FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

A Machine Learning Approach for Path Loss Estimation in Emerging Wireless Networks

João Rafael de Figueiredo Cabral

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Rui Lopes Campos, PhD

Second Supervisor: Helder Martins Fontes, MSc

February 22, 2019

A Machine Learning Approach for Path Loss Estimation in Emerging Wireless Networks

João Rafael de Figueiredo Cabral

Mestrado Integrado em Engenharia Informática e Computação

February 22, 2019

Abstract

The development of new wireless communications systems relies on simulation as an important tool for preliminary validation of new designs and for iterative improvement of implementations. Simulations are based on simplified models of the network and the nodes which compose it. One particularly important aspect modeled in simulation is the signal path loss. Yet, the simplified models used yield a significant loss of accuracy when applied to the simulation of highly dynamic emerging networking scenarios such as aerial networks. Current approaches to simulation in scenarios where traditional path loss models result in insufficient accuracy include the application of trace-based path loss models, where the behaviour of the physical layer of the network is replicated using data from past real experiments. Such approaches can more faithfully model the behaviour of the physical layer, but reduce the flexibility of the simulation, which is limited to scenarios mimicking the duration, mobility model, and the number of nodes of the experiments from which data was gathered.

The aim of this dissertation was to evaluate the application of machine learning techniques to data from past experiments to estimate the path loss for new trajectories of a mobile node such as an Unmanned Aerial Vehicle. This dissertation proposes a machine learning based approach for generating custom-tailored path loss models in the context of Emerging Wireless Networks such as Aerial Networks. The proposed approach includes a method for 1) preparing and cleaning a dataset gathered from a real experiment and 2) evaluating the accuracy of a selected set of machine learning models to assist the user in finding the best model to apply. The proposed machine learning based approach was implemented as a software platform to simplify and automate the application of machine learning techniques by networking researchers. To validate this approach some machine learning techniques were evaluated and their accuracy benchmarked against theoretical path loss models.

Resumo

O desenvolvimento de novos projetos na área dos sistemas de comunicações depende de técnicas de simulação como uma importante ferramenta para validação preliminar de novos modelos e para a melhoria iterativa de implementações. Estas simulações recorrem a modelos simplificados da rede e dos nós que a compõe, sendo a atenuação do sinal um aspeto particularmente relevante da realidade a modelar. Porém, tais modelos simplificados resultam numa perda significativa de precisão quando aplicados à simulaçao de cenários altamente dinâmicos, como sejam os cenários que envolvem veículos aéreos. As abordagens atuais para a simulação de tais cenários passam pela aplicação de modelos de atenuação *trace-based*. Nestas abordagens, o comportamento da camada física da rede é replicado usando dados provenientes de experiências reais passadas. Este tipo de modelo possibilita reproduzir de forma mais fiel o comportamento da camada física, mas reduz a flexibilidade da simulação, que fica limitada a cenários que apresentem a mesma duração, modelo de mobilidade e número de nós das experiências de onde provêm os dados usados para gerar os *traces*.

O objetivo desta dissertação foi avaliar a aplicação de técnicas de *machine learning* a dados provenientes de experiências passadas para estimar o comportamento da atenuação do sinal para novas trajetórias de um nó, como seja um veículo aéreo não tripulado. Esta dissertação propõe uma abordagem para a geração de modelos de atenuação específicos de um determinado cenário no contexto de uma rede móvel emergente como seja uma rede aérea. A abordagem proposta inclui um método para 1) preparar e limpar o *dataset* recolhido em experiências reais e 2) calcular métricas de precisão para diversos algoritmos de *machine learning*, para auxiliar o utilizador na identificação do melhor tipo de modelo a aplicar. A abordagem proposta foi implementada como uma plataforma de *software* para simplificar e automatizar a aplicação de técnicas de *machine learning* por investigadores na área das comunicações sem fios. Para validar esta abordagem, algumas técnicas de *machine learning* foram avaliadas e a sua precisão comparada com a de alguns modelos teóricos de propagação.

Acknowledgements

Now, at the closing of the chapter of my life dedicated to my formal education, it is appropriate to acknowledge and give thanks for the friendship, collaboration, and generosity of all who helped provide me with this opportunity.

I would like to thank my supervisors, Rui Campos and Helder Fontes, for their assistance in the development of this dissertation. Without their advice and contributions it would surely not have been possible to carry it through to completion.

I must also express my utmost gratitude to my family, whose tireless support and considerable investment in my education I cannot possibly have done enough to deserve.

A final word must be dedicated to my friends, who accompanied me through the best and worst of times. Your presence and companionship were invaluable. Any irrascible outbursts I may have inflicted upon you at the darkest moments of this journey are not, could not possibly be, a worthy reward for your kind-heartedness. Thank you for all your support.

João Cabral

"But all science would be superfluous if the outward appearance and the essence of things directly coincided."

Karl Marx

Contents

List of Figures

LIST OF FIGURES

List of Tables

Abbreviations

ACK	Acknowledgement
AMQP	Advanced Message Queuing Protocol
API	Application Programming Interface
BASE	Basically Available, Soft State, Eventually Consistent
CRUD	Create, Read, Update and Delete
GPS	Global Positioning System
HTTP	Hyper Text Transfer Protocol
NoSQL	Not only SQL
ORM	Object Relational Mapper
REST	Representational State Transfer
RSSI	Received Signal Strength Indication
SQL	Standard Query Language
UAV	Unmanned Aerial Vehicle
URL	Uniform Resource Locator
WWW	World Wide Web

Chapter 1

Introduction

1.1 Problem and Motivation

When a new application or protocol is being developed for a wireless communications system, the software and network engineers usually rely on simulation and experimentation to perform the necessary evaluation and validation of the solution being developed. Experimentation is essential as it provides real-world results, although it usually incurs significant expenses and logistic challenges, which make a proper validation in large scale impractical to achieve. Furthermore, real world experiments are often influenced by complex phenomena, such as the mobility of nodes and multipath, making them very difficult and expensive to reliably repeat and reproduce. Simulation, on the other hand, allows the developers to easily create complex simulation environments, in a fully controllable, repeatable and easily reproducible fashion, while reducing logistic costs. This allows validation of a higher number of scenarios, and of a larger scale, when compared to a typical real world experiment. Nonetheless, simulation uses models that are inherently abstractions of complex real world phenomena, resulting in less accurate results. For instance, complex and very dynamic scenarios such as the ones involving Unmanned Aerial Vehicles (*UAVs*) are the most challenging to simulate, given the less accurate results mainly caused by the simplified path loss models. These scenarios are also the most challenging to run experiments on, since they are difficult to repeat, and their results difficult to reproduce. Thus, the validation of the solution under development may become compromised.

In order to solve this problem researchers have proposed and implemented an approach where datasets (hereinafter the terms traces and datasets are used as synonyms) from past wireless experiments can be used to inform a simulated path loss model [FCR17]. Such a model, known as a trace-based path loss model, is extracted from the real Received Signal Strength Indication (*RSSI*) of past experiments. Using the trace-based simulation approach the developers can continue to evaluate the solution being developed in close-to-real simulations, reproducing the past

experiments conditions. This approach, however, is limited to reproducing the same exact scenario captured in the experimental traces. In a trace-based simulation it is not possible to change the trajectory, number of nodes used, and the durations of the experiment to test different scenario configurations.

It is thus important to overcome the limitations of trace-based simulation, allowing to utilize experimental traces more flexibly for generating new scenarios instead of replaying past scenario conditions. More specifically, the goal is to be able to use information about the behaviour of the signal in a particular environment and given a particular type of node to predict the behaviour of the path loss for any node trajectory in the scenario.

1.2 Objectives

The objectives of this dissertation are two-fold:

- Explore how the information embedded in datasets extracted from real-world experiments can be applied to generate new environment-specific propagation models, in which new trajectories, different numbers of nodes, and experiment durations are represented, which may, in turn, be used to feed trace-based simulations.
- Develop a software platform which allows the user to store datasets from real experiments, train propagation models using a set of machine learning algorithms, and retrieve error metrics to assist the user in the selection of the best algorithm for each environment.

1.3 Challenges

Three main challenges can be identified as obstacles to achieving the objectives laid out in Section 1.2:

1. Identifying the most relevant information which can be extracted from the traces of real world experiments to train the machine learning models.
2. Selecting machine learning algorithms for training the models which will be used to generate the new environment-specific propagation models.
3. Cleaning and preparing the datasets for the application of the machine learning algorithms to a specific dataset which will be used to validate the approach.

1.4 Main Contributions

The two main contributions provided by this dissertation are:

1. A machine learning approach to generate custom-tailored path loss models, reflecting the specific characteristics of a given environment and mobile node, based on traces from real past experiments.

2. A software platform to store traces, evaluate, and select the best machine learning model to approximate the real world conditions described by the traces, and generate new, synthetic traces which can be used as inputs for trace-based path loss models in simulation.

1.5 Document Structure

This document is structured as follows: Chapter 2 reviews the state of the art on trace-based simulations and machine learning techniques applicable to this domain alongside the related work which has been published. Chapter 3 contains a description of the proposed machine learning approach. In Chapter 4 a description of the software platform which was implemented to automate and store the results of the machine learning path loss models is provided, going into detail on the components and technological decisions taken in the development of the system. Finally Chapter 5 draws the main conclusions and points out the future work.

Chapter 2

Literature Review

This chapter reviews the state of the art on the important topics related to this dissertation. This review is divided in four sections, concerning: path loss models and their application in the context of network simulators; mechanisms for perpetuating network experiments in simulation; machine learning, with a specific focus on training techniques for regression models; and related work.

2.1 Path Loss Models

Path loss Models are used to predict the signal attenuation characteristics of a given propagation medium, and are useful for estimating network coverage and throughput. This work focuses on the application in the context of wireless communication. Nevertheless significant variation exists in terms of parametrisation, purpose, and application between different models.

In the context of wireless and radio communications path loss is described by [PSG13] as the sum of the factors that cause attenuation of the signal between the transmitter and the recipient, as shown in Equation 2.1, where PL is the path loss, L_t is the free-space loss given by the characteristics of the antennas, L_s is the slow fading effect of buildings and other objects, and $L_f(t)$ is the fast fading component which varies with time and frequency. $L_f(t)$ is usually not modelled by path loss models, or computed stochastically, following a probability distribution.

$$PL = L_t + L_s + L_f(t) \tag{2.1}$$

Several authors have performed surveys on the field of path loss models, proposing taxonomies for categorising the different models according to the methodology they employ, the physical phenomena they take into account, and the field of application [PSG13, Uni12].

Most path loss models use *a priori* information to model the path loss characteristics of the link, where electromagnetic propagation is predicted according to idealised models and information on the physical properties of the scenario where the model is to be applied. The simplest, most basic of these models, take into account the path loss over free space. Others take into account

the shadowing effects caused by obstacles of various types as well as the multipath components caused by reflection and other phenomena. These models make no use of concrete measurements of signal strength to inform or alter the output of the calculation, although the employed formulas may be based on empirical observations. In particular, some of these methods are based on models collected in different but similar environments, leading to models which are especially well suited for signal strength prediction in the context of urban [AP77], rural [MK00] or suburban areas [EGT+99]. Another approach consists in the usage of a ray tracing algorithm to calculate multiple different paths from the transmitter to the receiver and estimate the loss according to each path. This approach, while computationally intensive and demanding in terms of knowledge of the geometry of the region to be studied, can be used to calculate the delay spread and consequently the effects of inter-symbol interference. Finally, some models are concerned not with modelling path loss as a whole but a specific component of path loss, or the effect of a specific circumstance on path loss. These models, while not useful on their own, may be refined by the use of other models, producing more realistic results.

Other models use explicit measurements of the behaviour of the medium under evaluation across a given area, assuming that the characteristics of the medium are too complex to model analytically or empirically. These models define methods for gathering data points and for interpolating them in order to obtain estimates of signal quality at geographical positions which were not sampled. One of the first examples of such a method is given by Lee in [Lee85], where the author establishes the means to gather local averages of signal strength, noting that instantaneous readings are not a reliable measure of path loss effects, as they are affected by fast fading phenomena. A common approach is to divide the space into a grid where measurements are taken and then interpolated by some mechanism. In [FP04] a similar mechanism is used to estimate link quality by having multiple nodes take measurements and then use a voting mechanism to reach an average value for link quality with regards to a wireless network.

2.1.1 Path loss Models in Network Simulators

One of the areas where path loss models assume a vital importance is in the area of network simulators. The goal of a network simulator is to allow for experiments to be ran without the associated cost in equipment and logistical work that is present when real experiments are ran. In the context of wireless communications, simulators require models which enable them to approximate the real world signal strength measured at the receiver between each pair of nodes. This signal strength is then used to calculate the behaviour of the link in terms of availability and throughput.

One approach used by network simulators is to apply known path loss models in the context of the simulation, by feeding the aforementioned models with the inputs which correspond to the positions of the nodes in the simulation. In the case of the ns-3 network simulator, for example, the simulation may be configured to estimate link strength based on several models available in the literature. Additionally, such models may be chained, allowing for the application of supplementary models which describe the behaviour of specific components of path loss as previously

described. The three most used path loss models for network simulation are the Friis model, the two-ray model and the log-distance model. In the sections that follow, we refer to each of them.

