Master of Science Thesis in Communication Systems
Department of Electrical Engineering, Linköping University, 2017

Machine Learning for Beam Based Mobility Optimization in NR

Björn Ekman

Master of Science Thesis in Communication Systems

Machine Learning for Beam Based Mobility Optimization in NR

Björn Ekman

LiTH-ISY-EX--17/5024--SE

Supervisor: **Julia Vinogradova**
ISY, Linköpings universitet
Pradeepa Ramachandra
Ericsson Research, Ericsson AB
Steven Corroy
Ericsson Research, Ericsson AB

Examiner: **Danyo Danev**
ISY, Linköpings universitet

Division of Communication Systems
Department of Electrical Engineering
Linköping University
SE-581 83 Linköping, Sweden

Abstract

One option for enabling mobility between 5G nodes is to use a set of area-fixed reference beams in the downlink direction from each node. To save power these reference beams should be turned on only on demand, *i.e.* only if a mobile needs it. An User Equipment (UE) moving out of a beam's coverage will require a switch from one beam to another, preferably without having to turn on all possible beams to find out which one is the best.

This thesis investigates how to transform the beam selection problem into a format suitable for machine learning and how good such solutions are compared to baseline models. The baseline models considered were beam overlap and average Reference Signal Received Power (RSRP), both building beam-to-beam maps. Emphasis in the thesis was on handovers between nodes and finding the beam with the highest RSRP. Beam-hit-rate and RSRP-difference (selected minus best) were key performance indicators and were compared for different numbers of activated beams.

The problem was modeled as a Multiple Output Regression (MOR) problem and as a Multi-Class Classification (MCC) problem. Both problems are possible to solve with the random forest model, which was the learning model of choice during this work. An Ericsson simulator was used to simulate and collect data from a seven-site scenario with 40 UEs. Primary features available were the current serving beam index and its RSRP. Additional features, like position and distance, were suggested, though many ended up being limited either by the simulated scenario or by the cost of acquiring the feature in a real-world scenario.

Using primary features only, learned models' performance were equal to or worse than the baseline models' performance. Adding distance improved the performance considerably, beating the baseline models, but still leaving room for more improvements.

Acknowledgments

I would like to express my gratitude to all persons who have helped and supported me through work with the thesis:

- Pradeepa Ramachandra, who have guided me through the whole process: asking the uncomfortable questions, looking from a different angle, proofreading my first draft and patiently waited for my last.
- Steven Corry for contributing new ideas and answering all my (odd) questions about machine learning.
- Ericsson LINLAB and all who worked or wrote their thesis there for their friendliness and great atmosphere. Really the best place to write a thesis.
- Julia and Danyo at the Communication Systems division for their support and patience.
- Simon Sörman for sharing his LaTeX-style and LaTeX-knowledge.
- My parents, sister and friends for helping me think of something else than features, beams and references sometimes.

Linköping, January 2017
Björn Ekman

Contents

Notation

Acronyms

Acronym	Description
AUC	Area Under Curve
BS	Base Station
CART	Classification And Regression Tree
CP	Cyclic Prefix
CQI	Channel Quality Information
CRS	Cell-specific Reference Signal
FVV	Feature Vector Virtualization
ISI	Inter Symbol Interference
LTE	Long Term Evolution
MCC	Multi-Class Classification
MIMO	Multiple Input Multiple Output
MO	Multiple-Output
MRS	Mobility Reference Signal
MSE	Mean Square Error
MTR	Multi-Target Regression
NR	New Radio
OFDM	Orthogonal Frequency Division Multiplexing
PP	Pairwise Preference
RC	Regressor Chains
ROC	Receiver Operating Characteristics
RSRP	Reference Signal Received Power
RSRQ	Reference Signal Received Quality
RSSI	Received Signal Strength Indicator
SCM	Spatial Channel Model
SON	Self Organizing Network
SST	Stacked Single-Target
ST	Single-Target
SVM	Support Vector Machine
UE	User Equipment

1

Introduction

Today's radio communication systems operate in very diverse environments with many tuning parameters controlling their behavior and performance. Traditionally these parameters were set manually, but more and more complex systems lead to a search for an automated and adaptive solution. These demands, combined with large amounts of available data, make statistical algorithms that learns from data highly interesting.

One central concept in mobile radio communication systems is mobility, *i.e.* the system's ability to keep a connection to a *User Equipment* (UE) alive even if that UE needs to switch access point. The procedure of transferring a connection to another *Base Station*s (BS or node) is called a handover. Handover-procedures are important to tune correctly as failure to do so will result in either dropped connections or a ping-pong behavior between BSs. To be able to choose the correct BS to handover to, the serving BS needs some information on expected signal quality. In today's systems that information is gathered through the use of regular and frequent reference signals sent by the BSs for the UEs to measure against.

In LTE, each cell has a *Cell-specific Reference Signal* (CRS) associated with it. The CRS contains information used for random access, handover and data demodulation. The CRSs are transmitted several times a millisecond, see Section 3.1.2, which makes a lot of information always measurable in the system. This information makes it possible to trigger a handover when an UE detects another cell being stronger. In that case a measurement report is sent to the serving BS, which then decides where to handover the UE to. Unfortunately, the reference signals consume a lot of power making them one of the main sources of power consumption in LTE, regardless of load [9].

One suggestion for 5G, or *New Radio* (NR) as 3GPP call the new radio access

technology, is to provide mobility using several narrow and area-fixed beams, so-called mobility beams, sent in the downlink direction from each BS to UEs in the vicinity. A UE will be served by one of the mobility beams and handed over to another when a beam switch condition is triggered, serving a somewhat similar concept to that of LTE cells and their CRSs. These mobility beams will however be many more and only activated on demand. Their number and narrowness make measurements on them better represent the quality of a beam focused directly towards the UE, but at the same time impossible to measure the quality on all beams. With hundreds of beams to choose from, only a fraction of them being relevant, there is a need to choose which beams to measure in a smart way.

Machine learning methods and algorithms have successfully been used in different fields. There are several reasons why machine learning seems likely to perform in this case as well: it benefits from large amount of data, models built using the data collected in one node will adapt to the conditions in that node, no manual labeling of data is needed (it does however require dedication of the system's time and resources).

1.1 Purpose

The purpose of this thesis is to investigate how supervised learning can be used to help a node select the destination beam during handovers. The study should provide a better understanding of the problem from a machine learning perspective and given an indication of expected performance.

1.2 Problem Formulation

Ideally mobility would be handled by an algorithm able to always keep each UE in its best possible beam, while still minimizing the number of handovers, reference measurements, active beams and other costs. However, finding such an ideal algorithm lies a bit outside the time scope of a master thesis.

Instead, the problem was limited to predicting reference signal strength, *Reference Signal Received Power* (RSRP) as it is called in LTE, based on a limited number of features in a simulated environment, see Figure 1.1. The ability to accurately predict the RSRP is crucial to perform NR handovers between mobility beams. Minimization of the other costs will then be more like LTE and are not treated in this thesis. Mainly inter-node handovers were considered as it is in this scenario this type of handover is believed to be needed.

1.2.1 Goal

Construct and evaluate a supervised learning procedure that, when given a sample from the simulated data, suggests a set of candidate beams, which with high probability contains the best beam. The probability should be higher than the baseline models: Section 5.5.2. The best beam is, in this thesis, the beam with the highest RSRP.

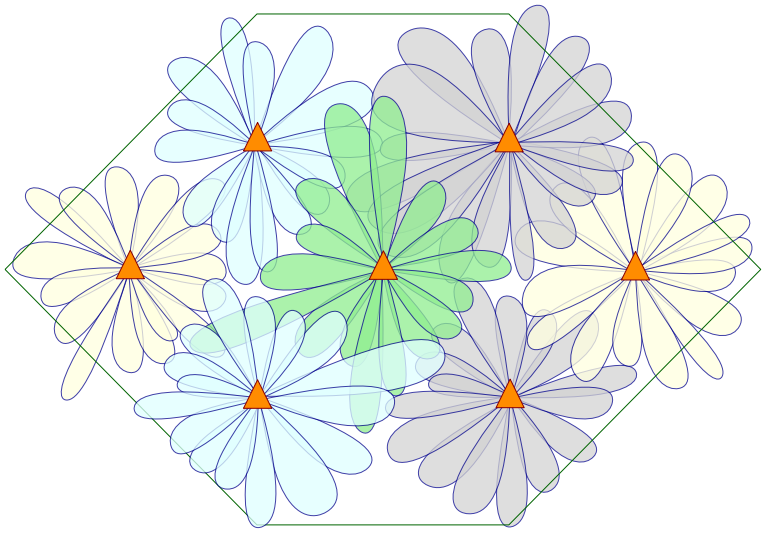


Figure 1.1: *Conceptual image of the simulated deployment*

Desired Outcome

This study on candidate beam selection should result in:

- an method of transforming the problem to supervised learning
- suggestion on features and their importance
- performance metrics and comparison to simpler methods
- insight into the trade-off between number of candidates beams and probability of finding the best beam.

1.3 Delimitation

One can easily imagine more questions of interests before a learning algorithm like this can be deployed in a real network. These have not been top priority during this master thesis, but could be something to consider in future studies.

- How to collect training data?
- How to adjust for different UEs and UEs' characteristics?
- For how long is the learned model valid?

1.4 Disposition

It is possible to read the thesis from front to end, but readers experienced with machine learning and/or LTE might find the theory chapters, Chapter 2-4, somewhat basic. In that case skipping straight to Chapter 5 might be more interesting.

Introduction

Provides a brief introduction and motivation to the thesis and its goals.

Background

Further describes the thesis background and the changes anticipated moving from LTE to NR.

Communication Theory

Describes some details on LTE resource management and its definition of RSRP.

Learning Theory

Presents a brief introduction to machine learning before going deeper into some of the methods used in the thesis.

Method

Starts with a system description, then a data overview then a quick introduction to the chosen learning algorithm and finally the performance metrics and base line models.

Software Overview

Contains a detailed description of simulator parameters and setup.

Results

Five different scenarios are studied and their results presented.

Discussion

Ideas for improvement and future work are provided.

2
Background

This chapter further describes and compares aspects of LTE and NR relevant for this thesis.

