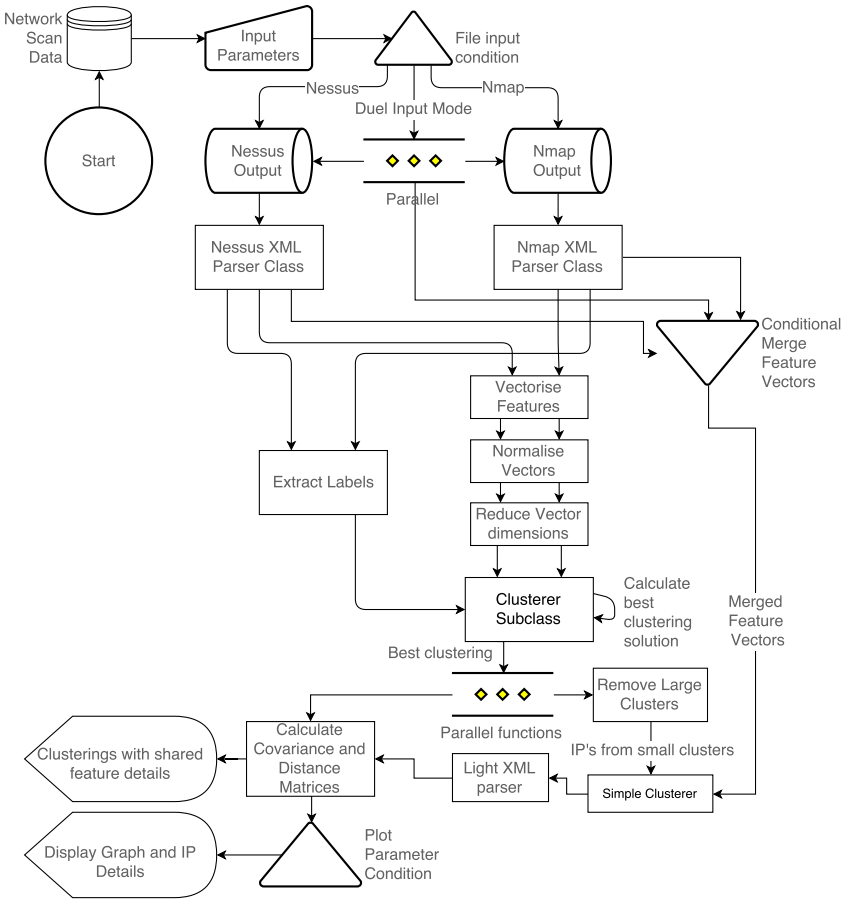

Figure 2.1: Application Infrastructure flow Diagram

Appendix A

Code

A.1 code1

Bibliography

Black, J. & Zernik, U. (1994), 'Method for natural language data processing using morphological and part-of-speech information'. US Patent 5,331,556.
URL: *http://www.google.com/patents/US5331556*

DeepMind (2016), 'Alphago'.
URL: *https://deepmind.com/alpha-go*

Karpathy, A. (2011), 'Lessons learned from manually classifying cifar-10'.
URL: *http://karpathy.github.io/2011/04/27/manually-classifying-cifar10/*

Kohavi, R. & Provost, F. (1998), 'Glossary of terms', *Machine Learning* **30**(2-3), 271–274.

Kurenkov, A. (2016), 'A 'brief' history of game ai up to alphago, part 1'.
URL: *http://www.andreykurenkov.com/writing/a-brief-history-of-game-ai/*

LeCun, Y., Bengio, Y. & Hinton, G. (2015), 'Deep learning', *Nature* **521**(7553), 436–444.

Matthias Feurer, Aaron Klein, F. H. (2016), 'Contest winner: Winning the automl challenge with auto-sklearn'.
URL: *http://www.kdnuggets.com/2016/08/winning-automl-challenge-auto-sklearn.html*

Mattsson, N. (2016), 'Classification performance of convolutional neural networks'.

Mayo, M. (2017), 'The current state of automated machine learning'.
URL: *http://www.kdnuggets.com/2017/01/current-state-automated-machine-learning.html*

Munoz, A. (2012), 'Machine learning and optimization@ONLINE'.
URL: *"https://www.cims.nyu.edu/ munoz/files/ml_optimization.pdf"*

Rossant, C. (2014), *Introduction to Machine Learning in Python with scikit-learn*, Packt Publishing.
URL: *http://ipython-books.github.io/featured-04/*

Simon, P. (2013), *Too Big to Ignore: The Business Case for Big Data*, Wiley and SAS Business Series, Wiley.
URL: *https://books.google.co.uk/books?id=1ekYIAoEBrEC*

Valiant, L. G. (1984), 'A theory of the learnable', *Communications of the ACM* **27**(11), 1134–1142.