2.1.2 Friis Model

The Friis model, first described in [Fri46], is a simple model which describes the path loss between two antennas based on the wavelength of the signal (λ), the transmission power (P_t), the distance between the antennas (d), the gain of the transmitting and receiving antenna (G_t and G_r respectively) and the effective area of both antennas. The ns-3 model assumes isotropic antennas with no heat loss are used, resulting in the $\frac{\lambda^2}{4\pi}$ expression. The resulting received power P_r is then given by Equation 2.2. This model is a very simple representation of the electromagnetic properties of radio signals and does not take into consideration any shadowing effects caused by obstructing objects.

$$P_r = \frac{P_t G_t G_r \lambda^2}{(4\pi d)^2} \tag{2.2}$$

2.1.3 Log-distance Model

The log-distance model is used to predict loss propagation in situations where shadowing occurs. It uses a reference consisting of a distance (d_0) and a loss value (L), and a path loss exponent (γ), which is set based on empirical observations, to estimate path loss (L) according to Equation 2.3. In situations where $d < d_0$, the path loss is equal to the reference value for path loss, calculated using the Friis model [F+].

$$L = L_0 + 10\gamma \log 10(\frac{d}{d_0}) \tag{2.3}$$

2.1.4 Two-Ray Model

The last of the commonly used models in simulation, the two-ray model, is used when the constructive or destructive effect of reflection on the ground are important for the result of the simulation. This model divides the path loss components into a direct line of sight component and a reflection component originating from interactions between the signal transmitter and the ground. Figure 2.1 presents an illustration of a simple two-ray model scenario. Phase differences between both signals account for path loss effects that cannot be rigorously modeled according to empirical approaches such as the one described in Section 2.1.3. In [SJD12] the authors present Equation 2.4 to model path loss according to the two-ray interpretation, where φ represents the phase difference between the two components and Γ represents the reflection coefficient, resulting in a complex and computationally heavy calculation. Most simulators use the simplified Equation 2.5 which works well after a given critical distance between transmitter and receiver has been exceeded. In particular, ns-3 approaches this problem by calculating a crossover distance, below which it will use

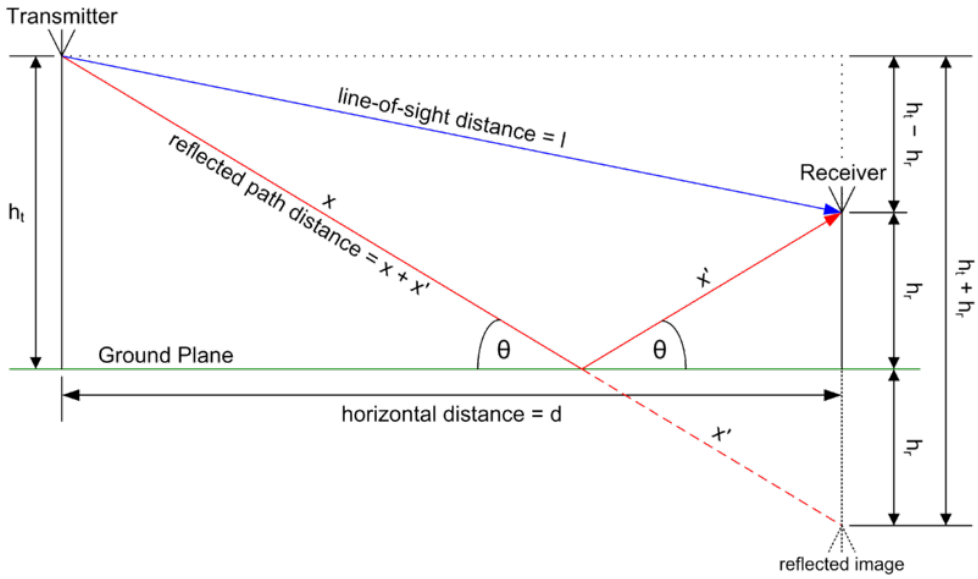

Figure 2.1: Two-Ray model illustration [Cha14].

the Friis model described previously, and above which the simplified equation is used to calculate signal strength. However, in the previously referenced paper, the authors show that such an approximation may not wield considerable benefits over the log-distance model when performing simulations of a vehicular environment.

$$L = 20\log 10(4\pi\frac{d}{\lambda}|1+\Gamma_{\perp}e^{i\varphi}|^{-1}) \tag{2.4}$$

$$L = 20\log 10(\frac{d^2}{h_t h_r}) \tag{2.5}$$

2.2 Perpetuating Network Experiments through Simulation

Network simulation is a useful tool for simulating scenarios without the need for real world experimentation while iterating on new protocols or network solutions. Typically, it uses path loss models to estimate the behaviour of signal propagation. Such models are abstractions of reality which yield acceptable results for static scenarios but may degrade when used in unstable scenarios where nodes travel quickly, their geometry causes intermittent obstruction of Line of Sight between nodes, there is antenna misalignment, among other factors. In such scenarios, the error of path loss models rises significantly, leading to optimistic results when compared to the results observed in experimentation.

This situation has led to the development of multiple solutions aiming to utilize information extracted from real world experiments, in order to allow for the execution of more accurate simulations. One approach to reproduce real-world conditions inside a simulator is proposed by Owezarski *et al.* in [OL04] and consists of using flow-level information based on traces of packets

sent through a real network and to then accurately reproduce them inside the simulation. Agrawal *et al.* [AV16] take a different approach, using instead application level traces to simulate the delays and throughput conditions experienced by the user at the application level, foregoing reproduction of the real-world conditions of the lower layers inside the simulation. Such an approach provides a more faithful model of the behaviour of real world applications than the models typically provided by network simulators, and by ns-3 in particular. While both these approaches provide advantages over plain simulation techniques with regards to the accuracy of the results they provide, by their nature they constrain the simulation to represent only the data traffic behaviour that was observed at the moment of data acquisition. This makes it difficult to use these approaches to provide an environment where network researchers can interactively design or improve network protocols or applications and iteratively change the design based on simulation results.

As an answer to the difficulties and limitations presented, Fontes *et al.* proposed a Physical layer replay approach [FCR17], where the path loss model used by the simulator is fed with information captured in a real world scenario and faithfully reproduces it in the simulation. Under this type of replay system the network simulator is responsible for controlling the behaviour of the upper layer of the network. This allows users to freely specify and implement network and application layer changes and analyse their impact while the physical layer behaviour is reproduced by replaying the behaviour of the physical layer, captured in a real-world scenario. This approach is thus suitable for iteratively improving protocols and applications and testing the results based on a faithful reproduction of the characteristics of the physical channel. In [FCR18] the author presents an improvement of this approach which allows it to be applied in multiple access mediums such as a wireless channel.

The physical layer replay approach to network simulation allows users of the simulator to iteratively improve their designs. However, it comes with the disadvantage that any test and validation scenario must faithfully adhere to the topology of the network used at the time of data acquisition, as well as to the duration of the original experiment. This presents two main problems, which the present work tries to address:

1. The behaviour of the protocol or application may drastically change based on apparently innocuous changes to the topology of the network.

2. The deployment scenarios for the solution under development may consist of a number of nodes which is too large and consequently expensive to acquire data for in a preliminary test.

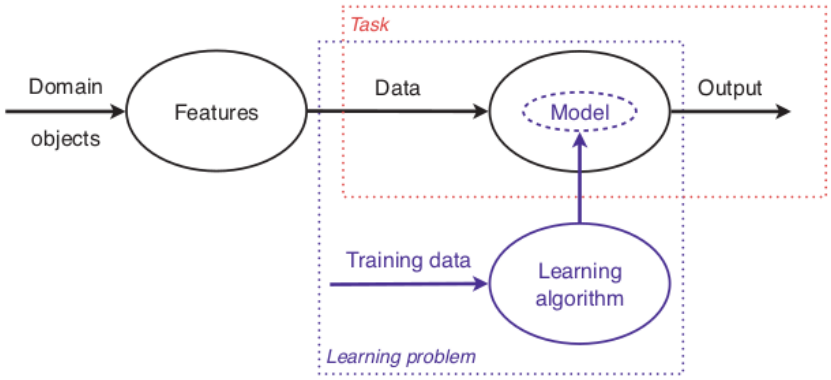

Figure 2.2: Illustration of the machine learning process [Fla12, p. 11].

2.3 Machine Learning

In the context of Artificial Intelligence, Machine Learning is the field of study which describes the mechanisms which allow a computer to learn to discern patterns from a given set of data. It aims to allow a system to improve its capability to perform a specific task without being explicitly programmed. According to Flach, machine learning is the process of building a model which allows the computer to perform a certain task based on a set of features provided by the training data [Fla12, p. 46]. This model is illustrated by Figure 2.2. Generally speaking a machine learning model is obtained by feeding an initial set of observations to an algorithm which trains a model used to perform predictions when given a new data item. Each piece of data which is presented to the algorithm for training purposes is generally composed of several features representing different aspects of the reality being modelled. The algorithm is expected to infer the correlation between the values of the features in the training data and a set of target variables whose value it aims to predict. The continuity and nature of the domain of both the entry features and the target variables are an important distinctive feature when building a taxonomy of machine learning methods. In particular, considering the domain of the target variables, machine learning algorithms can be divided into 1) classification algorithms when considering discrete, categorical domains, or 2) regression algorithms when considering continuous domains. While machine learning algorithms have advantages in that they need not be explicitly programmed to improve their predictive performance based on new evidence in the form of additional data, there are some drawbacks to their application. First of all, a machine learning algorithm, if not properly parameterised, is prone to a phenomenon known as overfitting, where the algorithm does not sufficiently generalize the model to be able to make predictions outside of the values which it has previously seen while in the training phase. For similar reasons, machine learning algorithms have limited applicability outside of the specific scenarios in which they were trained, particularly if the correlations on which it depends are specific to the context and the discrepancies with other settings accurately modelled by the selected features, or there is a lack of data for other settings.

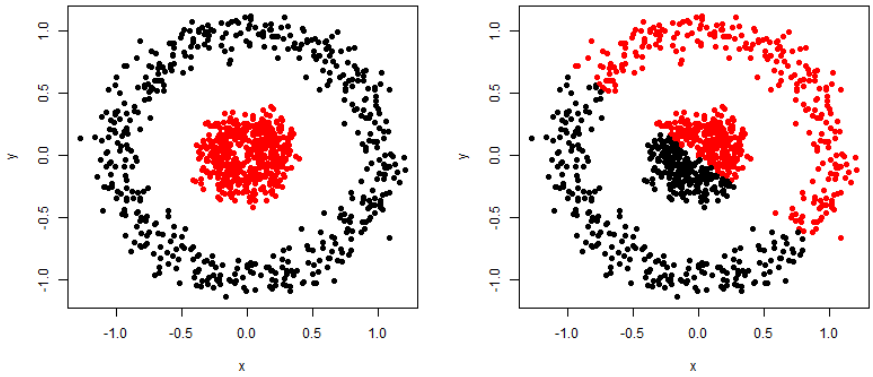

Figure 2.3: Different possible clustering outcomes [Dey17].

2.3.1 Unsupervised Learning

Unsupervised Machine Learning techniques are the set of techniques that are trained with data which does not contain a label to be predicted or any other indication of the desired outcome. This means that the input data for a unsupervised learning algorithm is identical to the data the algorithm will encounter after it has been trained. The usual goal for unsupervised learning algorithms is to identify substructures of samples which are useful for organising data [SSBD14, p. 308], known as a clustering of data. The accuracy of the clustering process is difficult to evaluate because there is no objective measure of what partition of the data into separate clusters is the most useful. This problem is illustrated by Figure 2.3 where two different clusters are identified by two different abstract processes and where the correct approach depends on user evaluation and not on an objective measure.

In this dissertation we focus on developing models based on training data where we possess a correct labelling of the outcome of past events, and as such this type of technique is not directly applicable to our problem.

2.3.2 Supervised Learning

Supervised Learning, in contrast with Unsupervised Learning, is the process of using correctly labeled data which has been acquired from past experimentation or observation to train a model which correctly maps a set of features into one or more outputs when presented with new examples of the same type on which it was previously trained. In [RN03] this process is more formally defined as the process of finding, for each input-output pair in $(x_1, y_1)(x_2, y_2)...(x_n, y_n)$ for which a function f such that $f(x_i) = y_i$ exists, a function h which approximates f.

As previously mentioned the nature of the domains of the y variables can be discrete or continuous, resulting in either a classification problem or a regression problem. In this dissertation we are tackling the problem of calculating wireless network signal strength, which due to its continuous nature makes for a regression problem.

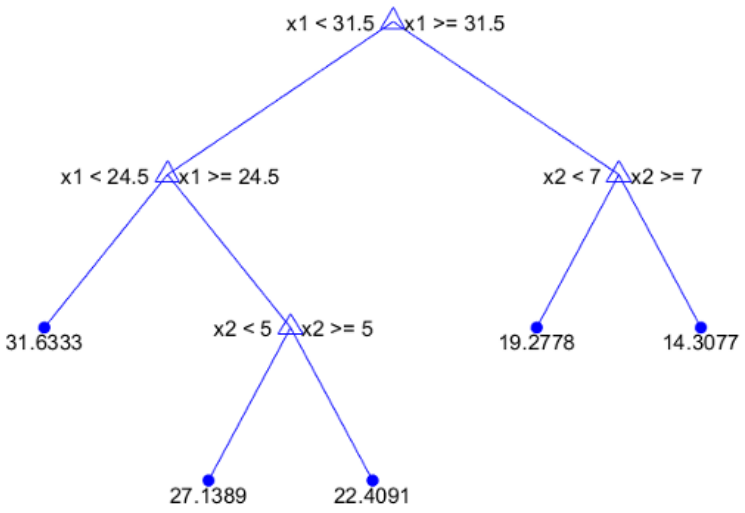

Figure 2.4: Illustration of a regression tree. [Mat18]

2.3.3 Decision Trees

A decision tree is one of the simplest supervised learning algorithms. It models the predicted outcome of an input as the path from the root of a tree to a leaf, where the decision of which branch to take is based on the value of a specific feature of the input. In the case where the domain of the feature is continuous this results in up to an infinite number of branches on each node corresponding to such a feature, which in turn may lead to the model overfitting based on the input samples [SSBD14, p. 251].

Such decisions trees are usually built recursively by splitting the input space according to the feature that most reduces the entropy in the results, i.e.m the feature that results in the most information gain. In the case of continuous values for the label this is usually performed using a sum of some measure of the variance in each of the new sets which is generated, with the goal of minimising that value.

Decision trees are advantageous in that the resulting model is fairly simple to analyse and understand, leading to intuitive results and easing the understanding of problems which may arise from issues such as overfitting or excessively deep trees. However, when used for regression, decision trees are not able to predict values in the full domain of the output variable, given that it can only classify inputs into any output which it has previously seen, making the coverage of a continuous domain impossible and leading to potential loss of information. Figure 2.4 serves as an illustration of such a tree and its limitations.