2.1 5G

The goals for NR are many and all quite ambitious. Included are several times higher data rate, lower latency and less consumed power [10]. To achieve this, several parts of LTE needs to be reworked. Described in this chapter is Ericsson's view on some of the concepts discussed for NR. 3GPP's work with NR standardization started roughly at the same time as this thesis, spring 2016 [19], which means some details are likely to change with time.

2.2 Beamforming

One important NR-building-block are multi-antenna techniques, most importantly transmitter side beamforming and spatial multiplexing. Both ideas use an array of antennas at the BS and make no demands on the number of antennas at the UE. With several antennas at the BS and either several antennas at the receiver or several receiving UEs we get a *Multiple Input Multiple Output*-system (MIMO). The basic idea of beamforming is to alter the output signal from each BS-antenna in such a way that the electromagnetic waves add up constructively in one area and destructively everywhere else. The main goal of transmitter side beamforming is to focus the signal energy more effectively and thereby increase received signal energy at a given UE. The main goal of spatial multiplexing is to reduce interference to other UEs to allow the BS to serve multiple UEs concurrently, using the same time and frequency slots [15]. The overall increase in received power is

called beamforming gain. In an ideal situation beamforming-gain can be as high as a multiple close to the number of transmitting antennas [8].

Advanced beamforming requires the BS to somehow estimate the channel responses for every UE-BS-antenna pair. If the channel estimates are updated often enough and are accurate enough, the BS will be able to always focus the signal energy at the UE tracking it as it moves around. It makes a sharp contrast to traditional broadcast based systems in which most of the signal energy goes wasted into regions where the UE is not located. The channel estimations are crucial for beamforming and are estimated using pilot signals, either sent from the UE in the uplink, relying on a TDD-channel and reciprocity, or by sending pilots in the downlink and UE reporting the result back. Using the channel estimates the BS computes complex weights which scale and phase-shift the output of each antenna in such a way that a beam is formed. The application of weights in this fashion on the transmit side is called precoding. A simpler version of beamforming uses static precoding, *i.e.* static weights and shifts, to transmit a beam into a fixed area.

A more extensive overview of different multi-antenna schemes and their usage in LTE is given in [8, chapter 5].

2.3 Mobility in LTE

In LTE there are several different downlink reference signals but most important for mobility are the CRS-signals. Through the CRS-signals the UE can measure the RSRP. Together with total received power, *Received Signal Strength Indicator* (RSSI), RSRP is used to compute a signal quality measurement called *Reference Signal Received Quality* (RSRQ).

There are several possible ways to trigger a measurement report from the UE to its BS, letting the BS know that an handover might be a good idea [4, s. 5.5.4]. The most common triggers are based on measurements of RSRP/RSRQ on both the serving node and surrounding neighbors. Reports can be triggered either when an absolute measurement passes a threshold or when the difference between serving and neighbor passes a threshold. These thresholds are set by the serving node and can be altered depending on the reports the UE transmit.

2.3.1 LTE Neighbor Relations

To help with handover decisions each BS builds a so called neighbor relation table. It is constructed and maintained using UE measurement reports. For this purpose extra reports can be requested by the BS. The serving node uses that info to establish a connection to each node that was in the report. Handover performance to a particular neighbor is also stored in the table and the system gradually builds an understanding of which neighbors that are suitable candidates for handovers. The table also allow operators to blacklist some neighboring cells which will never be considered for handovers. The table and its associated functionality is part of a series of algorithms referred to as *Self Organizing Network* (SON),

meant to optimize and configure parameter-heavy parts of the network.

2.4 Moving towards NR

Several key concepts on how we view cellular networks will need to change when moving towards NR. Driving these changes are new demands on an ultra lean design and an extensive use of beamforming. The ultra lean design strives limit unnecessary energy consumption [10]. Energy can be saved by limiting "always there" broadcast signals and send more information on demand using some form of beamforming.

At the same time as information is removed from the network, there are still demands on seamless handovers and services optimized for each user's needs. Increasing the number of BS-antennas increases the system's complexity, but also allows for a more flexible beamforming and better beamforming gain. Building systems in a smart way make it theoretically possible to use hundreds of antennas at the base station, all cooperating to serve several users concurrently. This concept is called *Massive MIMO*, [15], and is one of the big changes in the new system.

Both the lean design and the extra amount of antennas pose a problem for mobility. It is simply unfeasible for a UE to measure all the reference signals available in a timely manner. One way of combining the concept of Massive MIMO and mobility is to use several pre-configured precodings which results in some area-fixed/direction-fixed beams. These direction-fixed beams are referred to as mobility beams and are there to help NR provide mobility between nodes. The mobility beams can be seen as a NR's counterpart to CRS-signals in LTE.

2.5 Mobility in NR

The number of mobility beams will make it unfeasible to measure the RSRP of all mobility beams close by, which makes mobility similar to LTE impossible. Instead the BS has to determine when a UE might need a handover, guess which mobility beams that are best to measure against, ask the UE to do so and eventually make a handover decision based on the measurements from those selected few beams. This thesis focus on how to choose which beams to measure against.

3

Communication Theory

This chapter introduces some aspects of mobile radio communication relevant to this thesis.

3.1 LTE Measurements

This section describes the LTE resource structure and how RSSI, RSRP and RSRQ are computed from that.

3.1.1 OFDM

A complex number can be divided into a sine and cosine function of a certain frequency, $[0, 2\pi]$, and certain amplitude. This makes complex numbers very good at describing periodic functions and waves, *e.g.* an alternating electric current or an electromagnetic wave. Antennas make it possible to convert an electric current into a radio wave and back.

In digital radio communication systems bits are mapped to complex numbers, a.k.a. symbols. These symbols are then mapped to an electromagnetic waveform of much higher frequency, called carrier frequency. The wave's characteristics, such as how it travels and its ability to penetrate walls, is tightly connected to its carrier frequency.

The symbol rate, symbols/second, is an important tuning parameter as it determines both data throughput, decoding difficulty and used bandwidth. A high symbol rate might cause the transmitter to interfere with its own communication, denoted *Inter Symbol Interference* (ISI). ISI is caused by waves traveling different paths and arriving at different times at the receiver. The effect can be mitigated either by extending the duration of each symbol or place a guard period between

each symbol, both at the cost of lower symbol rate and data throughput. How long the guard interval needs to be depends on the channel properties and can vary considerably between different scenarios (rural/urban, slow/fast moving terminals).

To combat the ISI LTE uses *Orthogonal Frequency Division Multiplexing* (OFDM). OFDM is applied just before transmission, splitting the symbols into N different streams sent on N different carriers (called subcarriers). Instead of one stream with symbol rate R, this becomes N streams with symbol rate $\frac{R}{N}$. Placing guard intervals in the single-stream case is expensive as it requires one guard interval per symbol. In OFDM the guard intervals are placed between each OFDM-symbol effectively resulting in one guard interval per N symbols. The guard intervals in OFDM are commonly referred to as *Cyclic Prefix* (CP), as using the last part of a symbol as a guard before itself suits the OFDM Fourier transform implementation nicely.

3.1.2 Resource Blocks

It is convenient to visualize the LTE resources in a time-frequency plot, see Figure 3.1. The smallest time-unit is an OFDM-symbol. Six (extended CP) or seven (normal CP) OFDM-symbols make up a slot, two slots a subframe (one ms) and ten subframes a radio frame (ten ms). The smallest frequency-unit is a 15 kHz subcarrier. One subcarrier times one OFDM-symbol is called a resource element. Twelve subcarriers time one slot, *i.e.* 84 resource elements, make up one resource block. Several resource blocks like the one shown below are used to cover the whole bandwidth.

The smallest elements the scheduler can control is the resource elements. In a resource block, generally four resource elements, two in the first symbol and two in the fourth symbol, contain CRS-signals. The CRS-signals contain the information necessary to identify a cell, setup a connection with the cell and estimate the channel response. The channel response is used by the UE to correctly demodulate the data sent in the surrounding resource elements, as the channel response is roughly the same for elements close to each other.

The LTE version of MIMO allows the BS to send several resource blocks concurrently, using the same time and frequency resource. This spatial multiplexing is in LTE commonly referred to as using multiple layers. All overlapping resource elements will interfere with each other, but as long as the channel responses are correctly estimated the UE will be able to demodulate the data correctly. To achieve this, all layers need to be silent when another layer sends a CRS. This reduces the maximum number of usable layers. In the first versions of LTE a maximum of four layers was allowed. Later revisions have enable different types of reference signals and constructed ways to allow for even more layers, but are not discussed here.

3.1.3 Signal Measurements

The following details are important when defining RSSI and RSRP: 1) only some OFDM-symbols have resource elements with CRS-signals 2) only some of the re-

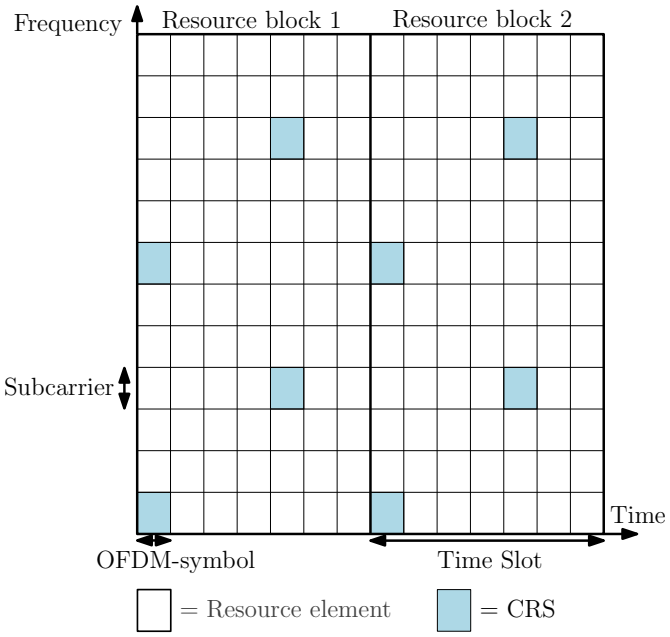


Figure 3.1: *A time-frequency division of a LTE subframe.*

source elements in a block carries CRS-signals 3) LTE systems create a lot of self-interference when using multiple layers 4) resource elements with CRS-signals are free from interference.

From the 3GPP specification 36.214 [3]:

> *E-UTRA Carrier Received Signal Strength Indicator (RSSI), comprises the linear average of the total received power (in [W]) observed only in OFDM symbols containing reference symbols for antenna port 0, in the measurement bandwidth, over N number of resource blocks by the UE from all sources, including co-channel serving and non-serving cells, adjacent channel interference, thermal noise etc.*

In short: RSSI measures the total received power over all resource elements in one OFDM-symbol. Note that the average is computed over the time-axis and the total is computed over the frequency-axis, across N resource blocks (generally over the six resource blocks closest to the carrier frequency). This makes RSSI grow with more allocated spectrum and/or if more data/interference is received. Because of this property RSSI is considered unsuitable as a channel quality estimate.

Instead LTE introduces RSRP and RSRQ.

From the 3GPP specification 36.214 [3]:

> *Reference signal received power (RSRP), is defined as the linear average over the power contributions (in [W]) of the resource elements that carry*

> *cell-specific reference signals within the considered measurement frequency bandwidth.*

In short: RSRP measures the average received power over CRS-resource elements in one OFDM-symbol. Note that the average is computed over the frequency-axis, but only considering elements containing CRS. While not very clearly stated here, the bandwidth considered is the same as for RSSI (*i.e.* over N resource blocks). The way RSRP is computed makes it a fair signal strength estimate good for cell strength comparisons.

Although RSRP is better than RSSI it can't tell the whole truth. In an attempt to capture the best of both worlds, RSRQ is computed as a weighted ratio between them.

From the 3GPP specification 36.214 [3]:

> *Reference Signal Received Quality (RSRQ) is defined as the ratio N ∗ RSRP/(E-UTRA carrier RSSI), where N is the number of RB's of the E-UTRA carrier RSSI measurement bandwidth. The measurements in the numerator and denominator shall be made over the same set of resource blocks.*

All three measurements are computed by the UE but only RSRP and RSRQ are reported back to the serving node. Both measurements are usually reported in dBm, which is a logarithmic scale power unit. It is computed as the ratio in decibels of the measured power to the power of one milliwatt, see Equation 3.1.

$$rsrp[dBm] = 10 * log_{10}(1000 * rsrp[W]) = 30 + 10 * log_{10}(rsrp[W]) \tag{3.1}$$

4

Learning Theory

This chapter introduces machine learning and gives an overview of the literature studied for this thesis. First traditional supervised learning and the general goal of learning is introduced. Focus is then shifted to learning problems with several output variables, followed by a brief overview on the random forest algorithm. Finally, an honorable mention to the learning-to-rank problem.

4.1 Machine Learning Introduction

Machine learning is a rather loosely defined field with strong ties to computational science, statistics and optimization. The goal in machine learning is to learn something from data, either to make predictions or to extract patterns. Today, data is available in abundance which makes machine learning methods applicable to a wide area of problems in very different fields, such as medicine, search engines, movie-rankings, object recognition etc.

Machine learning and pattern recognition is usually divided into three sub-domains: supervised learning, unsupervised learning and reinforcement learning. Although the focus in this thesis will be on supervised learning, all three of them are described very briefly here.

Supervised Learning

The goal in supervised learning is to approximate some unknown function using several observed input-output pairs. The goal is to find a function that generalizes well, *i.e.* has a low error on previously unseen inputs.

Unsupervised Learning

In unsupervised learning, also called data mining, the algorithm is given a lot of data and asked to find a pattern. Comparing it to supervised learning: the

learner is only given the inputs and asked to find patterns among them. Usually this is done by finding clusters or by analyzing which feature/dimension is the most important one.

Reinforcement Learning

Reinforcement learning is somewhat different than the other two. It can be described as an agent, the learner, interacting with an environment through a set of actions. After each action the agent is updated with the new state of the environment and the reward associated with the new state. One central aspect of reinforcement learning is the concept of delayed reward - an action profitable right now might lead to a lower total reward in the end (*e.g.* in chess: winning a pawn, but losing the game five moves later).

4.2 Supervised Learning

Road map

Supervised learning involves a lot of choices and things to consider, but most of them can be grouped in to one of four steps: data collection, pre-processing, learning and evaluation. In this chapter the description of machine learning will start at step three, basically assuming data served on a silver plate with no quirks and errors. Continuing with that assumption step four, performance metrics, is then investigated. Finally, we go back to step two and deals with data closer to the real-world. Details on step one, data collection, will follow later in Chapter 5 with more focus on the practical aspects of the thesis.

Learning Framework

In supervised learning the learner is given many labeled samples and tries to generalize from them. Each sample is described by p features arranged in feature vector, $\mathbf{x} = (x_1, x_2, ..., x_p)$, and a target, y. Features (and targets) can be any property represented as a number, *e.g.* height, light intensity, signal strength and number of pixels.

In traditional supervised learning the target is a single output value, taken either from a binary set (binary classification), a discrete set (multi-class classification) or a real value (regression). Given a dataset of N samples, $(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)$, drawn from some, generally unknown, distribution $Pr\{X, Y\}$, the goal is to learn the function that best describes the relationship between X and Y, here denoted f. The N samples are divided into a training set, (X_{train}, Y_{train}), used to learn a classifier $c : \mathbf{X} \mapsto Y$ and a test set, (X_{test}, Y_{test}), used to estimate the classifier performance according to some loss function $L(c(X_{test}), Y_{test})$.

Any c could in theory be close to f and it might take too long time to test all possible cs. To limit the search space a model is chosen which represents a set of functions, sometimes called hypothesis set, and the task is reduced to find the best function in that set. A learning algorithm searches, led by training examples, through the hypothesis set to find a good approximation of f. Let c^* denote the function in the hypothesis set that best approximates f.

How good the solution is depends very much on the given data and on the chosen model. A simple model, *e.g.* a linear model $\hat{y} = \sum_{i=0}^{p} x_i w_i$, will be unable to capture any complex interactions between X and Y and will fare poorly if the sought function is more complex than that. On the other hand, a polynomial of high degree might fit the given training data perfectly but instead be unable to get new points correct. The former problem is called underfitting and the latter overfitting. Overfitting is the more common problem and is often more difficult to detect.

Bias & Variance

These problems are also well described by a bias-variance decomposition. In such a decomposition the prediction error of a model is divided into three parts: bias, variance and irreducible error. The irreducible error is from noise inherent to the problem and cannot be reduced. The bias comes from the model's ability to express f - a high bias indicates that c^* is far away from f. Variance captures the model's sensitivity to the samples used to train the model. With a higher model complexity (and flexibility) comes a higher variance and risk of overfitting. Using the examples mentioned above, the linear model have a high bias but low variance and the polynomial model have a low bias but a high variance. More detailed, mathematical oriented, bias-variance decompositions can be found in most machine learning textbooks, *e.g.* Bishop's "Pattern Recognition and Machine Learning" [5] or Louppe's "Understanding Random Forest" [18, p. 55-60].

4.2.1 Performance Metrics

To evaluate a classifier c, a loss function L needs to be chosen. Depending on the type of learning problem several different performance metrics can be applied.

Regression

MSE is probably the most common loss function in regression. It has nice analytic properties, making it easy to analyses and optimize.

Classification

In classification most metrics originates from a binary setup. In binary classification one class is denoted positive and the other negative. From this a confusion matrix, see table 4.1, can be constructed giving rise to four counters: true-positive (tp), true-negative (tn), false-positive (fp) and false-negative (fn). A good description of the confusion matrix and some of the metrics derived from it can be found in [20].

Table 4.1: *Confusion matrix*

	Estimated class	
True class	positive	negative
positive	true positive (tp)	false negative (fn)
negative	false positive (fp)	true negative (tn)

Combining these one can construct many metrics used for classification problems. A convenient notation when discussing these metrics is to let $B = B(tp, fp, tn, fn)$ stand for any of the binary metrics based on the confusion matrix.

Derived metrics

- $Accuracy = \dfrac{tp + tn}{tp + fn + fp + tn}$
- $Precision = \dfrac{tp}{tp + fp}$
- $Recall = \dfrac{tp}{tp + fn}$
- $F1 = \dfrac{2tp}{2tp + fn + fp}$
- *Receiver Operating Characteristics* (ROC)
- *Area Under Curve* (AUC)

The last two metrics demands a bit more explanation. The ROC-metrics is actually a curve describing a trade-off between false positive rate (a.k.a fall-out) and true positive rate (a.k.a recall). It is used in situations where the classifier outputs a class-probability rather than a absolute class. Instead of only zeros and ones, the algorithm estimates its confidence and gives a number between zero and one for each sample. Depending on which class is most important to classify correctly, different thresholds can be applied. In some cases it might be appropriate to split 0.5-0.5, in other cases 0.3-0.7 might be best. Each threshold will result in a different recall and fall-out. By testing all possible thresholds, from zero to one it is possible to plot the relationship between these two measurements, see Figure 4.1. The point on the curve closest to the upper left corner is usually regarded as the best threshold as the (0,1) coordinate represents a perfect classifier.

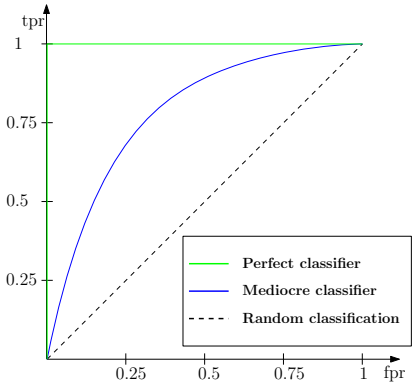


Figure 4.1: *Examples of typical ROC-curves. The area under each curve is the AUC-score of that classifier.*

However, the exact threshold is seldom of interest when comparing classifier performance, instead the general shape of the curve is the interesting part. The AUC-metric tries to capture that in one value by computing the area under the ROC-curve, hence the name. An AUC score of one represents a perfect score, one half random guessing and zero a perfect miss-classification (changing which class is positive and negative would result in a perfect score again).

4.2.2 Cross-validation

Cross-validation is the most common way of estimating model performance and comparing different models, or different sets of meta-parameters, with each other. In cross-validation the data is split into K folds, one fold is used for testing and the rest for training. The model is trained K times so that each fold is used as a test set once, see Figure 4.2. The performance metrics are then computed once for each test fold and averaged across the folds. K=10 is usually a good starting point to provide a good enough estimation [12, p. 216].