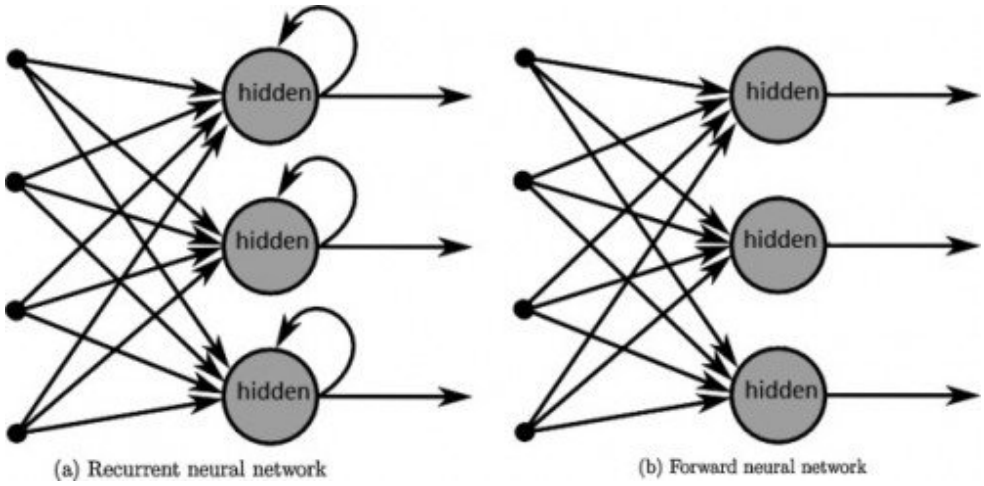

Figure 2.5: Recurrent vs feed-forward neural networks [dmBM14].

2.3.4 Neural Networks

Neural networks are structures vaguely inspired on the behaviour of the human nervous system. On a neural network each node is organised in layers which are ordered between themselves. Each node is connected to one or more layers on the following layer by a link to which a weight is associated, and in turn receives inputs from similar links from nodes in the previous layer [RN03, p. 728]. The input features constitute the values of the first layer's nodes, and the output of the model is given by the last layer. Between each layer an activation function is used to remove the linearity from the model, without which it would only be capable of representing simple linear functions and would tend to perform about as well as a simple linear regression. Some possible activation functions are hard threshold functions or logistical functions [RN03, p. 729]. In the case of regression neural networks Specht *et al.* [Spe91] state that exponential activation functions are commonly employed.

Many network topologies are possible, with the broadest possible categories being 1) feed forward networks, where each set of nodes is only connected to nodes belonging to layers which are closer to the output layers, and 2) recurrent networks where layers may feed information back into their inputs. The difference between both types of networks is illustrated by Figure 2.5. Another important decision when building a neural network consists on determining the number of hidden layers, which are the layers which rest between the input and the output layer, as well as determining the number of nodes to be present in each such layer and the density of the connectivity between nodes.

The training of such a network is performed by feeding it labeled training data and propagating the error at the output back into the previous layers in a process intuitively known as back-propagation. For each input the features are fed into the input layer and an output and associated error is calculated. The weights of the links leading to the output layer are adjusted according to the value of the error, and then the process is repeated with the weights of the links leading to each

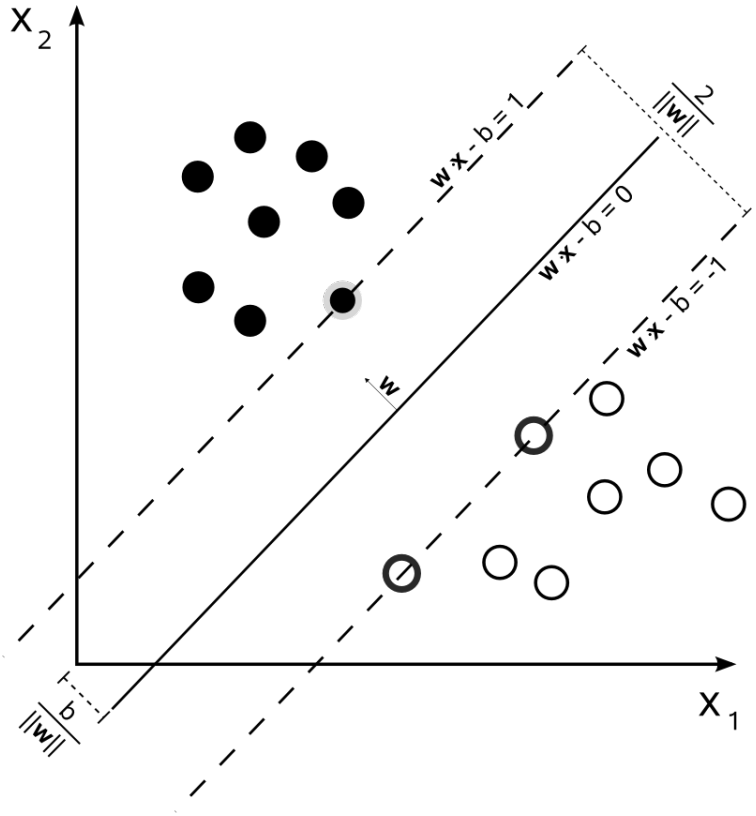

Figure 2.6: Linearly separable SVM problem [Lah18].

previous layer until all links have been updated. Then, the next value is fed into the network until all samples in the training set have been presented to the network an adequate amount of times, or until the average error of the network for the training samples falls below a particular threshold.

2.3.5 Support Vector Machines

Support Vector Machines are a supervised learning model which works by separating the different classes via a plane that maximizes the margin between the most similar samples of each class. According to [RN03, p. 745] the choice to maximize the margin relies on the assumption that the data points the model will classify during its operation are drawn from the same distribution from which the training data points were drawn. Consequently, it can be reasonably expected that by maximising the margin between the closest instances of each class and the decision line, the generalisation loss of the model will be minimised. This process of maximisation of the margin between both classes of data points is shown by Figure 2.6. The model deals with non-linearly separable problems by mapping the input space into a higher dimensional feature space where the problem becomes linearly separable and subsequently maximizes the margin on that space. This so called *kernel trick*, where the value of each feature is mapped into an additional set of features via a set of functions, is illustrated by Figure 2.7. In this particular instance the functions used were $f_1 = x_1^2$, $f_2 = x_2^2$ and $f_3 = \sqrt{2}x_1x_2$.

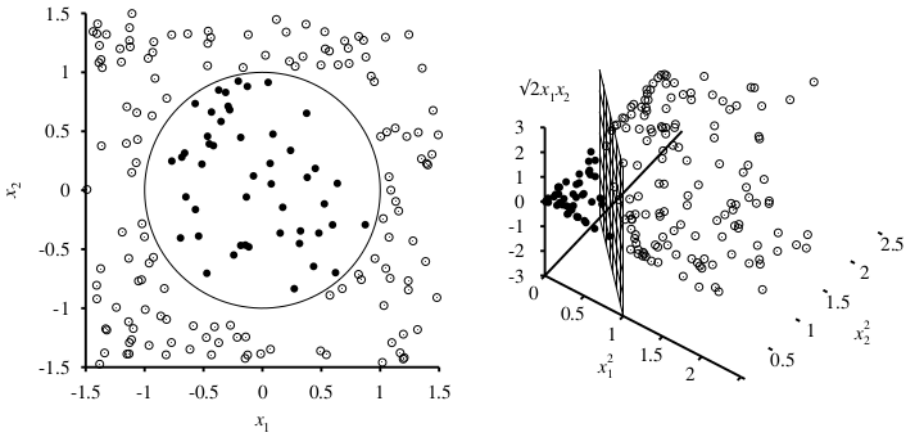

Figure 2.7: Mapping of a two-dimensional feature space into a three-dimensional feature space [VVLB+08].

2.3.6 Adaboost

Boosting is an approach to machine learning which seeks to combine several weak learners to produce a strong classifier by iteratively training each weak classifier on an weighted version of the dataset which was used to train the previous classifier. A weak learner is a model which produce results which are only slightly better than a random guess, while a strong learner is a model which much more closely predicts the value of the target variable [Sch90].

Adaboost works by constructing an ensemble classifier based on a set of weak classifiers. All classifiers are trained on a set of data-points drawn from the same dataset. As a classifier is trained, a weight is assigned to it corresponding to the strength of the classifier in correctly predicting the target variable on the samples it was presented with. Furthermore, for each sample in the dataset there is an associated probability for that sample to appear in the training data of a weak classifier. Samples which are incorrectly classified after a weak learner is trained have their probabilities increased, while correctly classified samples have their probabilities decreased [Sch13]. The trained classifiers are then combined according to their weight, resulting in the final classifier. While Adaboost is typically used with very shallow, two-level decision trees called decision stumps, other weak learners can be employed.

2.4 Related Work

In wireless communication scenarios comprised of mobile nodes using stationary base stations as intermediaries, such as in an infrastructured 802.11 network, a strong correlation exists between signal strength and the position of the mobile nodes. This relationship arises due the the attenuation characteristics of the wireless medium. Simultaneously, the shadowing component of the path loss is typically caused by buildings, walls or other static objects. Consequently, for any fixed position, it does not vary a lot over time. This characteristic of mobile communications has been

exploited to develop solutions which estimate the position of a given mobile node given a set of communication characteristics which can be gathered at the stationary node of the network. These techniques have applications in the context of locating individuals inside public buildings and detection of unlicensed usage of restricted frequency bands, among others. On the other hand, the aforementioned statistical correlation can be used to infer communication characteristics of hypothetical nodes at a particular location, given a series of observations gathered of similar mobile nodes at different locations inside the same communication context. The following paragraphs present some of the work done by researchers in exploiting these characteristics.

In the survey of Path Loss prediction models [PSG13], Phillips *et al.* discuss some techniques for predicting local path loss by taking measurements of signal strength at various points of this area and using interpolation to infer a model which may be applied to the entirety of the area under study. In particular, the author references findings by two other authors [Shi10, Lee85] which show that for areas in the range of two to three wavelengths of the signal being modelled it is possible to use the local average of the signal as a predictor of signal strength in that area since fast fading effects are either averaged out or minimised by signal modulation. The author identifies two main types of techniques which may be used to perform the interpolation. In the first one the area under analysis is explicitly mapped by using probes which sound the channel for signal strength. The gathered data is then used to determine the average signal strength for each element on a representation of the area divided in grids or cells. The second approach relies on partitioning the area into its constituent spaces and identifying, for each space, the types of obstructions which may cause slow fading effects on the signal being received. By independently calculating a loss function for each type of obstruction, such a model could in theory be used to calculate path losses in other environments presenting the same type of obstacles. Finally, the author identifies an open area of research based on applying active learning techniques to train models based on measurements of signal strength. These techniques consist in selecting the next set of measurements to be used for training purposes which would most improve the accuracy of the model, given a model trained based on some measurements.

A mixed approach between theoretical and measurement based estimation of path loss is illustrated by Robinson *et al.* in [RSK08]. The two notable aspects of this work on estimating network characteristics in a urban-scale wireless network consist of 1) the usage of heuristics to identify regions where the performance of the network varies monotonically with the distance of the node from the access point, and 2) a measurement based additive factor being used together with a traditional log-based model to estimate the characteristics of the signal being predicted. On the topic of using learning techniques to estimate signal strength the ASTRO project [PLK18] uses machine learning on a set of autonomous drones designed to locate unlicensed or unlawful usage of certain ranges of the frequency spectrum. In order to achieve their mission the drones must solve a problem similar to the problem this dissertation addresses, namely the relationship between the position of a transmitter and the strength of the signal. The solution adopted by the authors consists of using the log-distance model to estimate the distance between the nodes and the target, using a machine learning technique to progressively tune the parameters of the model

in order to obtain better accuracy. On a later phase of the operation the results are then used to perform triangulation between the distances estimated by all drones to locate the target. This process requires that the drones have been carefully positioned in order to maximise the accuracy of the triangulation process. The chosen algorithm is the batch gradient descent algorithm which is suitable to the online learning nature of the problem, as well as to the low computational power present onboard of a drone. While this choice is adequate to tackle the specific problem of the ASTRO project there are significant differences between the environment of that project and the goal of this dissertation, namely in that we do not require online learning capabilities and our computational environment can better tolerate heavier but more accurate algorithmic choices.

As we have seen many trained path loss models rely on knowledge of the topology of the area where the model is to be applied, using information about the types of obstacles that interfere with signal propagation. Such a model requires the user to first provide information about the 3D scenario where the simulation is to take place. He *et al.* [HLPR12] provide an automatic 3D scene reconstruction algorithm based on computer vision techniques to build a world model on which path loss is calculated via a ray tracing algorithm. Such an approach allows the model to rely on knowledge of concrete signal shadowing sources while simultaneously not imposing on the user the burden of constructing and feeding the algorithm such a model. Such an approach would be unpractical in the context of the challenges which this dissertation aims to tackle, given the difficulty involved in acquiring images which would provide sufficient information to allow an algorithm to construct a precise enough model of the environment where the experiment takes place.

In this section we have gone over some of the research work which has analysed the relationship between geographical position and signal strength by using a variety of radio propagation estimators, including some approaches which rely on machine learning techniques. The most relevant conclusions which can be drawn from this analysis are that there is potential in the usage of models which rely on previously acquired data to extrapolate a more generic model of signal propagation, and that the usage of machine learning to derive propagation models which provide better precision for environments and specific network node topologies is still an open research question requiring further study. The existing work on this area covers mainly the prediction of a network node's position based on the characteristics of its interactions with the wireless network. While some works have been performed which correlate signal characteristics with node positions, this review has not identified any existing literature applying machine learning techniques to the generation of custom-tailored path loss models based on traces from past experiments.

Chapter 3

Machine Learning Approach for Path Loss Estimation

In this chapter the methodology and approach taken to the machine learning problem is described. A deeper look into the domain of the problem, the dataset features and other conditioning factors is given. Then, an overview of the learning approach applied is provided, focusing on the preprocessing of data, the different approaches to modelling the target variables, and the learning algorithms chosen as well as their parametrisation. Furthermore, an overview of the target variables chosen and the error metrics from theoretical models used as a benchmark are described. Finally, the usage of the trained models for prediction of new values is described.

For the scope of this dissertation the problem we are trying to model is the unidirectional signal strength as perceived at the receiver node from a given stationary node designated as a transmitting station. Thus, in this chapter, we aim to train a model capable of outputting predictions of Received Signal Strength Indication (*RSSI*), measured in dBm, when provided with the position information of a target node.

3.1 Dataset Overview

The data to be applied for the purpose of training and evaluating machine learning models was obtained from the logs of a pre-existing real-world network experiment, considering an emerging scenario, carried out in Huelva in 2017. The experiment was performed using an Autonomous Unmanned Vehicle (*UAV*) performing a flight trajectory and equipped with an antenna, which operated a wireless point-to-point link to a base station. Communication was established between both nodes based on a TCP data flow, and logs of meta information were recorded concerning each packet, containing a set of data-points including the type of packet, the direction of the data flow to which it belonged and the RSSI at the time of reception of each packet. The UAV is additionally equipped with a GPS antenna and with accelerometers, resulting in a separate

Figure 3.1: UAV used to capture the dataset during a test flight.

set of records containing an information about the position of the UAV and the angle at regular fixed intervals for the duration of the experiment. Thus, two different sensors, one for network information, and another for positional information, are present on-board the UAV, each resulting in a separate dataset. An illustration of the UAV is given in Figure 3.1.