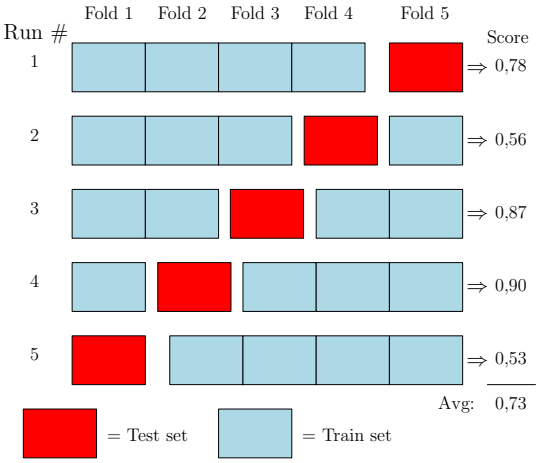


Figure 4.2: *Visualization of 5-fold cross-validation*

Things to consider:

- Which scoring/loss function to use?
- Which meta-parameters to include in the testing and which values to test for?
- How many combinations of meta-parameters to test?
- How many folds?

4.2.3 Pre-Processing

So far in our description of supervised learning there are several things to consider - which model to use, how to choose meta-parameters and which metric to use for evaluation. Introducing data to the mix will inevitably complicate it even

more. Collected data is usually not very suitable for learning, values might be missing or the model might demand normalized inputs.

Class Imbalance

Usually one wants to avoid an imbalanced classification problem - especially if it's important to get the minority classes correct. Otherwise the learning algorithm might return something that only ever predicts just one class. In such a case metrics help little as most common metrics (accuracy, precision, recall, F1) happily will report an almost perfect score. Accuracy will in fact report the initial class skew: in a case with 1% positive samples and 99% negative, a classifier predicting all samples as negative will still score an accuracy of 99%.

Imbalance can be avoided by selecting equal number of samples from each class, of course at the cost of reduced number of samples in the training set. One can also allow the minority class samples to be present more than once, drawing as many "new" samples as needed. Apart from clever sampling some learning algorithms also allow weights to be applied to the samples and in that way indicate which class is important and not.

Discrete Features

A feature belongs to one of three groups: 1) continuous and ordered 2) discrete and ordered 3) discrete and unordered a.k.a. categorical features. Many algorithms assumes features either to be of group 1 or group 2, and would like all features to be from the same group if possible. Categorical features are usually not handled by default and requires some extra work. One common way is so called one-hot-encoding, where each category gets its own binary feature vector (usually all but one of the categories are turned into new features as the last category can be derived from the values of the other categories). Another option is to convert the categorical feature into a discrete ordered feature by mapping the categories into integers instead. Both methods have their strengths and weaknesses and which one to use depends on the learning algorithm.

Feature Scaling

Most learning algorithms assumes features with nice properties, usually either zero mean with standard deviation of one or moved to a 0-1 interval. Otherwise the internal mathematics of some algorithms will favor features with large magnitude or large variance. Which one to use depend on the algorithm and possibly on the problem. This is usually done in a data pre-processing step where mean and standard deviation of each feature is estimated using the training set. The estimated values are then used to transform both the training and test set.

Missing Features

One problem that often occurs in practical application of machine learning are incomplete samples where one or more of the features are missing. In that situations one can either discard those samples or try to fill the spots with data that will affect the result as little as possible. The filling is called imputing and can be done in several ways. The simplest method is to pick a constant and replace all

missing values with that. Another alternative is to estimate the mean or median of the feature from the rest of the samples and use that to the missing values.

The imputation step can also be regarded as a supervised learning problem of it's own, where the goal is to predict the values for the intact data. An example is the missForest algorithm suggested by Stekhoven and Bühlmann [22].

4.3 Learning Multiple Targets

The traditional model of supervised learning assumes only one output target, but can be generalized to models with several targets.

4.3.1 Terminology

Multiple-target supervised learning is very flexible and therefore applicable to a wide arrange of problems. This also results in many similar names, sometimes with slightly different meaning.

Borachi et al. [6], gives an overview over the many faces of multiple-output and is a good starting point for further reading. In table 4.2 the most common model and their different characteristics are gathered. Unfortunately, when comparing with other sources, there seems to be a lack of consensus on the exact interpretations in some of these cases.

Table 4.2: *Names in supervised learning*

Name	Use	#Outputs	Output Type	Source
regression	R	scalar	real	[12]
binary	C	scalar scalar	binary class probability	[12]
multi-class	C	scalar vector	discrete class probabilities	[12], [20]
multi-target[1] multiple output[1]		vector		[6]
multi-variate[2] multi-response[2]	R/(C)	vector	real/(discrete)	[6]
multi-label[3]	C	vector	binary	[16], [20], [21]
multi-dimensional	C	vector	discrete	[6]
multi-task[4]		vector		[6]

Use: R - regression, C - classification, R/C - either R or C

1: All approaches with more than one target. Can be a mix of regression and classification.

2: Usually a regression task. Name used in traditional statistics.

3: Several binary targets. Usually a fixed set, but [16] considers an unknown number of targets.

4: More general than multi-target: *e.g.* different samples/features for different targets.

The names indicates slightly different problem setups which determines which learning algorithms are possible to use. Despite the different setups the multi-target nature also brings a set of issues and remarks true for most of them. In this thesis "multi-target regression" and "multi-class classification" are the main names used (see next paragraph for definition). Multiple-output is used when something concerns both problems.

In multi-class classification the learning problem is the same as the one described in Section 4.2. Y is a discrete set with d possible values. In multi-target regression we instead have $Y \subset R^d$, and y turns into a vector $\mathbf{y}$.

The multi-class classification can be brought closer to the multi-target regression by predicting class probabilities which results in input and output matrices of same size.

4.3.2 Strategies

A short review on some of the suggested strategies for dealing with multiple targets. More information can be found in [6].

Single Target

A common approach in dealing with a multi-target problems is to break apart the problem and study it as several separate single-output problems. In classification this is sometimes called binary-relevance. Similar regression problems lacks an name, but a name usable in both situations is the *single-target* model (ST). ST avoids some of the problem-complexity at the cost of computational complexity and potential loss of accuracy. In problems where there are dependencies between targets, considering all targets at once might help predictive performance something which is lost with the ST-approach.

In [21], Spyromitros-Xioufis et al. builds two models based on the ST-approach: *stacked single-target* (SST) and *regressor chains* (RC). These try to take advantage of the simplicity of the ST-approach but still make use of some of the dependencies between targets. In SST a first layer of STs are built, one for each target. Then the estimated targets are used as features into a second layer of ST models. In RC all ST-models are chained, the first one using the normal input features and then feeding its estimation to the next model and so forth.

Feature Vector Virtualization

Another way of transforming a multi-target problem into a single target problem is to use the target index as a feature. This creates a new data set with d times as many samples as in the original set. Each old sample is transformed into d new ones by copying the original feature vector and appending a new target index for each sample. It is then matched to the correct target value, either a single binary value (if the original problem was multi-class classification) or a single real value (if the original problem was multi-target regression). This data set transformation is described in [6] in the case of SVM models and is there referred to as *feature vector virtualization* (in this thesis FVV for short). It can in theory be applied to any learner, though in practice unclear how it effects the learner. It

is also unclear if it at all is applicable in the multi-class classification stage as the new data set will be heavily imbalanced (the ratio of positive to negative sample will be always be $1 : (d - 1)$).

Algorithm Adaptation

An alternative to the ST-approach is to extend a learning algorithm so that its loss function correctly measures the loss over all outputs. This is easy for some algorithms, *e.g.* decision trees, but might be more complicated for other methods. In [6] are more info on different models and algorithm adaptations, while this thesis focuses on Random Forest. Read more about Random Forest and its multiple output versions in Section 4.4.

4.3.3 Metrics

With the introduction of multiple targets or classes some adjustments are needed for some of the traditional metrics.

Regression

It's possible to ignore the multiple target nature to some extent when computing regression metrics as they are computed on a per-sample basis not effected by the outcome of other samples. Adding more targets only increases the number of terms when the value of all samples are averaged together.

It is also possible to define new metrics with slightly different properties, some which are considered in [6], though in this thesis traditional MSE was deemed sufficient, as it is the metric used internally in random forest.

Classification

In Section 4.2.1, some binary metrics were introduced. In multi-class classification and multiple output classification these metrics are usually extended using a "one-against-all" approach. When computing metrics one class/label at a time is considered positive and all other classes as negative. Combining the metrics to one value can be done in different ways depending on the nature of the problem.

Combining scores:

- **micro-average**: Add all confusion metric counts (tp, fp, fn, tn) for each class together, and then compute the relevant metric.
- **macro-average**: Compute the relevant metric for each class and then compute the average across the classes.
- **weighted-macro**: Similar to macro-average, but with a weighted average. The number of positive samples in a class, divided by the total number of samples, is used as weight.

In a situation where classes are imbalanced, micro-average will reflect the performance in the classes with many samples, whereas macro-average will be equally influenced by the score in each class. Which one to use depends on how severe the imbalance is and how important the minority classes are. The weighted-macro version is an attempt to get the best of both worlds.

4.4 Random Forest

Random Forest is a well known and easy to use, yet in some aspects complex, learning algorithm. It builds upon three ideas; ensemble of decision trees, bagging and random feature selection, evolved since the 1980 by numerous authors and eventually combined into one model. A famous paper by Leo Breiman et al. in 2001, [7], combined the concepts and introduced the name Random Forest. Because of that paper, and several contributions both before and after that, Breiman is quite commonly attributed the invention of the random forest concept. For a more thorough examination on the evolution of random forests, see [18, p. 69-73].

Forest Implementations

There are several free machine learning packages; *e.g.* scikit-learn (python), Weka (Java), TensorFlow and R; which helps users with the basic algorithm implementation and lets users focus on all the small details making the algorithms actually work. This thesis uses scikit-learn mainly because ease of use and well reputed documentation.

4.4.1 Building Blocks

Decision trees

Decision trees are tree-like structures using thresholds to split the decision region into several areas. There are several ways to build trees, here focus is on the *Classification And Regression Tree* (CART) model.

Each node in the tree corresponds to a threshold decision on one feature resulting in two child nodes. Several meta-parameters are used to control the depth of a tree. Training samples are eventually gathered in a leaf and all the samples in the leaf are used to approximate an output function. In CART this is a constant function (to allow for both regression and classification), usually an average in regression and a majority vote in classification.

In order to select which feature and threshold to use for a given split a loss function is optimized. Some choices are possible, but in general MSE is used in regression and gini-index or entropy in classification.

It is possible to split the nodes until there is only one sample per leaf, but such trees tend to have some problems: 1) low bias, but an extremely high variance 2) large trees are impractical as they require a lot of memory and time to build. To avoid these problems some sort of stopping criteria is needed. One can either stop splitting nodes when the samples in a node are too few or first fully develop the tree and then prune it to a satisfying size. However, with these methods comes extra meta-parameters that need to be tuned to each new problem which complicates the tree building process somewhat.

Important meta-parameter: max-depth, minimum samples in order to split a node and minimum samples in a leaf

Bagging

The idea with ensembles is to build very many weak classifiers that are computationally cheap to train and evaluate and then combine them. Each classifier will on its own be just better than guessing (roughly 51-60% accuracy) but when combined together and averaged they will converge to a solid classifier. One of the advantages with the averaging of many classifiers is that it only slightly increases the bias of the weak classifier (which is usually low) but greatly reduces the variance of it. This makes decision trees a popular candidate for the weak classifier, as their main drawback is their high variance.

It is however expensive to collect a new data set for each weak classifier trained. To save data bootstrap aggregation is used. Bootstrap aggregation, a.k.a. bagging, builds several new data sets from one set by sampling with replacement from the initial set. Assuming an initial training set with N samples, each new set is constructed by drawing, with replacement, N samples from the initial set. On average each weak classifier will be built using 63% of the original samples as well as multiples of these. [12, p. 217]

Important meta-parameter: number of estimators/trees

Random Feature Selection

Another way to further increase the randomness, and thus decrease the model variance, is to only consider a random subset of features at each split. Default values in scikit-learn is square root of the number of features for classification tasks and equal to the number of features (*i.e.* no random subset at all) in regression tasks. In a even more extreme fashion features and their threshold are chosen at random - this algorithm is called Extra Random Trees.

Important meta-parameter: number of features in subset

4.4.2 Constructing a Forest

To combine these three into a random forest the following procedure is used: use bagging to construct several datasets, build one tree per set using a random feature selection at each split, run test samples through each tree and average the result from each tree. This allows the random forest to massively reduce the variance of the decision tree model while only slightly increasing the bias.

As the variance is what the random forest combats best a general recommendation is to build the individual trees with as few restrictions and as deep as possible. However, this is a very general observation which might still be impractical due to memory consumption or simply not optimal for the particular problem at hand. Best is to set the meta-parameters of the forest through cross-validation.

4.4.3 Benefits of a Forest

The whole is greater than the sum of its parts is a classical saying and very true about a forest of randomized trees. Here some of the benefits from having many trees are described.

Out-Of-Bag Error

When constructing the data set of each tree some samples are left out, usually called out-of-bag samples. These can act as a test set for that tree to evaluate them and test. Assuming a forest of 100 trees, each sample will (on average) not be used in 37 of those. Instead those trees can be seen as a mini-forest capable to predict the value of that sample with quite good accuracy. With an increased number of trees this quickly becomes a relatively good estimate of prediction error and might serve as a replacement for the cross-validation (instead of building 10 forests for each set of meta-parameters only one forest should be enough).

Feature Importance

The random forest loses some of the easy interpretability of decision trees but offers instead new methods for analyzing results and data. A (rough) feature importance can be estimated by looking at how often it was used for splits. The measurement can be somewhat off in the case that features are heavily correlated.

4.4.4 Forest Pre-Processing Requirements

Decision trees are not very accurate but have some other nice properties: virtually no requirements on feature type and scaling, quite resistant to irrelevant features and easily interpretable [18, p. 26]. In [12, p. 313] and [14] a comparison is made between different types of learning algorithms which also highlights the merits of decision trees. Most notable is that it is the only algorithm noted as capable of handling both continues and discrete features. In [12, p. 272] it is also noted that decision trees have more options for handling missing features.

Most of these attractive properties stays when computing an ensemble of trees. The most obvious drawback is the loss of interpretability. The relative ease of use, while still providing good accuracy, is one of the main reasons why random forest and other tree based ensemble methods are popular.

Discrete Features

When considering thresholds, there isn't any difference between an continuous and a discrete (ordered) feature. This makes decision trees are good at dealing with a both continues and discrete features and can mix them freely.

Categorical features are a bit more complicated. A problem for random forest is random feature selection combined with one-hot-encoding. Ideally the encoded feature should still be treated as only one feature when creating the random subset, but support for that depends on the implementation. Transforming the categories to integers and treating them as a discrete, ordered feature works quite well with a forest as it ca deal with discrete features.

Decision trees also enables new opportunities for dealing with categorical variables. Just like for discrete ordered features it's possible to compute all possible splits on a categorical variable. However, the number of possible splits, S, increases exponentially with the number of categories, L, according to the formula $S = 2^{L-1} - 1$. In the special case of binary classification this can be reduced to

$L - 1$ splits, but otherwise exhaustive or random search over the full number of splits is needed. [18, p. 51-52]

Feature Scaling

As each split only considers one feature at a time, the magnitude/scaling of a feature compared to other features does not affect the results.

Missing Features

Decision trees offers an additional way of dealing with missing features called surrogate splits. In each node, a list of splits ordered from best to worst is constructed. If the best feature is missing in one sample, the next best feature will be checked and so forth until a valid split is found. Another alternative is to allow the sample to propagate down both sides of a split. [12, p. 272]

4.4.5 Multi-Target Forest

It is relatively easy to predict several targets in a random forest, at least as long as all targets are of the same type - either classification or regression. In that case the performance of each split is computed for each target and then averaged over all targets. This leads to splits with the best average performance. Correlation between targets are thus prioritized as it means more targets are help by just one split. Each leaf will in the end, instead of just one value, consists of a vector of output values. [18, p. 52-53]

Forests where the targets are mixed (both classification and regression) are also possible but a bit more involved, see [16] for further reading on that, and are not considered in this thesis.

4.4.6 Forest Limitations

It is also important to point to some of the weakness inherited in a random forest: 1) they are difficult to interpret 2) decision trees are generally bad at extrapolating as it is impossible to generate an output outside the range of input samples.

Scikit-learn

The essential parts of decision trees are well built and optimized in scikit-learn, but some details are unfortunately lacking (and unfortunately not so well documented). The scikit-learn decision tree model uses the CART-algorithm, but doesn't implement some of the extra details.

Notes on the scikit-learn implementation of random forests: no support for missing features and categorical are treated as discrete, ordered variables. It is possible to use one-hot-encoding, but all features will be treated equally and no knowledge that it in reality is only one feature will be used during feature randomization. Variable importance is supported, though not very clear on how it is implemented. Multi-target classification/regression is supported and it's possible to get class probabilities in a classification scenario.

4.5 Ranking

During the thesis some focus was directed towards the machine learning subfield ranking/learning-to-rank. This is a popular area usually focused on ranking documents when given a query. Methods are divided into three main groups: pointwise, pairwise and listwise. The pointwise approach is the easiest one, it focuses on predicting a score value for each sample which is then used to rank the objects and can be thought of as a traditional regression task. The pairwise methods expects input to be in the form of pairs of rankable objects, and the output indicating how important it is to rank one above the other. By passing all possible pairs through the ranker a complete ranking can eventually be constructed. The listwise approach is outside the scope of this thesis and not covered here. More information and example algorithms for each of these approaches can be found in Liu's Learning to Rank for Information Retrieval [17].

4.5.1 Label Preference Method

In Label Ranking by Pairwise Preferences, by Hüllermeier et al. [13], a quite specific ranking situation is studied and a suitable algorithm is suggested. They assume that each sample has one feature vector and a known finite number of labels/targets. The goal is to rank the labels for each new sample point.

The proposed solution build one binary classifier for each pair of labels returning a probability measure for how likely it is that lower indexed label should be ranked higher than the other label. One nice part of the idea is that each training sample only need to provide preference information on a subset of the label pairs (preferably random), not all of them. With enough samples the model will eventually be able to learn a complete ordering of the labels. One of the main drawbacks though, is the many models needed to learn: $l(l+1)/2$, where l is the number of labels.

5
Method

In this chapter the transformation from radio communication problem to a machine learning problem is described. Steps necessary to discuss are similar to those in the theory section: system description, data overview, pre-processing, learning models and performance metrics.

5.1 System Description

Studied in this thesis is a simulated LTE system, modified to allow for some fundamental ideas of NR mobility: more antennas and different reference signals. The layout of the map can be seen in Figure 5.1. In the figure there are seven nodes (also called sites or BS) and several conceptual beams. The actual number of beams per node were in this particular simulation set to 24. The smoothness and shapes of the beam are somewhat exaggerated: a beam can have very different shapes when taking reflections into account. Nevertheless, the picture is useful, showing beams of various sizes and shapes and the possibilities of them reaching far in to other beams and nodes.

Each node has three sectors with eight antennas each. Using eight different predefined precoding matrices, each sector can combine its antennas into eight mobility beams. In total, there are 168 antennas/beams present. For more details regarding simulation setup, parameters and assumptions see Chapter 6.

Each UE in the system is always served by one beam, denoted *serving beam*. Eventually, due to the movement of the UE, the quality of the serving beam will deteriorate and a new beam needs to be selected. To be able to select which beam to hand over to, the current serving BS needs some information about signal strength. With that it is possible to compare beams and judge which one is best.

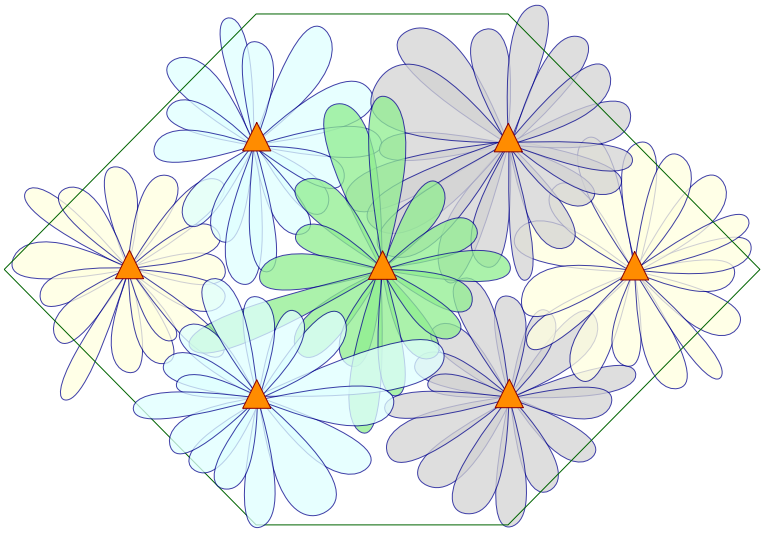


Figure 5.1: *The deployment used in the simulated scenario with conceptual beams drawn from each node. Colors added to easier differentiate between beams from different nodes.*

In LTE that is provided by UEs continuously monitoring and reporting the signal quality of surrounding cells. In NR this will be trickier since there are many more possible reference beams, which are not transmitted continuously.