As previously explained, the available data is split in two datasets, one containing information about the network communication between devices, and the other containing information about the position and angle of the UAV. Both datasets are a series of timestamped records with the associated state of the experiment at the given time.

The dataset regarding the network information consists of a tabulation separated set of lines, each consisting of a record of a packet which has been sent or received at the network interface in the UAV. Each record on this dataset is referenced by a timestamp given by the local calendar year, month, day, hour, minute, second and millisecond. The available data for each packet consists of:

1. **Type of packet** - Either an Acknowledge packet marked by the ACK string, or a data packet marked by the QDATA string.

2. **Source** - The MAC address of the source network interface.

3. **Destination** - The MAC address of the target network interface.

4. **RSSI** - The Received Signal Strength Indication in dBm, if captured at the destination interface, 0 otherwise. It is important to note that the precision of the measurement is to the unit, so this value is discrete, not continuous.

The dataset regarding the UAV position and telemetry consists of a series of comma separated values. Each line of this dataset consists of a sample obtained from the GPS module on-board

the UAV. In addition to the standard longitude, altitude and latitude features it also contains the measurements of the angle of the drone in terms of roll, pitch and yaw. The set of samples are annotated with a timestamp called *mtime*, indicating the amount of seconds elapsed between the start of the experiment and the recording of the GPS datum in question.

3.2 Data Preparation

In order to be able to apply the raw data as described in the previous section to learning a model for the wireless signal strength it is necessary to perform a preprocessing step to wrangle the data into a suitable shape for training, as well as to clean it of invalid or missing values. In very large datasets or unbalanced dataset it may also be necessary to perform sampling in order to reduce the learning step to a computationally tractable problem.

3.2.1 Data Transformation and Cleaning

This section provides an analysis of the data transformation and cleaning operations which were performed to eliminate spurious or incorrect information in each of the two datasets discussed in Section 3.1.

The network dataset contains a lot of irrelevant information for the problem at hand. The useful features which were extracted are the timestamp and the RSSI value. The timestamp is converted into elapsed seconds since the beginning of the experiment by fixing a point at which measurements from both sensors have stabilised. Subsequently the data-points which concern transmissions from the UAV to the Base Station were discarded, as no RSSI data is provided for the transmitted packets.

The position and telemetry dataset also contain a number of features with no relevance. From this dataset we extract the measures of longitude, latitude, and altitude, together with the measurements of the roll, pitch, and yaw of the UAV. Additionally the timestamp was aligned with the timestamp of the network dataset by referencing the GPS time. This does not eliminate completely the chance of a clock drift, as time elapsed measurements were taken by different clocks. This clock problem could be solved, for instance, by frequently synchronising the system clock with the GPS clock. Nevertheless, this problem is out of the scope of this dissertation and will not be addressed.

3.2.2 Timestamp Interpolation

In analysing and treating the data available, we are faced with the problem that our features are gathered from the output of two different sensors. As explained in Section 3.1 two different sensors are used for gathering network information and positional information. These sensors exhibit different sampling behaviour. In particular the GPS sensor captures position information at a rate of approximately 120 Hz, while the network interface intercepts packet data exactly once per packet received. The consequence of this behaviour is that network datapoint sampling varies

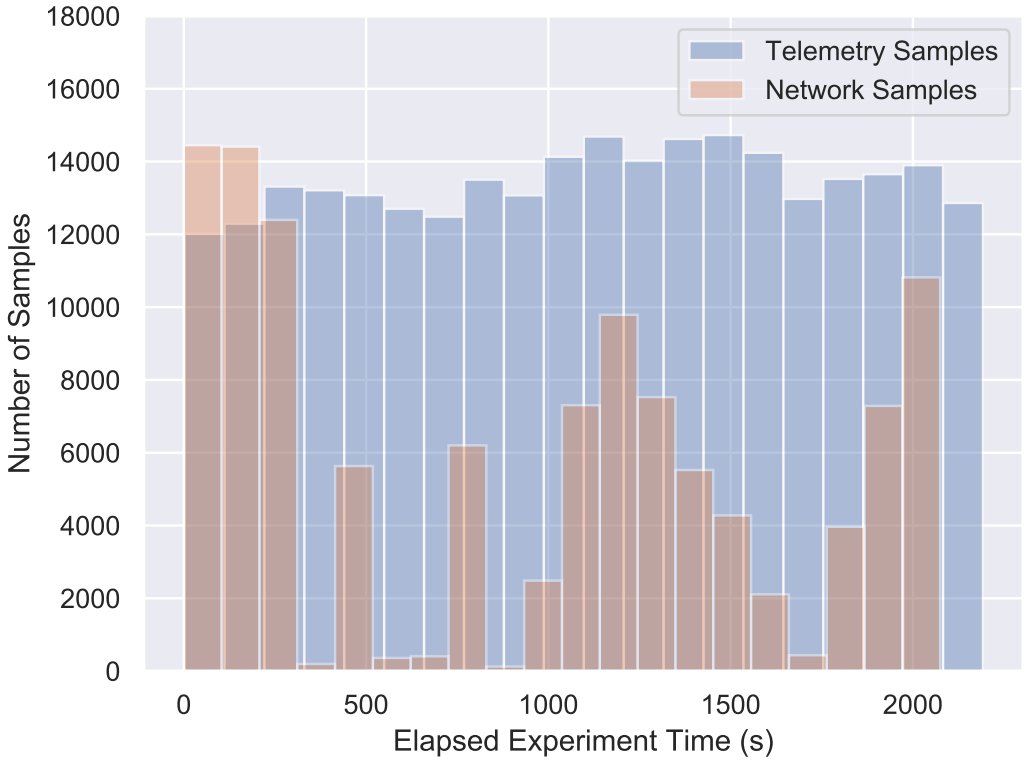

Figure 3.2: Histogram of the samples collected by time, according sensor.

significantly with time, exhibiting periods of high activity and periods where no packet is received and thus no data is available. This difference of sampling frequency between telemetry (hereinafter also called positional information) and network samples is illustrated by Figure 3.2.

To resolve this issue there is a need to align the two disjoint datasets into a single dataset [MPP+10]. This unified dataset should contain, for each row, a value consisting of the union of the set of features available in the positional and network datasets. Since there are very reduced chances that a network sample was obtained at the same time as a positional sample, the process of aligning the two datasets is forcefully a lossy one, where the data-points in one of the datasets are interpolated in order to estimate their value at a given point in time where there exists a sample for the other dataset.

As we can observe in Figure 3.2 there are significantly more positional information samples than network samples. Furthermore the distribution of the positional samples are much more homogeneous when compared to the network samples, where significant periods of very little to no information are observed, followed by periods of increased activity. Furthermore, the information contained in the positional dataset is much smoother than the information contained in the network dataset, i.e., it is less prone to sudden changes in velocity. From a physical point of view this is due to the largely continuous nature of the movement the UAV describes along its trajectory.

Regarding the signal strength at the receiver, it can be modelled as a function of both slow fading or shadowing components, which vary slowly, and fast fading components, both constructive and destructive, which may cause rapid variation of the signal strength. Finally, it is important to notice that the network dataset constitutes the target variable of the modelling problem being tackled, – i.e., the Signal Strength –, while the positional information will merely provide a set of features which will be used to model the problem.

Given the considerations laid out above, it was decided to interpolate the dataset containing positional data-points in order to align it to the network dataset. Consequently, for each packet captured, the position at which the UAV was located and the other positional information will be interpolated from the respective dataset. This process was performed by retrieving, for each network datapoint, the two corresponding positional datapoint whose timestamp were immediately greater and lower compared with its own. The final values for the positional features were then calculated by computing the weighted average by timestamp value of their individual values. This approach preserves the integrity of the signal strength information, but at the same time leads to a heavily unbalanced dataset, with many periods of little to no data-points being available followed by periods of significant sample volume.

3.2.3 Sampling

The unbalanced nature of the aligned dataset means for some positions of the UAV we have a lot of samples providing information about signal strength at that particular point, while other positions have significantly less samples. This overrepresentation of some of the positions and sub-representation of others leads to excessive computational load in processing all the samples.

To solve this problem the dataset was binned by timestamp, with each bin corresponding to samples obtained in a period of one second. Each bin is converted to a single datapoint by finding the mode of the RSSI for each bin and selecting randomly from the set of data-points whose RSSI value equals the mode. The resulting dataset contains one sample for each second of the experiment, except for time periods on which no packets were received, in which case there are no samples.

3.2.4 Normalisation

The features which will be used in the training of the machine learning model have values whose magnitudes are very different, depending on the domain. For example, the altitude feature is represented in meters, and has a maximum magnitude of around 500 meters. Latitude and longitude on the other hand, has a value amplitude of only a few thousandths, due to being represented in degrees according to the standard coordinate system.

Some regression algorithms perform badly when subject to large discrepancies in the scale of the input features. Support Vector Machines, for example, are notoriously reliant on standardisation of the data-points being used for training [Kum12]. Others, such as Decision Trees, are

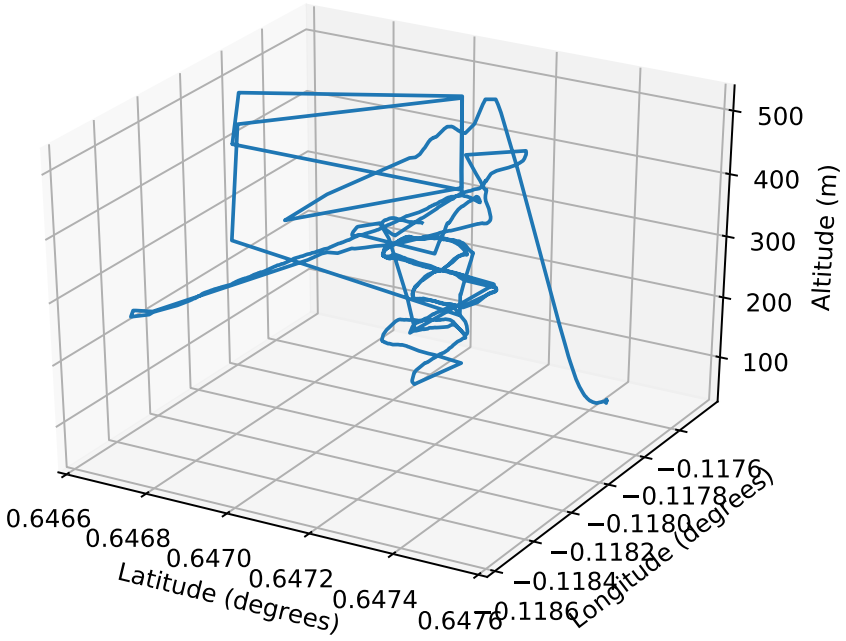

Figure 3.3: Trajectory of the UAV during the experiment, altitude not to scale.

independent on invariant in terms of individual transformations to the value of the features, and as such are not sensitive to data which has not been normalised.

For this dissertation the decision was made to normalize features using the Min-Max normalisation procedure. This means that for each feature in the input set, the minimum and maximum value were computed. Subsequently each datapoint was rescaled so that the maximum value of the feature is equal to 1 and the minimum value equal to 0. The formula applied for the transformation is given by Equation 3.1.

$$value_i = \frac{f_i - f_{min}}{f_{max} - f_{min}} \tag{3.1}$$

3.3 Final Dataset

In this section we present the dataset which resulted from the data transformation and cleaning process described in Section 3.2. The resulting dataset contains 7 features and a timestamp.

The timestamp is a useful metric to visualize how the UAV's position changed during the experiment, and was useful to align the two datasets. However, this problem cannot be modelled as a time series problem. While signal strength and path loss changes over time, it does not change

	Mean	St. Dev.	Minimum	25%	50%	75%	Maximum
lat	0.65	0.00	0.65	0.65	0.65	0.65	0.65
lon	-0.12	0.00	-0.12	-0.12	-0.12	-0.12	-0.12
alt	311.94	146.35	1.00	225.00	295.00	480.00	536.00
roll	0.06	0.40	-3.09	-0.03	-0.00	0.09	3.13
pitch	0.06	0.07	-1.23	0.03	0.05	0.07	1.53
yaw	-0.12	1.87	-3.14	-1.73	-0.22	1.81	3.14

Table 3.1: Central tendency and dispersion characteristics for the dataset features.

as a consequence of the passing of time. As such the timestamp is not a feature which can be used to train the model. In Figure 3.3 the trajectory of the UAV for the duration of the experiment is shown.

The features we are left with are the roll, pitch and yaw of the UAV along with its geographical coordinates, namely latitude, longitude and altitude. In Table 3.1 some of the central tendency and dispersion characteristics of the features are described in more detail.

In order to gain insight into the predictive power any given feature may have, the values of each feature and the corresponding RSSI were plotted and a regression line was fitted, resulting in the series of plots available in Figure 3.4. The main conclusions are that the features which most strongly correlate with RSSI are longitude and altitude of the UAV.

3.4 Regressor Evaluation

In the previous sections of this chapter the process to build the dataset and the resulting structure was explained. In this section the usage of the dataset to evaluate models for the prediction of signal strength is described. In particular this section goes over the two different target variables and the various algorithms used to train the model, together with the relevant parameterisation choices and cross-validation results.

In order to compare algorithms and tune hyperparameters[1] cross validation is performed in conjunction with an exhaustive search on a subset of the domain of the possible values of all hyperparameters. This approach allows for the comparison of different machine learning algorithms to the dataset, and can be replicated in other scenarios to help select the best model for the scenario.

3.4.1 Target Variables

The machine learning models described in the next sections are trained to predict a value which represents the path loss according to a spatial location and angle. Two different approaches to modelling this value were taken:

[1] A parameter which is set prior to the model being trained, parametrising the training process.