In NR, a learned algorithm will help the node to come up with a good set of candidate beams, originating either from the serving node or from its neighboring nodes. These candidate beams will then be activated by the corresponding nodes, measured by the UE and reported back to the serving node. The serving node will then decide whether to handover or not and if so to which beam (and thus which node).

The role of machine learning is to help the node with the selection of the candidate beams. A set of beams is considered good if it: 1) contains few beams 2) has a high probability of containing the best beam. It is vital to limit the number of beams it activated, as each active beam consumes system resources. Best beam is the beam with the highest signal strength, a.k.a. RSRP. These two demands work against each other: more active beams will make it more likely that the best beam is among them but also consume more resources. Taken to its extreme, the learner will try to find the best beam and only suggest that one. This extreme case is a good starting point, as it is more easily converted into a machine learning problem. In Section 5.4, this strict demand will be lessened somewhat.

On a side note: the beam switch triggering in NR (which also needs to be reworked compared to LTE) was still in a conceptual state when starting this thesis, so some experimentation with that was also needed.

5.2 Data Overview

Here follows an overview of the available data and how it can be turned into features and targets.

5.2.1 Available data

The simulator offered much freedom regarding collection and storage of data to log. Early in the thesis a survey over available data was conducted and the following logs looked most promising to use for machine learning:

- beam indexes
- beam RSRP/CQI
- position/distance
- UE speed
- timing advance*
- node activity/load*

There are two types of costs associated with acquiring data, one simulator-based and one reality-based. The simulator-based cost is a combination of implementation time, memory consumption and machine learning pre-processing. The reality-based cost is the cost in system resources of acquiring a certain value, for example asking a UE to transmit something it would not normally do. The simulator costs are possible to mitigate given enough time, the reality-based costs are often more difficult to change.

*: Timing advance and node activity were not extracted as they were not prioritized and the simulator-cost was judged to be too high.

5.2.2 Features

Here the available data is turned into machine learning features. The features were eventually divided into two features groups: instant and history features. The former focused on values updated at each UE measurement, and the latter focused on past events and measurements. An asterisk marks data mainly used as learning target.

Instant features

- serving beam index
- destination beam index*
- serving beam RSRP
- non-serving beam RSRP*
- distance to serving BS
- position

- UE speed
- CQI

History features

- previous serving beam
- time spent in serving beam
- trends in the instance features (mainly RSRP and distance)

Beam Indexes

The current serving beam index is one of the more obvious features, freely available to the system and giving quite a lot of information on where in the system a UE is located. It is however a discrete semi-ordered feature where beams within the same sector generally share properties. An example of that can be seen Figure 5.2, where the correlation between the RSRP values of each beam is plotted. The different nodes and sectors are easy to find by looking for the yellow squares.

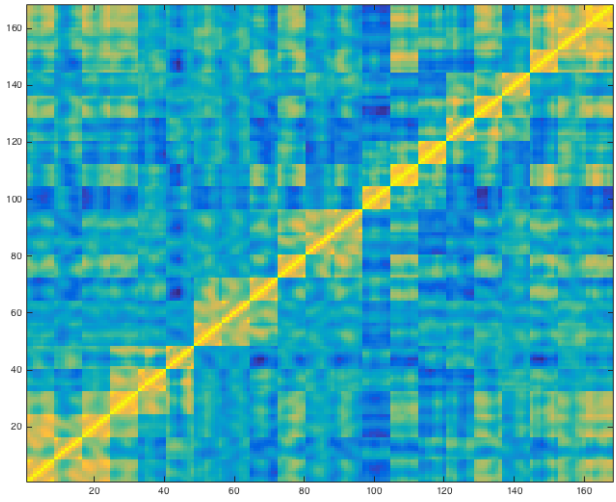


Figure 5.2: *RSRP correlation between all the beams. Yellow denotes a correlation close to 1, dark blue close to -1*

Destination beam index is any beam but the serving one. It is used as an actual feature in the FVV idea of multi-target learning, see Section 4.3.2, and as a convenient model naming in the ST. Just like serving beam index, destination beam index is a discrete semi-ordered feature.

Position and Distance

Exact position information was available in the simulator, which isn't very realistic that the final system would be able to acquire in a cheap way. To emulate a more realistic scenario normal distributed noise was added to both x and y values. The simulations didn't allow for a z-component so all UEs were at the same height at all time.

Instead of bouncing of the "walls" of the simulated area, see Figure 5.1, the simulation uses a wrap around. When moving out of the allowed area at one side they will be teleported to the opposite side.

Distance was computed using the perfect x and y positions and then noise was added on top of that ($N(\mu = 0, \sigma^2 = 30)$). Position was, when used, had more noise added to it as it was assumed to be more difficult to estimate, $N(\mu = 0, \sigma^2 = 100)$. The noise levels were chosen without any research on what could be reasonable values.

Reference Power

Serving RSRP is considered relatively cheap, though making do without it would be even better. Destination beam RSRP is collected from the simulator and used as target information when training and testing the learning algorithm. From a reality-cost perspective, these are the most expensive features to acquire (indeed with RSRP from all beams the whole beam selection problem would already be solved).

All values below -120 dBm were seen as unusable, as this is regarded as the lowest RSRP-level in LTE which data can be communicated on [2, table 9.1.2.1-1], and were truncated to -120 dBm. Above that threshold, a 32-bit float was used to represent the RSRP.

Channel Quality

CQI is a channel quality measure with a range from 0 (bad) to 15 (best), integer steps. In the simulated NR-system each beam can consist of either one or two codewords, each with their own CQI value. Used in the thesis is the sum of these codewords, making the valid range go between 0 and 30.

CQI is used during data transmission for link-adaptation and has thereby very low reality-cost. From a simulator-cost perspective it is quite cheap, though only available for the serving beam and when data is transmitted.

The primary use of CQI in this thesis was not as a machine learning feature, but rather as a beam switch trigger. A low CQI indicated that another beam probably should be better and triggered a beam switch in the simulation, see Section 6.1.2.

Speed

Using speed as a feature assumes the UE can report its speed or that the network can estimate it. Together with movement patterns and history-like feature it could perhaps be useful to help determine how the UE will behave in the future. That level of detail is difficult to achieve in this thesis. A UE with higher speed is

more likely to have a bad channel, or even bad channel estimations, which could perhaps be useful.

Targets

The learner tries to find the best destination beam, which in this study is the same as the beam with the highest RSRP. This makes two options available: either learn the RSRP of all beams and use that to determine which beam is best, or learn the index of the best beam directly. The first one leads to a multi-target regression problem and the latter one to a multi-class classification problem. These choices will be discussed more in Section 5.4.

History Features

For all the history features, there is a choice in how long and how much data is logged and how the values in a log are turned into features. One can either imagine some sort of filtering features, where several logged values are combined into one feature, or separate features where each history value makes up its own feature.

In this thesis, the current beam and the beam before that were considered as separate features. The time spent in those beams were also logged and considered as two separate features.

In total four history features were considered. This was the case because of two reasons: 1) straight-moving UEs are unlikely to introduce very realistic or long-lasting beam switch patterns 2) a scenario where UE move and request data continuously is a special case, chances are that most real connections won't make much more than one or two handovers during short update connections.

5.3 Pre-processing

Here relevant pre-processing is described. As will be described in Section 5.4 random forest was used, partly to avoid heavy pre-processing requirements.

Reference Power

RSRP values below -120 dBm were truncated to -120 dBm.

At first, serving beam RSRP was considered to be one feature, regardless of which serving beam it originated from. As serving beam index also was a feature it was assumed the learning algorithm should be able to find patterns anyway. Later on in the thesis RSRP was instead split into several features, as many as the number of possible input beams.

However, the split of feature enlarged the feature matrix but without adding any more information. This caused the problem of a feature matrix with missing values. To combat this some standard imputation schemes were considered and tested: constant, mean and median imputation.

Using the constant imputation, all missing values were replaced by a very low, somewhat arbitrary value, here chosen as -999 dBm. The mean and median

impute-values depend on the imputed feature's distribution and need to be learned from the training data. These values were then used to replace missing values both in train and test set.

Sample Selection

Simulations were done in a quite straight forward manner, sampling the feature values of all UEs at regular intervals. This created a lot of data, but mostly with sample points where the serving beam was good or at least no inter node handover was needed.

To allow for better control over the data and "simulate" different triggering scenarios, a sample selection scheme was implemented. Learning algorithms naturally depend on what type of data they are given. If trained on mostly intra node handover samples, it will prioritize such handovers even for samples where it is sub optimal. With the sample selection it was possible to construct different data sets, each set trying to describe a different handover triggering condition.

While quite useful, sample selection also created a lot of options making it possible to create very different data sets. Focus in this thesis was directed towards a sample selection where only samples fullfilling "$servingNode \neq bestNode$" were kept. This sample selection is in the results referred as "otherNode" (training and testing) or "Node-HO" (metrics computed on a subset of the testing data).

Data Division

Evaluating model performance on the same data that was used to train it is meaningless as then the algorithm's ability to generalize is not tested at all. Instead data is usually divided into several parts, one for training, one for testing and, possibly, one for validation. The validation-set is ideally locked away at the start of the project and only brought forth and used once at the end of the project. It is also popular to estimate a model's performance using cross-validation.

In this thesis, data was first divided into two groups: half of the seeds as training set and the other half as testing set. Cross-validation was used inside the training set to determine the meta-parameters, and the testing set was used to compute metrics closer to radio communication. No validation set was used due to lack of time.

5.4 Learning Models

In this section the system problem and available data is brought together into several supervised learning solutions.

5.4.1 Different Problem Perspectives

The traditional supervised machine learning methods are seldom used to select a subset of values from a larger set. To make the problem fit the supervised learning model, it was divided into two steps. First machine learning was employed to predict a quantity which makes it possible to rank the beams. Then, from the ranking, it was possible to select n candidate beams. Described below are the

three different approaches for the initial machine learning and ranking used in this thesis.

Regression Perspective

Predict RSRP for all beams, and rank per predicted RSRP. If the predictions are accurate enough this could have the added benefit of being comparable with the current serving RSRP to avoid activating beams worse than the current serving beam.

This can either be done as a multi-target regression, where the RSRP of each destination beam is considered as a target, or as several single-target regression tasks where one model is learned for each beam.

Classification Perspective

The target is the destination beam index with the highest RSRP. Using a classifier capable of estimating probabilities makes it possible to rank the different beams. This can be done as a multi-class classification.

Pairwise Preference Perspective

A bit different from the other two, which are two examples of pointwise ranking methods, is the pairwise preference ranking, see Section 4.5.1. Many binary classification tasks are created (one for each beam-pair), and in each case the goal is to predict which one of the two beams are the strongest (note, exact RSRP isn't needed in the node only the relation between the beams).

The following steps briefly describes the algorithm: 1) compute the probability of beam i being stronger than beam j using a binary classifier 2) do that for each beam-pair and store the result in a matrix 3) add the results in the matrix together into a final ranking.

5.4.2 Source Options

From the system's perspective, it's important to consider what data the learning algorithm has access to. One such aspect is whether to learn one model per serving beam or one model per sector/node or even several nodes at once. Data from beams will be stored and processed on a node level, making the first two options equally cheap from a node perspective. The last option requires node to share data, which is not covered in this thesis. In general, fewer models means less time spent on training and allow for the possibility of finding patterns between sources. However, this requires a learning algorithm capable of handling discrete, possibly unordered features.

5.4.3 Target Options

With multiple-output problems come several options on how to tackle them. The alternatives considered in this thesis are single-target (ST), feature vector virtualization (FVV), *multiple-output* (MO) and *pairwise preference* (PP). The evolved single-target approaches (single-target-stacking and regressor-chains) were not considered in this thesis as the implementation effort wasn't considered worth it compared to their estimated run-time.

These target options combined with the source options mentioned earlier can result in very many models. Let there be B_n beams in a node and B_d beams to predict for. There are then $P = 0.5 * B_d(B_d - 1)$ pairs of destination beams. Considering both the option for sources and the one for targets, the total number of models needed to be learned are displayed in Table 5.1 column two and three.

Table 5.1: *Number of models in each case*

	Source Options	
Target Options	node	beam
FVV	1	B_n
MO	1	B_n
ST	B_d	$B_d * B_n$
PP	P	$P * B_n$

Two limiting factors to consider are the number of training samples (more models require more data to train) and computational resources (more models take longer time to train).

There is also a difference in number of features used. This also determines the computational complexity of the problem. Suppose a ST-model, with "beam" as source option have d features in its feature vector. A FVV model with beam as source would require $d + 1$ features and with node as source, $d + 2$ features.

The models above, except PP, were built for regression. The MO-model was easily converted to a multi-class classifications problem with as many classes as there previously were targets. To differ between the problems setups the algorithms were called *multi-target regression* (MTR) and *multi-class classification* (MCC) respectively. The FVV model was thought unsuitable for classification and both FVV and ST would have required additional implementation to suit classification, which time did not admit. In the case of PP, only one model was build using regression input but only offering a final ranking instead of RSRP-estimates when used.

Important to note is the different demands on available training data. Currently the MO-models requires that a UE reports the RSRP of all beams each time it makes a report. This is doable in the simulator but will be expensive in a real network. It ought to be possible to adapt the MO-models to accommodate for the missing data, but it might be tricky and is not done in this thesis. The pairwise preference model is built around the idea that each report only contains information on a random subset of beams. The size of the subset size could vary for each sample, making the method quite flexible.

5.4.4 Choice of Learning Algorithm

Because of its resistance to messy data, and some previous positive experiences with it on Ericsson's part, random forest was chosen as the main model for this

thesis. The main benefits that were attractive were the reduced demand on pre-processed and continuous features, feature importance and a support for multi-target problems. Other learning algorithms than random forest was not included due to lack of time.

The regression trees tried to predict RSRP. The classification trees tried to classify each sample according to the best beam in that sample. Beam index was simply converted into a class index. Predicting class probabilities in the multi-class case made input and output to the two models have the same dimensions. Ranking was done by applying a max-function on the output matrix in both cases (maximizing either estimated RSRP or estimated "probability of being best").

Random Forest Limitations

Unfortunately, some unexpected quirks in the problem setup made it difficult to make use of all benefits of a random forest. Out-of-bag error was not useful as samples within a seed could not be considered i.i.d. Scikit-learn implementation left some to be wished for in terms of categorical and missing features.

5.5 Performance

Metrics and baseline models are described.

5.5.1 Problem Specific Metrics

Traditional machine learning metrics are important but fails to capture some of the radio communication aspects of the problem. To get a better picture beam-hit-ratio and RSRP-difference were introduced as metrics.

- **Beam-hit-ratio**: % of test samples where the best beam was selected.
- **Sector-hit-ratio**: % of test samples where the best sector was selected.
- **Node-hit-ratio**: % of test samples where the best node was selected.
- **RSRP-difference**: average difference between the RSRP of the best beam and the beam that the algorithm selected.

5.5.2 Baseline Models

During the first half of the thesis, the primary comparison point was a completely random selection of beams. The random selection was, unsurprisingly, not very a good method. Instead focus turned to two marginally more complex methods, Beam-Overlap and Average-RSRP.

To describe them mathematically the following notation will be used:

- Number of samples: N
- Serving beam index of sample n: s_n
- RSRP of sample n, beam i: $r_{n,i}$

- Indicator function: $I(x, y) = \begin{cases} 1 & x = y \\ 0 & x \neq y \end{cases}$
- Number of samples served by beam i: $S_i = \sum_n I(s_n, i)$
- Best beam for sample n: $b_n = \text{argmax}_i(r_{n,i})$

Average-RSRP

The Average-RSRP model is a simple predictor. For each serving beam training samples are used to compute the average RSRP of each destination beam, see Equation 5.1. The destination beams are then ranked according to the average RSRPs and the ranking used for beam selection.

$$AvgRSRP(i, j) = \frac{\sum_{n=1}^{N} r_{n,j} * I(s_n, i)}{S_i} \tag{5.1}$$

Beam Overlap

Beam overlap is easiest viewed as an area-overlap described as a ratio: 100% illustrates that both beams cover the same area and 0% indicates that they have no area in common. With this definition follows the problem of defining coverage area of a beam. In a system with quite many beams and were coverage area is narrow and highly influenced by surrounding buildings this is a bit cumbersome. Used instead in this thesis is an approximation of beam-to-beam overlap. By looking at the samples served by a particular beam, that beams overlap with another beam is computed as the percentage of time that the other beam was the best beam, see Equation 5.2.

$$Overlap(i, j) = \frac{\sum_{n=1}^{N} I(b_n, j) * I(s_n, i)}{S_i} \tag{5.2}$$

One problem with this approximation is that it allows for asymmetric overlap relations, *i.e.* it is possible that $Overlap(i, j) \neq Overlap(j, i)$. The good thing of the approximation is that the overlap values can be seen as the probability of a certain destination beam being the best beam. The destination beams are then ranked according to these probabilities and the ranking is used for beam selection.

5.6 Beam Selection

The first results, see Scenario A in Chapter 7 using all available samples without any sample selection, showed that the learned algorithm preferred choosing beams from the same sector. This behavior was detrimental to inter-node handover performance.

To combat that, a simple beam selection scheme was devised. On top of never

allowing the current serving beam (assuming a handover is needed), it limited the maximum number of beams sent from 1) other node sectors 2) serving node sectors. The limits were different for serving node sectors and sectors from other nodes. In the results the limits are described using pairs of numbers: *e.g.* (8,1) meaning max 8 beams per sector from other nodes than the serving node, max 1 beam per sector from the serving node. In the simulations used there where 8 beams per sector, so the numbers were between 0 and 8.

This method of limiting beams was only used in Scenario A, where both intra and inter node handover were considered.

6

Simulation Overview

In this chapter the data collection process is described.

6.1 Data Collection

Data was collected using an Ericsson simulator with moving UEs and base stations with many antennas. The propagation model used was 3GPP's *Spatial Channel Model*-model (SCM) [1]. The simulator ran for a while, updating serving beams as needed through a dummy algorithm (see Section 6.1.2) and at regular intervals logged data from each UE. The collected data was then post-processed in MatLab to construct valid feature and target matrices. After that the data was loaded into a python script where the learning was done.

The UEs started at random positions with a random direction and then moved in a straight line straight. Upon reaching the border, a map wrap-around was used allowing UEs and RF-signals to move out of the edge of the map on one side and then reappear at the other side.

Using a simulator made it possible to sample the ground truth for all features and UEs. This was used to simplify the problem significantly by allowing the assumption that training data was basically free. Training data was instead limited by RAM and CPU consumption in the training stage of the algorithms.

6.1.1 Simulator Parameters

Simulator Setup

- Propagation: SCM, urban-macro
- Carrier frequency: 4 GHz
- Duplex: TDD
- Inter Site Distance: 500 m
- Deployment (3gpp-case1)
  - 7 sites
  - 3 sectors/site
  - 8 antennas/sector
  - 1 beam/antenna = 168 beams
  - beam cover wrap-around at scenario edge
- UE properties:
  - always requesting data
  - static number of UEs
  - straight-moving (random start and direction) with wrap-around at scenario edge
  - 5 groups of 10 UEs, only speed differ between groups
  - speed: 6, 9, 12, 15, 18 m/s (only groups 1-4 used during learning*)
  - 2 antennas
- Beam selection dummy used in simulation
  - start to look for new beam if CQI is less than 8
  - stop looking when CQI is greater than 10
  - select the beam with highest RSRP

*=The fifth group, moving at 18 m/s, where excluded as that speed was seen as implausible in an urban macro scenario.

6.1.2 Simulator Considerations

iid samples

UEs moving around isn't always a nice thing. Samples from the same UE and roughly same position cannot be seen as independent. Slow moving UE (or just a fast sampling speed) will result in such samples. If not careful it is easy to end up in situations where similar/identical samples are used for both training and testing, making any result meaningless. This was solved by running the simulator with many different seeds and using some seeds for testing and some for training. Yet again some care needed to ensure that the same shadow map (*i.e.* simulator's statistical version of buildings) was used across all seeds.