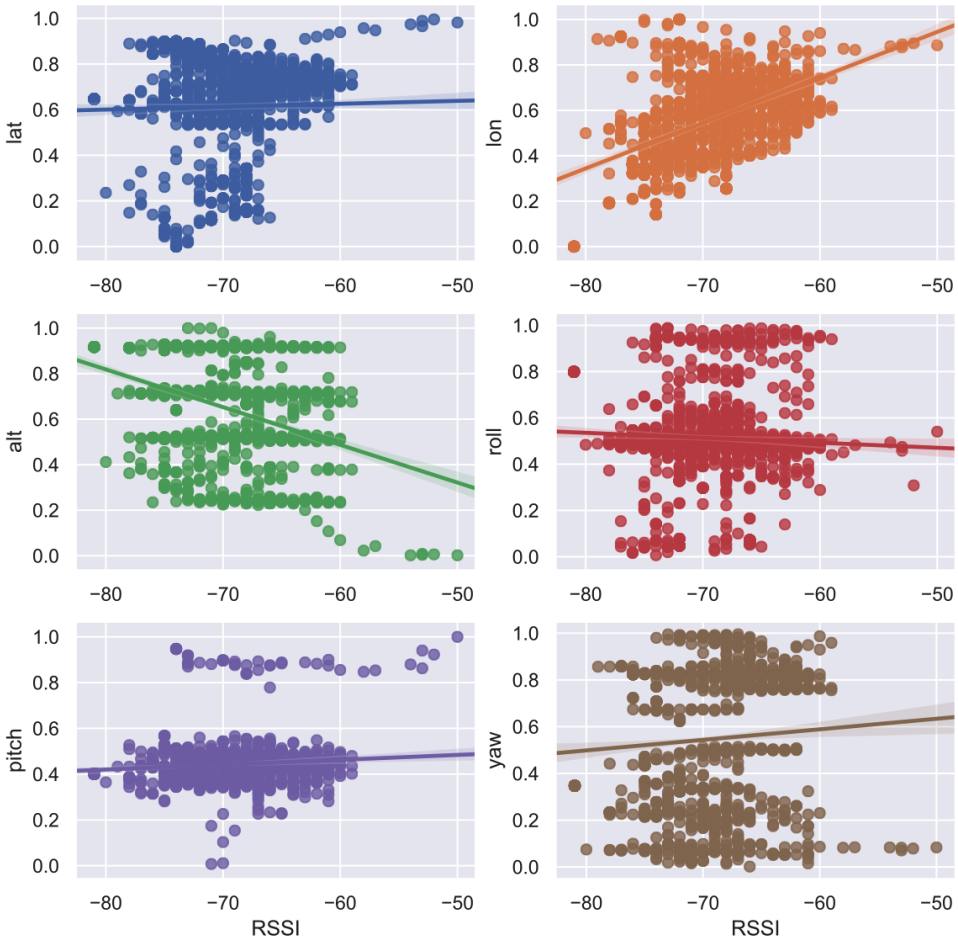

Figure 3.4: Linear regression of individual features and RSSI.

1. **Direct RSSI Prediction** - This approach trains the model to predict the value of the real RSSI observed at each data-point in the training set.

2. **Correction of Friis** - This approach trains the model to learn the difference between the RSSI predicted by the Friis model and the real RSSI observed at each data-point in the training set. The predicted value in the validation data-points is then given by adding the RSSI value given by the Friis model at that point with the prediction provided by the model.

The error metrics for both of these approaches are addressed in the following sections.

	Mean Absolute Error	**Median Absolute Error**	**Mean Squared Error**
Friis	4.56	3.64	31.42
Two-Ray	6.95	6.14	74.92
Log-Distance (γ=1.8)	9.58	9.24	105.15
Log-Distance (γ=2.2)	11.15	11.11	139.13

Table 3.2: Error metrics for the benchmark models.

3.4.2 Benchmark Metrics

For the purpose of evaluating the accuracy of the predictions of the simulated models, Table 3.2 presents the error metrics of three common path loss models, namely the Friis model, the Two-Ray model and the Log-Distance model with a γ value of 1.8 and 2.2. For the Log-Distance model the γ parameter is set based on a scenario such as a corridor where obstructions may cause the signal to be stronger in a particular direction, and on a open field scenario, respectively.

3.4.3 Methodology

In order to avoid overfitting the model to a particular division of the dataset between training and testing data, cross validation was employed. The dataset splits were generated by employing a unshuffled k-fold scheme, with 3 splits being extracted. The decision to employ a unshuffled split derives from the continuous nature of the dataset, where each datapoint is numerically close to those who precede and succeed it in the original dataset ordering. If the dataset was shuffled before being split, each division of the dataset would contain data which would be representative of the entire dataset, leading to an optimistic evaluation of the dataset, as samples taken mere moments before or after those present in the testing dataset would be present in the training data.

After the dataset has been split into partitions a given algorithm is then trained on the aggregation of all the partitions except one. The training process generically consists of feeding the chosen algorithm with a set of hyperparameters, data-points and the corresponding target variables. The outcome of this process is a trained model, as defined by a set of parameters. The partition which is left out from the training process is then used to evaluate the trained model, resulting in one or more metrics which describe the performance of the model for that particular train-test split of the data. The process is then repeated until every partition of the dataset has been used for evaluating the model. The metrics are then composed by calculating the mean value and the standard deviation for each metric.

The process of training an algorithm can be considered to be the process of setting the parameters of a model according to a training dataset. Hyperparameters, in contrast with the aforementioned parameters, are properties of the learning algorithm which control the process through which the parameters are learned. These values must be set to suit the target problem. In this dissertation the value of the hyperparameters was set by performing a grid search over a range of possible values for each hyperparameter. For each combination of values, the cross validation

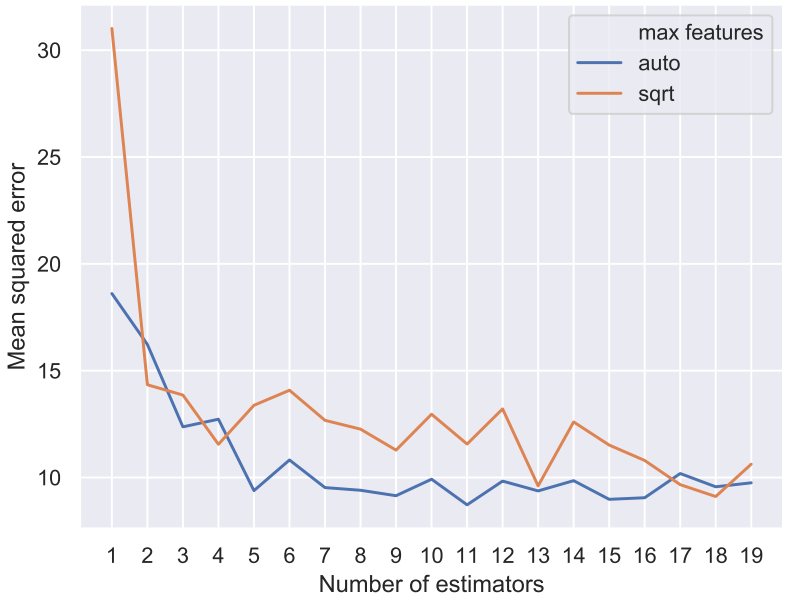

Figure 3.5: Grid search for Random Forest hyperparameters.

scheme described above is applied. The hyperparameters which yield the best results according to the employed metric are then selected.

The following sections describe the hyperparameter optimisation and the results of the evaluation of two different machine learning algorithms.

3.4.4 Random Forest

The Random Forest algorithm consists of training multiple decision trees from samples of the data, which are drawn with replacement. Each node is split according to the best split for a subset of the features being considered, instead of the entirety of the feature set. The hyperparameters that are analysed are the number of decision trees that compose the forest and the subset of features which are considered for each split. The former hyperparameter is made to vary between 1 and 20, while for the latter the two possibilities are using the entire set of features or using the square root of the cardinality of the set of features. The results of the hyperparameter search are summarized in Figure 3.5. The best combination of hyperparameters found was using 17 decision trees and considering the entire set of features.

Evaluation measures were computed using cross-validation both for the direct RSSI prediction and the correction of Friis model. The results are presented in Table 3.3. This algorithm appears to perform better when calculating a correction to the Friis model than when directly predicting RSSI values.

Direct RSSI Prediction	Split 1	Split 2	Split 3	Aggregate
Mean Error	2.70	3.09	3.58	3.12+-0.44
Median Error	2.2	-2.5	3.4	-2.7+-0.62
Squared Mean Error	7.29	9.49	12.81	14.59+-3.87
Correction of Friis				
Mean Error	2.22	2.93	3.31	2.82+-0.56
Median Error	1.38	2.88	2.90	2.39+-0.58
Squared Mean Error	7.07	12.63	15.77	11.82+-4.41

Table 3.3: Evaluation measures for the Random Forest algorithm.

3.4.5 Adaboost

The Adaboost algorithm was applied by using decision trees with a depth value of 9. The hyperparameters to set are the loss function with each to train, the learning rate to be applied and the number of estimators to train. The grid search for optimal hyperparameters was performed with the values of the learning rate ranging from 0 to 1, and the number of estimators ranging from 50 to 400. The loss function which is used to calculate the weights of the weak learners can either be linear, square or exponential. The results of the grid search are summarised in Figure 3.6. The combination of hyperparameters found to perform the best from among those searched are an exponential loss function, with a learning rate of $1,0$ and 400 estimators.

Evaluation measures were computed using cross-validation both for the direct RSSI prediction and the correction of Friis model. The results are presented in Table 3.4. As can be seen the direct RSSI prediction leads to slightly lower error in prediction when compared to the approach where a correction of the Friis model is targeted.

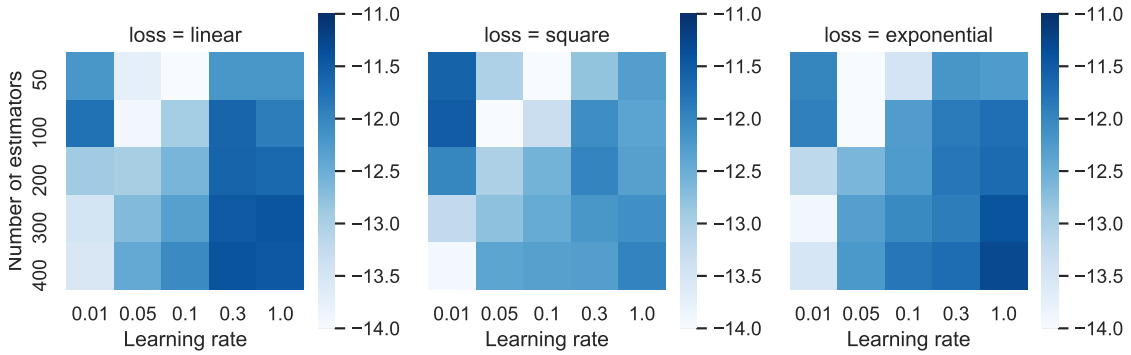

Figure 3.6: Grid search for Adaboost hyperparameters, with negative square of errors metric.

Direct RSSI Prediction	Split 1	Split 2	Split 3	Aggregate
Mean Error	2.33	2.76	2.67	2.59+-0.22
Median Error	1.44	2.07	2.31	1.94+-0.45
Squared Mean Error	11.04	11.61	10.38	11.01+-0.62
Correction of Friis				
Mean Error	2.75	2.74	3.26	2.91+-0.30
Median Error	2	3	3	2.67+-0.58
Squared Mean Error	16.85	11.31	15.23	14.46+-2.85

Table 3.4: Evaluation measures for the Adaboost algorithm.

3.4.6 Support Vector Regression

A Support Vector Machine was trained to perform regression on the dataset. The kernel choices which were included in the parameter sweeping process were the linear, radial basis function and polynomial kernels, each kernel corresponding to a different process through which the input data is transformed so that a non-linear boundary can be learned as a regression curve. The hyperparameters to select are the soft-margin parameter known as C and the gamma value. A grid search was performed with C varying between 0.001 and 10 and gamma varying from 0.001 to 1. The results of this grid search process can be seen in Figure 3.7. The best combination of parameters that could be found in the grid search was the radial basis function kernel with a soft-margin parameter of 1.0 and a gamma value of 0.1.

Evaluation measures were computed using cross-validation both for the direct RSSI prediction and the correction of Friis model. The results are presented in Table 3.4. As can be seen the direct RSSI prediction leads to slightly lower error in prediction when compared to the approach where a correction of the Friis model is targeted.

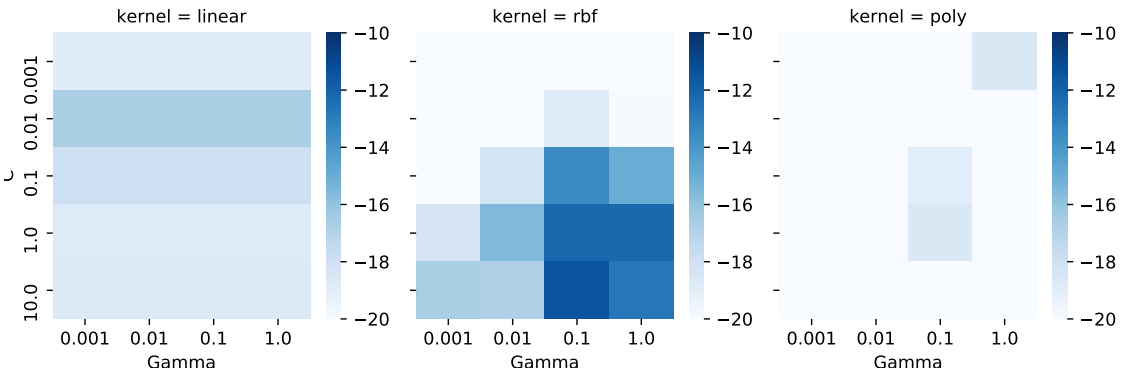

Figure 3.7: Grid search for SVM hyperparameters, with negative square of errors metric.

Direct RSSI Prediction	Split 1	Split 2	Split 3	Aggregate
Mean Error	2.35	2.43	2.78	2.52+-0.23
Median Error	1.52	2.26	2.18	1.98+-0.41
Squared Mean Error	11.63	8.86,	12.43	10.97+-1.88
Correction of Friis				
Mean Error	2.43	2.51	3.14	2.69+-0.39
Median Error	1.69	2.60	2.50	2.26+0.50
Squared Mean Error	13.92	9.19	16.79	13.30+-3.83

Table 3.5: Evaluation measures for the SVM algorithm.

3.4.7 Result Analysis

Through comparing the benchmarks extracted by the application of the benchmark model with the results obtained by applying the three different algorithms and two different target variables some conclusions may be drawn:

1. While some approaches exhibit higher accuracies than others, the difference between the best and the worst machine learning model is 19.2%.

2. The performance impact of choosing either target variable is minimal, and the approach which performs better varies according to the selected algorithm.

3. The performance improvement achieved by the machine learning approach when comparing to the most accurate benchmark result is 44.7%.

Figure 3.8 provides a visual comparison of the results obtained and the benchmarks.

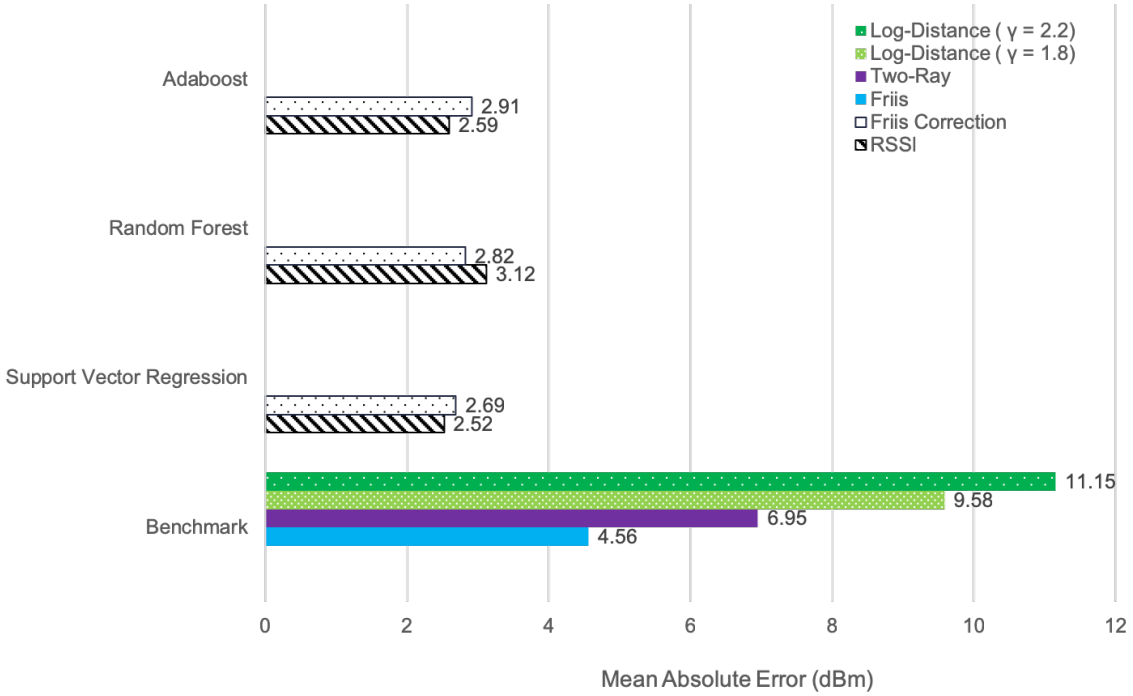

Figure 3.8: Summary of the accuracy of the different approaches when compared to the benchmarks.

Chapter 4

Proposed Software Platform

In this chapter the design of the proposed software platform to support the Machine Learning approach for path loss estimation is presented. The designed software platform is decomposed into its constituent parts and the rationale for the architectural choices involved in the logical components is given. While in Chapter 3 the application and evaluation of machine learning techniques to the goals of the dissertation are described, in this chapter the supporting infrastructure allowing for the application of said techniques is detailed, together with the technologies which were employed.

4.1 System Components

The developed system is structured in multiple components with weak dependencies on each other. There are multiple advantages, from the perspective of a software engineer, from such a decomposition into loosely coupled components. It allows each system in the component to evolve without impacting other components in the system. Furthermore it allows the developers to swap out one component for the other without needing to perform extensive changes to the architecture of the system.

The following components comprise the developed solution:

- **Data repository** - Database for the storage of datasets and models, giving persistence support to other modules of the system.
- **Machine Learning module** - Module for training and evaluating new models based on datasets and for generating new datasets based on models.
- **Web Service** - A high level interface for requesting operations to the system, abstracting the functioning of the system to the end user clients.

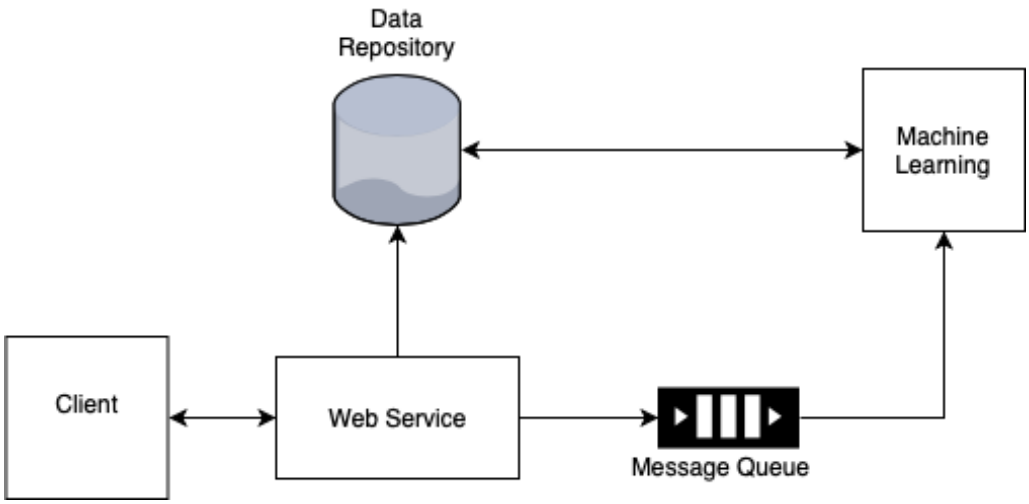

Figure 4.1: High-level overview of the system components.

- **Message Broker Service** - A message broker allows for non-blocking message passing between components, allowing for long running operations without keeping the system unresponsive.
- **Client** - A command line interface for interacting with the system.

When a user uploads a new dataset into the system, when a new model is trained from an existing dataset, or when a new dataset is derived from a previously trained model, the corresponding data is stored in the data repository for future usage. The system's capabilities are activated via a web service interface. It handles user requests by containing the business logic which parses and stores datasets in the data repository, verifies user credentials for access to protected resources and actions and triggers machine learning jobs. The message broker component is useful due to the high computational requirements associated with training machine learning models over large datasets, keeping the web service responsive during said operations.

A visual representation of the system's high level components and their relationships is given in Figure 4.1.

4.2 Data Repository

As previously described, the data repository serves as a central point for persisting data such as datasets and models, which are required for future computations or access.

4.2.1 Requirements

This dissertation places no heavy burden in terms of performance on the data storage solution chosen, since any given operation to be performed by the system interacts sporadically with the database, usually once to retrieve and again to store a dataset or model. We do not require high

availability, as data storage outages are not a catastrophic failure in terms of the capability of the system to deliver its benefits. Furthermore, the schema of the data to be stored is not dynamic and not expected to change often or radically. On the other hand it is important to note that the number of data-points in any given dataset may be large, and it is required that the data storage system used is able to quickly retrieve a large quantity of entries in an efficient manner, while filtering according to the dataset which is being operated on. Additionally, the system should be able to apply a number of formatting constraints on the datasets it stores. Having a fixed schema enforced in the data storage solution eases the development of the other components of the system, in particular the machine learning component, and helps users avoid incorrect modelling of data based on wrongly formatted or corrupt input datasets.

The database solution chosen for this dissertation should thus consist of a solution which can gracefully handle a large volume of records while simultaneously achieving good performance while querying according to some particular parameters and ensuring conformance to a data schema.

4.2.2 Data Schema

In order to guarantee correct treatment of the datasets, it is important to establish the kinds of data the system will operate on. These can be broadly divided in as follows: data-points which will be used in order to train new models and represent synthetic models derived through the machine learning process, models which represent the outcome of the machine learning process and can be used to generate new datasets; and auxiliary data which is not directly used for the core purpose of the system but provides information which serves to structure interactions with the system. The newly generated datasets previously mentioned correspond to representations of the custom tailored path loss models derived from the machine learning process.

Three data types are identified for the purpose of the data schema:

- **Data point** - a sample of a value in the dataset representing a node position and the wireless signal strength.
- **Dataset** - a grouping of data points to which some auxiliary meta information is added, including whether the dataset was harvested from real data or generated from a model.
- **Model** - a representation of a trained machine learning model which can be used to generate new datasets or for validation. It additionally contains meta-information regarding the type of model and the dataset from which it was derived.
- **User** - a representation of a user which interacts with the system. It is used for authorisation purposes and contains information such as a username and password which can be used to authenticate in the platform.

A visual representation of the data schema is provided in Figure 4.2.

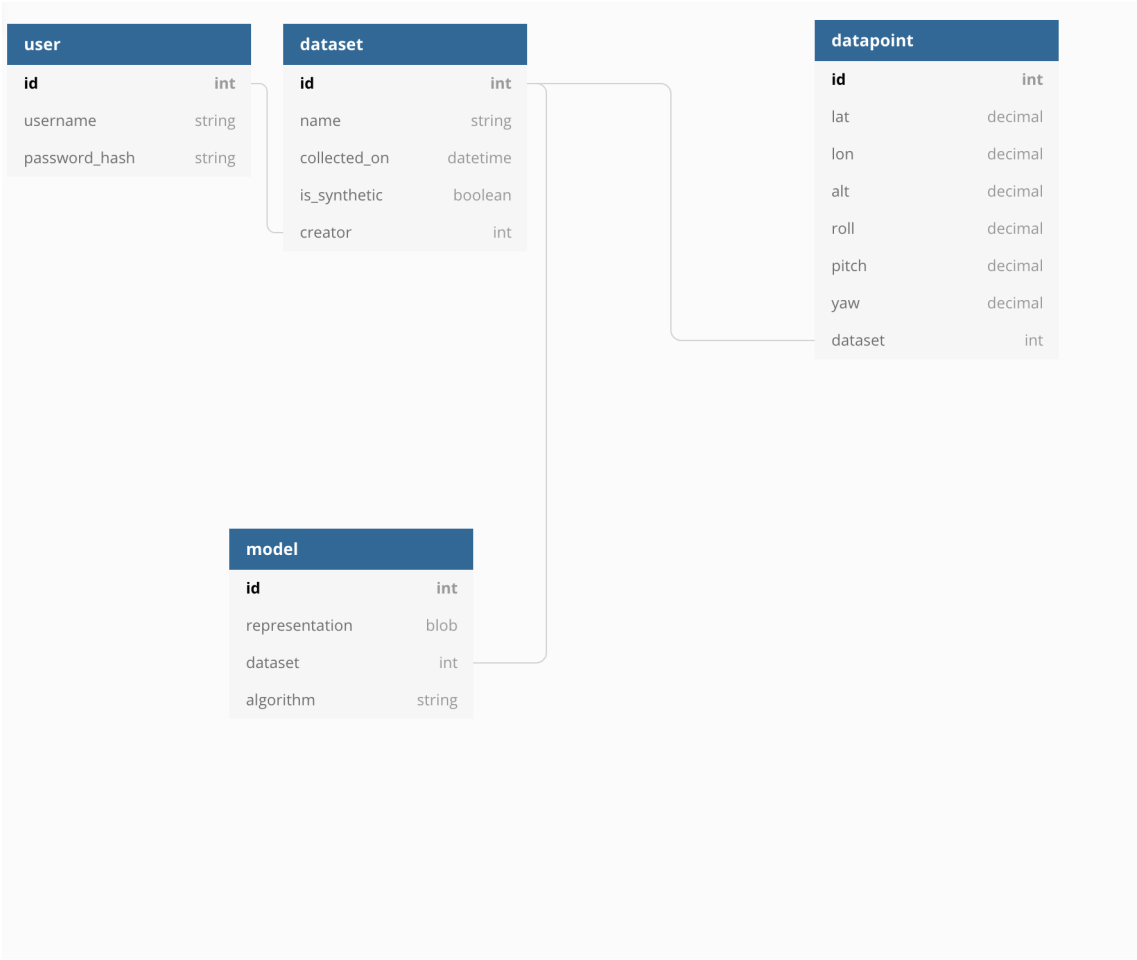

Figure 4.2: Database schema.

4.2.3 Database Choice

Database solutions can be divided into two large groups: relational databases and NoSQL databases, standing for "Not only SQL". Relational databases enforce a set of characteristics represented by the acronym ACID: Atomicity, Consistency, Isolation and Durability [GR92]. It respectively means that:

- Each transaction to the database system must either fully succeed or fail entirely, with no partial change to the state of the database as a consequence of the transaction failing midways.
- No transaction can cause the database to evolve from a valid state to an inconsistent state where database invariants or data constraints are violated.
- The concurrent execution of any set of transactions will not yield a different database state than some sequential ordering of said transactions.
- Any successful transaction is durably stored, which is to say it is somehow persisted and will remain even in the occurrence of a system failure.

NoSQL technologies relax some of the properties described above in order to relieve performance and replication constraints which are caused by trying to implement them fully, while simultaneously boosting performance and availability by using data replication. They mostly adhere to the properties described by the BASE acronym: Basically Available, Soft State, Eventual consistency [Cha15]. Such systems are aimed at providing high availability while relaxing ACID guarantees such that the state of the database may change over time.

As we show in Section 4.2.1 we do not require very high performance or availability on the database, which would make the choice of NoSQL technologies attractive. On the other hand, regular relational databases provide helpful guarantees and are at this point more widely understood and integrated in other technologies. As such, we decided to use a relational database.

The relational database chosen for the implementation of the work was the PostgreSQL Relational Database Management System. It is an open-source relational database which has solid integration with web frameworks and provides sufficient functionality to implement the simple use cases this dissertation requires of the chosen database solution.

Operations on the PostgreSQL relational database are usually performed using the Standard Query Language (*SQL*), a declarative language for describing database transactions. While powerful and flexible, the burden of writing SQL queries for every operation on the system can become burdensome and error prone, particularly when performing simple Create, Read, Update and Delete operations (*CRUD*). To ease database integration in applications a common approach is to use a so called Object Relational Mapper (*ORM*). The ORM consists of a thin layer mapping the relational concepts which are used in the database system into a object-oriented domain for ease of integration in applicational codebases. In the case of this work we leverage the ORM integrated in the chosen Web Service framework to simplify interactions with the database.

The requirement for performant datapoint queries in the database is satisfied by the usage of indices on the foreign key of the datapoint data schema. This feature, supported by PostgreSQL, speeds up queries which apply a filter over tables containing many rows, such as the datapoint table. This is very helpful for the common use case of retrieving all data-points associated with a particular dataset.

4.3 Web Service

The Web Service serves as a single point of contact for outside interaction with the system. Any software client used to provide access to the system can operate by performing requests to this service.

4.3.1 Protocols and Architectural Styles

The main protocol for communication in web services is the Hyper Text Transfer Protocol (*HTTP*) [FGM+99]. This protocol operates under a request-response model, where one of the communicating entities acts as server of which requests are made and the other entity acts as a client which retrieves information and requests operations from the server. HTTP provides support for network optimisations where intermediate machines in the communication may inspect requests and replies, and use the contained data to cache information.

HTTP requests are performed by the client to a specific Uniform Resource Locator (*URL*). In addition to identifying the server to which the request is made, a URL specifies a resource in the server which is to be operated upon. An HTTP request specifies a method to be invoked on the URL to which it is addressed, and may contain additional data in a series of headers, which can be used for negotiating applicational details such as content encoding and user authentication. Some methods also allow for the use of a request body, which can contain details of the request which are not suitable for specification in the URL. As a result of a client request the HTTP server produces a reply. This reply includes a status code indicating the outcome of the operation and an optional response body, together with meta information provided by headers.

While HTTP provides a series of primitives and actions which may be used to implement APIs to enable communication over the World Wide Web (*WWW*), it does not contain strict indications on the structuring of the programs which implement the protocol. Several architectural styles have been conceived to provide guidelines for the definition of APIs, including the HTTP methods to be used, how endpoints should be defined and the precise semantics of client requests and the status codes in server responses. The two biggest families of architectures are 1) action-oriented architectures, structured around a remote procedure calls paradigm where the client specifies the behaviour and logic he intends the server to invoke as a response to his request, and 2) resource-oriented architectures, where the client identifies or provides the server with resources he wishes to access, create, update or delete and the server is responsible for executing the logic necessary to carry out the request or reject it. Under the latter architectures URLs are usually used to represent the type and identity of the resource to be operated, with HTTP methods being used to provide

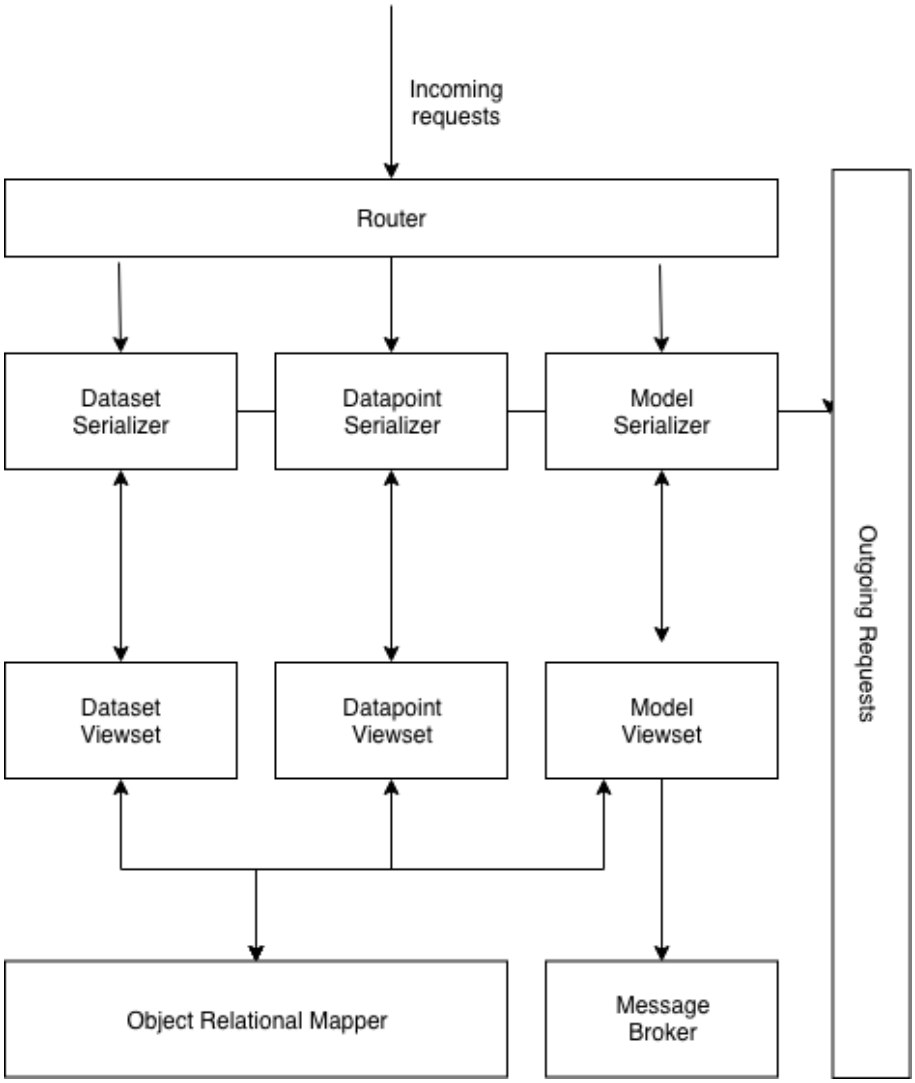

Figure 4.3: Overview of the architecture of the web service.

the semantic meaning of the request. This approach makes more efficient usage of the caching mechanisms enabled by HTTP.

The architectural style of choice for this work is the Representational State Transfer architecture (*REST*), which is a resource-oriented architecture first described in 2000 [FT00]. This architecture establishes a clear separation of concerns between client code and server code, by abstracting server actions as resources, and by abstracting a resource from its encoding, enabling the loose coupling of client and server logic [ASJH11].

4.3.2 Architecture and Technology Selection

In order to treat and respond to external calls to the application in a structured fashion the web service implements a series of components to extract resource identities and request metadata

from the incoming requests and perform the logic and data operations required to provide a valid response.

To aid in the process of developing this service the Django framework was used [Fou18]. This framework provides a Model View Controller architectural pattern for implementing a web application, dividing the components of the aforementioned application into those three categories.

The Model components are responsible for communicating with the Data Store in order to provide the application with a means to persist the resources it operates on. Django facilitates the implementation of such components by providing an ORM enabling seamless access to data stored in a remote database in a object-oriented fashion. Further details on the value of such a solution were explored in Section 4.2.3.

The View components, in turn, are responsible for formatting data which will be returned from the service to the client using a mutually supported encoding. The REST architectural style specifies that the data types and their encoding should be separated, and clients should be able to request, through the usage of a HTTP header, the encodings they accept in the responses. Django facilitates the process of implementing View components which provide this flexibility by providing an API for the definition of serialisers which can be used to describe the process in which the data should be encoded in a format-agnostic fashion. These serialisers are responsible for decoding resource representations provided by the client into the web services internal models, and for encoding said models into user-acceptable representations.

The Controller components, represented in Django by the concept of viewsets, are responsible for implementing the logic which drives transformations to the model and guarantees the compliance with constraints such as authorisation verification. In this dissertation, Django viewsets are used to 1) retrieve the correct datasets and machine learning models from the Model component, and 2) enqueue requests to the message broker when machine learning functionality is requested.

Finally, as an auxiliary component, Django provides routers, which are used to parse the URL of a request and determine which Controller should be responsible for handling it. Routers are used to define the endpoints exposed by the system to the exterior.

An illustration of a conceptual overview of the architecture of the system is given by Figure 4.3.

4.3.3 Functionality

The two previous sections provided a high level overview of the protocols and architectural choices which respectively supported and guided the development of the web service. In this section, a functional description of the module is provided.

The developed web service acts as a co-ordinator for the execution of machine learning tasks on the corresponding module, as well as providing an abstraction for accessing the datasets. The machine learning module will use those web services to train models, store new datasets and trained models, and synthesise new datasets for posterior application in trace-based simulation. In accordance to the architectural guidelines explained in the previous section, this functionality is exposed through a series of HTTP endpoints, which provide the entry point of requests into

URL schema	Parameters	Possible Status Codes	Allowed Methods
/dataset	name(filter)	200, 400	GET, POST
/dataset/{id}	N/A	200, 400, 404	GET, PUT, DELETE
/datapoint/	datasetName(filter)	200, 400	GET, POST
/datapoint/{id}	N/A	200, 400, 404	GET, PUT, DELETE
/model	N/A	200, 400	GET, POST
/model/{id}	N/A	200, 400, 404	GET, PUT

Table 4.1: Endpoints on the web service.

the web service. In Table 4.1 a summary of the endpoints made available by the web service is provided.

New datasets are uploaded into the system by sending a HTTP request with a POST method to the dataset endpoint, supplying the relevant metadata as show in the data schemas (Figure 4.2). On successful creation of the dataset the internal representation is returned to the client. The datapoints which comprise the dataset may then be created in the system either individually or in bulk using the POST method and the corresponding endpoint. In order to associate the individual datapoints with the overarching dataset to which they belong, an additional ID field must be associated with each one of them.

Machine learning models are trained by performing a POST request to the model endpoint, containing the dataset on which training is to be performed together with the desired algorithm for training. The system registers the request into the Message Broker and persists the untrained model into the Data Store. When training is complete, the Machine Learning module will perform a PUT operation into the specific model via the same endpoint, storing the representation of the model and other relevant metadata.

Data present in the system may be extracted by performing GET requests to the relevant endpoints. When no identifier is provided, all entities are returned, whereas when an identifier is present in the request only the relevant entity representation is returned, in accordance to standard REST architecture. Some entities can also be removed by performing a DELETE request to the relevant endpoint.

4.4 Message Broker

A message broker is a component which allows for asynchronous machine-to-machine communication by using a messaging paradigm where applications may communicate with each other by sending and retrieving messages to and from the broker. In this dissertation a Message Broker component is used to communicate between the web service and the machine learning module. This allows the former to forward requests for long-running training computations to be performed by the latter.

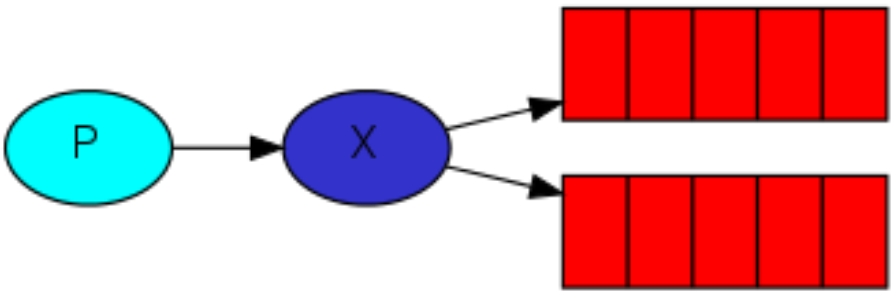

Figure 4.4: Publisher-subscriber system with multiple subscribers [rab18].

4.4.1 Requirements

A variety of message broking and delivery systems exist which support a varying set of features. For this work, the role the message broker will take is that of a simple non-blocking task queue, and thus the following requirements can be identified:

1. Support for a publisher-consumer message exchange, where the web service will serve as a publisher and the Machine Learning module as the consumer, allowing for multiple instances of the Machine Learning module to concurrently consume messages and perform tasks. This is illustrated by Figure 4.4.

2. Asynchronous delivery, allowing for messages to be queued while no Machine Learning module computation capability is available. This enables non-blocking delivery of messages from the web server, which is critical for performance reasons and to avoid timeouts of the underlying TCP connections of the HTTP server.

There is no expectation that the system will be deployed in a very high activity scenario, and there is no mission critical need for data loss prevention, and as such there is no need for some advanced features such as multi-broker architectures, clustering capabilities, or data replication which are available in some message broker technologies.

4.4.2 Technology Choice

The message broker selected for the work is the RabbitMQ message broker. Given the relatively straightforward and light-weight requirements we have defined for this application, almost all message brokers in wide use are valid choices for the stated purpose. The choice of message broker thus relied on availability of documentation as well as the suite of tools which integrate well with the web service technologies and the Python programming language used in the Machine Learning module. In particular the Dramatiq [dra17] allows for easily queueing operation requests and executing them inside Python programs, and has bindings for the Django web service framework.

4.5 Machine Learning Module

The machine learning module is the central piece of the system, containing the logic and the dependencies which allow for the training of models predicting wireless signal strength by extrapolating based on real-world experiments. In this section an overview of the components of this module is provided, alongside an explanation of some of the supported interactions.

4.5.1 Components

The machine learning module has three main sub-functionalities which are required for the derivation of models and their accuracy. In particular the following components can be identified:

- A subscriber component which will obtain messages from the message broker and orchestrate the execution of the command they contain.
- A data retrieval component, responsible for handling communication with the dataset in order to fetch datasets for training and persisting trained models.
- A training component, responsible for using the provided dataset to derive a machine learning regression model which is capable of performing predictions of signal strengths for fictitious positions.
- A predictive component, which receives a set of unlabelled node positions and angles, and a previously trained model, and uses them to emit predictions in the form of a new, synthetic dataset.
- A data preprocessor, responsible, among other features, for the removal of data-points containing incomplete information and for, when required, performing normalisation on the values of the data-points which will be used as an input for model training.

4.5.2 Technology Choices

Machine learning techniques and implementations nowadays consist of a combination of sophisticated mathematic models with highly optimised computer programs allowing for increasingly more accurate models, while at the same time minimising training computational requirements, allowing for the timely processing of large datasets.

In the context of this work, taking into account that one of the main goals is to evaluate the concept of using machine learning regression for extrapolating new datasets, it was considered that the implementation of the machine learning models itself is out of scope. The objective is to use existing machine learning frameworks which come with a large set of training algorithms and auxiliary tooling such 1) as model validation functions targeting a wide range of metrics, 2) unified classifier representations allowing for standardised persistence and retrieval of trained classifiers for posterior data prediction, and 3) data normalisation tools. These implementation are also much

more thoroughly tested than any new machine learning framework that could be implemented in the time frame of this dissertation.

As such, the decision was made to use the scikit-learn Python framework for machine learning [PVG+11]. This framework is a good fit for our work, because it integrates a large variety of regression algorithms, allowing for comparisons between different types of classifiers. Furthermore it allows for easy integration with the joblib Python library for serialising data structures, allowing classifiers to be stored in binary format for persistence in the database system. It also has good data normalisation facilities which are vital for the accuracy of some of the models tested, and allows for the normalizer to be persisted. Finally, it has good result visualisation capabilities by integrating with the matplotlib library, which allowed for the extraction of plots showing the results obtained by the system on a validation dataset.

4.5.3 Functional Description

As previously described the entry point into the Machine Learning module is the Message Broker component. Messages published in the chosen technology for the Machine Broker are encoded in the Advanced Message Queuing Protocol (*AMQP*) protocol format. In order to easily use the RabbitMQ message broker to transmit orders to execute a function and at the same time pass in the relevant parameters, the Dramatiq [dra17] library is used. This library encodes and decodes function invocations and parameters into the AMQP protocol. As such, from the perspective of the Machine Learning module, a remote invocation is perceived as a simple normal invocation of a function. While this library also possesses the capability to automatically encode the invoked function's return value and enqueue it in the message broker, this functionality is not used as in our architecture messages flow unidirectionally from the web service into the Machine Learning module.

After the request has been received and parsed as described above, some validation must be performed on the invocation parameters. In particular it must be verified that one of the two supported machine learning operations has been called, and that the parameters passed to the invocation are correct. Each of the supported operations is identified by a string and defines a set of mandatory parameters. The supported operations are:

- **Train** - the Machine Learning module is expected to derive a model from a dataset according to some algorithm. The mandatory parameters are the dataset identifier to be operated on as well as the algorithm to be used for training, alongside the relevant algorithm hyper-parameters.

- **Predict** - the Machine Learning module is expected to take an existing model which it has previously trained, and a set of points for which signal strength predictions are requested, using them to generate a new dataset which results from applying the model to the aforementioned set of points. The mandatory parameters are the set of points for which predictions are expected and the model to be applied.

After validation has been performed successfully the request flows into the data retrieval component. In the case of a train request this component is responsible for contacting the web service via the REST API to request a representation of the dataset. The requested algorithm is then used to fit a model to the specified dataset. A representation of this model is then sent to the REST API to be persisted in the database. In the case of a prediction request this component instead asks the REST API for the binary representation of the previously trained model. The model is then used to predict the value of the RSSI for a given list of positions, and the results are persisted through the REST API as a new dataset.

Chapter 5

Conclusions

This final chapter presents an overview of the work developed, recalls the main contributions of our work, and enumerates the possible improvements and features which may be the subject of future work.

5.1 Overview of the Work Developed

Chapter 2 presented a literature review on the state of the art relating to path loss models and their application in simulation, on machine learning, and some of the algorithms used to train models in the context of machine learning. It also presented an analysis on the work related to the modelling of the signal strength of wireless transmissions, with a focus on the reverse problem of determining the position of the node based on the signal strength at the receiver, combined with other network information. Continuing the related work analysis, this chapter also presented platforms which enable the usage of network traces obtained in real-world experiments to feed simulation models obtaining closer to real results using simulators such as ns-3. The main conclusions from this review were that there are some previous works using machine learning for modelling the behaviour of wireless signals, but they tend to be focused on predicting node positions as they correlate to signal strength, instead of on modelling the characteristics of a network signal given a previous experiment. Additionally, the existence of various algorithms for applying machine learning to regression problems were described, to help the process of selecting and evaluating algorithms in later chapters.

In Chapter 3 an overview of the methodology taken to the machine learning problem in question was provided. The problem was explained in more detail, the format of the experimental datasets used in conceiving the approach was summarised, the approach taken to data cleaning and preparation was described, and the process of selecting model hyperparameters and evaluating their behaviour was laid out. Furthermore, the results of the experiments with two models was provided. This chapter highlighted some of the difficulties presented by the characteristics of the

datasets used for this dissertation. An enumeration of some of the techniques used to shape the data into useful features on which machine learning can be applied, and an evaluation of the accuracy of these models were also presented. The machine learning approach proposed in this chapter allows for running more realistic simulations based on real scenarios, when compared to the usage of classic path loss models, while not requiring the real scenario to be entirely reproduced, as is the case in trace-based models. Furthermore, the results generated by the new models are formatted as traces of new, fictitious, trajectories, allowing for compatibility with current trace-based simulation implementations.

Chapter 4 described the framework for the application of the method presented and evaluated in Chapter 3. The design of the system was explained by decomposing it into its constituent components, explaining the functional behaviour of each and the role it occupies in the complete system. Additionally, for each component the technologies used to implement it were described, and a rationale for the decision was given. This chapter also highlighted the main technological choices taken both in the machine learning component and on its supporting platform. The proposed system allows the application of the proposed machine learning approach detailed in Chapter 3 to be exposed as a web service, enabling its use without having to run the computationally expensive training processes locally or having to manually configure the training environment.

5.2 Main Contributions

The main contributions resulting from this dissertation can be summarised as follows:

1. A machine learning approach to generate custom-tailored path loss models, reflecting the specific characteristics of a given environment and mobile node, based on traces from real past experiments.

2. A software platform to store traces, evaluate and select the best machine learning model to approximate the real world conditions described by the traces, and generate new, synthetic traces which can be used for trace-based path loss models in simulation.

5.3 Future Work

Several improvements and features can be considered in the future to improve the work developed. They are mentioned in this section as possible guidelines for future work.

The most evident lacking feature is a graphical user interface for interacting with the platform. Such an interface would allow the end user to more intuitively operate the system and understand the results of the prediction process since data visualisation tools such as plots or tables could be employed. Other features missing from the platform, such as authentication and authorisation features, the possibility of the user defining custom algorithms to apply to datasets, or more fine-grained operations on datasets such as allowing for defining processes for automatically cleaning and preprocessing data would add to the usefulness of the platform. Furthermore, integration

with existing platforms for the automation of the generation of trace based network experiments [Mot18] would allow for the automatic generating of synthetic trace-based experiments to be directly run in the ns-3 network simulator.

In terms of the machine learning approach, several improvements can be added to evaluate the fitness of a machine learning approach to the problem at hand. While in this work two different approaches to the regression problem were tried – one where the models were trained to predict the expected RSSI value for a given sample directly, and another where the model was trained to predict the residual error of a baseline model such as the Friis path loss model, where predictions obtained from the model consist of the difference between the value for the signal strength given by the baseline model and a residual error predicted by the model – other approaches should be considered. In particular, several path loss models such as the Log distance model described in Section 2.1.3, take into account parameters which must be set according to the experiment, often by hand. One potential approach to using machine learning to estimate path loss could be to train a model which optimises the value of such parameters. Yet another possible approach could be considered by combining a traditional path loss model and the predictions obtained from a trained machine learning model.

In this work the evaluation of the training algorithms was performed by splitting the dataset into contiguous splits used to perform cross validation. This approach is not sufficient to validate the suitability of the models for generation of new network traces which can be used in network simulators with better results than the simple application of a conventional path loss model. In the future an experiment could be conceived where two similar nodes perform different routes over the experiment area, resulting in two datasets, one of which can be used to train a model, and the other to validate that the trained model outperforms the current path loss estimators present in the simulators.

References

[AP77] K. Allsebrook and J. D. Parsons. Mobile radio propagation in British cities at frequencies in the VHF and UHF bands. *IEEE Transactions on Vehicular Technology*, 26(4):313–323, November 1977.

[ASJH11] Paul Adamczyk, Patrick H Smith, Ralph E Johnson, and Munawar Hafiz. Rest and web services: In theory and in practice. In *REST: from research to practice*, pages 35–57. Springer, 2011.

[AV16] P. Agrawal and M. Vutukuru. Trace based application layer modeling in ns-3. In *2016 Twenty Second National Conference on Communication (NCC)*, pages 1–6, March 2016.

[Cha14] Derek Chandler. 2-ray ground reflection diagram including variables for the 2-ray ground reflection propagation algorithm., 2014. https://commons.wikimedia.org/wiki/File:2-Ray_Ground_Reflection.png.

[Cha15] Deka Ganesh Chandra. Base analysis of nosql database. *Future Generation Computer Systems*, 52:13 – 21, 2015. Special Section: Cloud Computing: Security, Privacy and Practice.

[Dey17] Sandipan Dey. Comparing spectral partitioning / clustering (with normalized graph laplacian) with kmeans clustering in r, 2017.

[dmBM14] Wim de mulder, Steven Bethard, and Marie-Francine Moens. A survey on the application of recurrent neural networks to statistical language modeling. *Computer Speech and Language*, 30, 01 2014.

[dra17] Dramatiq (https://dramatiq.io/). https://dramatiq.io/, 2017.

[EGT+99] V. Erceg, L. J. Greenstein, S. Y. Tjandra, S. R. Parkoff, A. Gupta, B. Kulic, A. A. Julius, and R. Bianchi. An empirically based path loss model for wireless channels in suburban environments. *IEEE Journal on Selected Areas in Communications*, 17(7):1205–1211, July 1999.

[F+] Daniel B Faria et al. Modeling signal attenuation in ieee 802.11 wireless lans-vol. 1.

[FCR17] Helder Fontes, Rui Campos, and Manuel Ricardo. A trace-based ns-3 simulation approach for perpetuating real-world experiments. In *Proceedings of the Workshop on Ns-3*, WNS3 '17, pages 118–124, New York, NY, USA, 2017. ACM.

[FCR18] Helder Fontes, Rui Campos, and Manuel Ricardo. Improving the Ns-3 TraceBasedPropagationLossModel to Support Multiple Access Wireless Scenarios. In *Proceedings of the 10th Workshop on Ns-3*, WNS3 '18, pages 77–83, New York, NY, USA, 2018. ACM.

REFERENCES

[FGM+99] Roy Fielding, Jim Gettys, Jeffrey Mogul, Henrik Frystyk, Larry Masinter, Paul Leach, and Tim Berners-Lee. Hypertext transfer protocol–http/1.1. Technical report, 1999.

[Fla12] Peter Flach. *Machine Learning: The Art and Science of Algorithms that Make Sense of Data*. Cambridge University Press, 2012.

[Fou18] Django Software Foundation. Django (version 2.0). https://djangoproject.com, 2018.

[FP04] C. Fretzagias and M. Papadopouli. Cooperative location-sensing for wireless networks. In *Second IEEE Annual Conference on Pervasive Computing and Communications, 2004. Proceedings of the*, pages 121–131, March 2004.

[Fri46] H. T. Friis. A Note on a Simple Transmission Formula. *Proceedings of the IRE*, 34(5):254–256, May 1946.

[FT00] Roy T Fielding and Richard N Taylor. *Architectural styles and the design of network-based software architectures*, volume 7. University of California, Irvine Irvine, USA, 2000.

[GR92] Jim Gray and Andreas Reuter. *Transaction Processing: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition, 1992.

[HLPR12] D. He, G. Liang, J. Portilla, and T. Riesgo. A novel method for radio propagation simulation based on automatic 3d environment reconstruction. In *2012 6th European Conference on Antennas and Propagation (EUCAP)*, pages 1445–1449, March 2012.

[Kum12] Neeraj Kumar. Using support vector machines effectively, 2012.

[Lah18] Lahrmam. Maximum-margin hyperplane and margin for an svm trained on two classes. samples on margins are called support vectors., 2018. https://commons.wikimedia.org/wiki/File:SVM_margin.png.

[Lee85] W. C. Y. Lee. Estimate of local average power of a mobile radio signal. *IEEE Transactions on Vehicular Technology*, 34(1):22–27, February 1985.

[Mat18] MathWorks. Train regression trees using regression learner app, 2018.

[MK00] A. Medeisis and A. Kajackas. On the use of the universal Okumura-Hata propagation prediction model in rural areas. In *VTC2000-Spring. 2000 IEEE 51st Vehicular Technology Conference Proceedings (Cat. No.00CH37026)*, volume 3, pages 1815–1818 vol.3, May 2000.

[Mot18] João Mota. Framework for Offline Wireless Network Experimentation. Master's thesis, Faculdade de Engenharia da Universidade do Porto, Portugal, 2018.

[MPP+10] Alexandra Moraru, Marko Pesko, Maria Porcius, Carolina Fortuna, and Dunja Mladenić. Using machine learning on sensor data. *CIT*, 18, 01 2010.

[OL04] P. Owezarski and N. Larrieu. A trace based method for realistic simulation. In *2004 IEEE International Conference on Communications (IEEE Cat. No.04CH37577)*, volume 4, pages 2236–2239 Vol.4, June 2004.

REFERENCES

[PLK18] Riccardo Petrolo, Yingyan Lin, and Edward Knightly. ASTRO: Autonomous, Sensing, and Tetherless netwoRked drOnes. pages 1–6, June 2018.

[PSG13] C. Phillips, D. Sicker, and D. Grunwald. A Survey of Wireless Path Loss Prediction and Coverage Mapping Methods. *IEEE Communications Surveys Tutorials*, 15(1):255–270, 2013.

[PVG+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[rab18] Rabbitmq tutorials. https://www.rabbitmq.com/tutorials/tutorial-three-python.html, 2018.

[RN03] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2 edition, 2003.

[RSK08] Joshua Robinson, Ram Swaminathan, and Edward W. Knightly. Assessment of Urban-scale Wireless Networks with a Small Number of Measurements. In *Proceedings of the 14th ACM International Conference on Mobile Computing and Networking*, MobiCom '08, pages 187–198, New York, NY, USA, 2008. ACM.

[Sch90] Robert E. Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–227, Jun 1990.

[Sch13] Robert E. Schapire. *Explaining AdaBoost*, pages 37–52. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[Shi10] Hweechul Shin. *Measurements and Models of 802.11B Signal Strength Variation Over Small Distances*. Thesis, University of Delaware, 2010.

[SJD12] C. Sommer, S. Joerer, and F. Dressler. On the applicability of Two-Ray path loss models for vehicular network simulation. In *2012 IEEE Vehicular Networking Conference (VNC)*, pages 64–69, November 2012.

[Spe91] D. F. Specht. A general regression neural network. *Trans. Neur. Netw.*, 2(6):568–576, November 1991.

[SSBD14] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014.

[Uni12] International Telecommunication Union. Modelling techniques for propagation prediction. pages 25–50, July 2012.

[VVLB+08] T. Verplancke, S. Van Looy, D. Benoit, S. Vansteelandt, P. Depuydt, F. De Turck, and J. Decruyenaere. Support vector machine versus logistic regression modeling for prediction of hospital mortality in critically ill patients with haematological malignancies. *BMC Medical Informatics and Decision Making*, 8(1):56, Dec 2008.

REFERENCES