Sample Frequency

The RSRP-logs were sampled at 4 Hz for each UE and beam pair, partly because of the sheer number of values and partly because of i.i.d. issues (samples more often than would be more similar and add little new information). Other logs were stored in their original format, sampled very often but not timed together with the RSRP-logs. When the logs were combined into feature vectors, the values closest to the RSRP timings were used. The resulting time-offset was less than 10 ms.

Training Data Catch 22

To function properly, the simulator needs a beam switch procedure. Ideally it would be the same as learned in this thesis, which however creates a catch 22: to learn a model data is needed and to produce data a model is needed. It should be possible to construct a bootstrap procedure were a learner first guesses, and then gradually retrains on the data produced by itself to eventually converge to a stable solution. That problem was however judge too large for this thesis. Beam switches in the simulator were instead triggered on low CQI and the simulator then automatically measured all MRSs and picked the best one as the candidate beam. A switch was then carried out if the candidate beam RSRP was 3 dB better than the serving beam RSRP.

The good thing using this procedure was that the learning did not need to be online. The bad thing was that the resulting data set was somewhat artificial, created according to rules no algorithm have access to.

6.2 Learning Implementation

The learning was done offline with the data collected from the simulator. Sample selection, see Section 5.3, made it possible to limit the training and testing to samples where performance was deemed more important.

The code handled pre-processing (truncating RSRP, split RSRP and sample selection) and the split into training and test sets. The search space for cross-validation was defined and then the actual learning was carried out by the scikit-learn version of random forest. After that the additional metrics were computed and plotted.

Cross-Validation Setup

A randomized cross-validation search was constructed using the built in models of scikit-learn. The configuration of meta-parameters was randomized inside set intervals and then cross-validation was used to estimate its score. This procedure was repeated several times, searching for the optimal set of meta-parameters. The search aimed to minimize MSE for regression models and the F1 (weighted-macro) score for classification models. See next page for details on the cross-validation used.

Main cross-validation setup

- Number of folds: 10
- Number of repetitions: 40
- random subspace size: maxFeatures

Cross-validation setup for PP

- Number of folds: 5
- Number of repetitions: 25
- random subspace size: one of $[1, \sqrt{maxFeatures}, maxFeatures]$

Meta-parameters considered in all models

- The number of trees: integers in range $[100, 900]$
- Tree depth control
  - min-sample-leaf: integers in range $[1, 30]$
  - min-sample-split: integers in range $[1, 30]$
  - max-depth: one of $[3, 25, 45, \infty]$

7
Results

The beam selection problem can be studied in many different angles and settings, using different data and/or different models. In this chapter five different setups, each with its own focus, are constructed and the results of them presented and examined.

7.1 Model Options

In Chapter 5 many choices and ideas were introduced. Here those choices are presented in Table 7.1.

Table 7.1: *Choices*

Category	**Options**
Sample Selection	no selection/otherNode[1]
Source Option	node/sector/beam
Models	FVV/MCC/MTR/PP/ST
Beam Selection	(8,8)/(8,1)/(8,0)

1: An idealized situation where the learner was trained using samples where it was known that $servingNode \neq bestNode$. This was used to simulate a case where the learner was built to perform well on inter node handover samples.

Added to these options are the possible features presented in the list below.

Features:

- serving beam index (only for sector and node source option)
- destination beam index (FVV only)
- serving beam RSRP (no split, constant, median, mean)
- distance to serving BS (perfect, noisy)
- coarse position (perfect, noisy)
- UE speed
- CQI
- time spent in serving beam

7.1.1 Scenarios

From the models and features five scenarios were constructed.

Scenarios

A Uses all captured data and studies differences in intra and inter node handover.

B Studies the effects of RSRP imputation in the MTR model. Inter node focus.

C Compares the different regression models: FVV/MTR/PP/ST. Inter node focus.

D Studies the difference between MTR and MCC. Inter node focus.

E Studies the difference between MTR and MCC using additional, possible expensive, features. Inter node focus.

An overview of the settings for each scenario can be found in Table 7.2.

Table 7.2: *Scenarios*

#	Sample Select	Source	Models	BS	Features
A	no selection	node	MO	**vs.**	distance or RSRP
B	otherNode	node	MO	(8,0)	RSRP (vs.)
C	otherNode	sector/beam	**vs.**	(8,0)	CQI, distance
D	otherNode	node/beam	MO	(8,0)	CQI, distance
E	otherNode	node/beam	MO	(8,0)	**vs.**

In the results below beam-hit-ratio and RSRP-difference are plotted as functions of the number of candidates beams. Beam-Overlap and Average-RSRP are plotted as purple and black dashed lines respectively.

7.2 Scenario A - Beam Selection

Goal

The goal of Scenario A is to illustrate the thesis's initial approach, training on all captured data and evaluating performance on all beams, and the problems related to that approach. The effects of beam selection, that is limiting the number of beams sent form certain sectors, are also investigated.

This scenario has access to all data captured from the middle node (see Figure 5.1), but draws randomly from that to avoid a too large data sets (RAM and run time constraints). It shows some insight in how beam selection works and provides a comparison point for the other scenarios which uses otherNode sample selection.

Result Summary

Most of the sample points have their best possible beam in the current serving node, and are considered as intra node handovers. This creates a learner that prioritizes intra node handovers. Beam selection can somewhat mitigate the focus on intra node beams by forcing the algorithm to select some beams from other nodes too. This is however not ideal a result in mediocre performance in both cases.

Setup

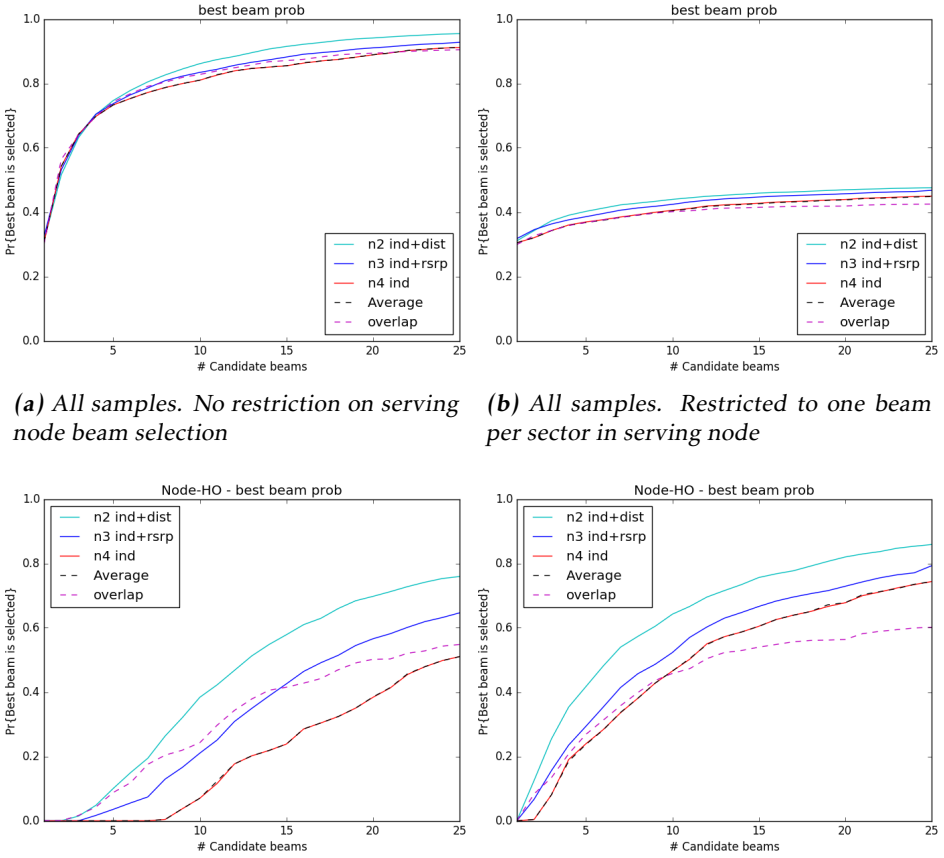
Table 7.3: *A - Scenario Settings*

Parameter	Setting	Comment
OtherNode	False	no restriction
Beam select	(8,8) vs (8,1)	no restrictions vs quite restricted
Train set size	3000	large, quite restricted
Test set size	6000 (965 otherNode)	large, quite restricted

Table 7.4: *A - Model Setting*

#	Color	Source	Target	Features
n2	cyan	node	MO	index, distance
n3	blue	node	MO	index, RSRP (median)
n4	red	node	MO	index

Results



(a) All samples. No restriction on serving node beam selection

(b) All samples. Restricted to one beam per sector in serving node

(c) Only node-handover samples. No restriction on serving node beam selection

(d) Only node-handover samples. Restricted to one beam per sector in serving node

Figure 7.1: *Beam-hit-ratio in top row and node-handover beam-hit-ratio in bottom row. Unrestricted beam selection in the left column and restricted beam selection in the right. Scenario A, node 4*

Performance measured across all samples, Figure 7.1a, looks good, while inter node handover performance looks very bad, Figure 7.1c. Further investigations shows that roughly 16% of the samples are considered as inter-node handovers. The learner prioritizes the majority class, which in this case are the intra-node handovers.

Notice the initial flat parts of the curves in Figure 7.1c. In that figure performance is measured on inter-node handover samples only, which means all candidates chosen from the serving node will be wrong. The initial flat parts shows

how difficult it is for the learner to identify samples as inter-node handover and how heavily it prioritizes beams from the serving node. This behavior remains somewhat in the restricted version, Figure 7.1d, but is notably less troublesome as it is restricted to only three beams from the own node.

Regardless of beam selection, distance and RSRP seem to be helpful when determining whether to make a intra or inter node handover, see Figure 7.1c and 7.1d.

Comments

It showed rather clearly how easy it is to get fooled by a seemingly good number or graph. Which samples that are used for training and evaluation are extremely important. The choice in this case was either to use beam selection to try get good performance on otherNode samples, or limit the training to such samples. The first one with a low reality cost, the latter with a high reality cost.

The results in this scenario, implicitly, hint at good performance in an intra-node handover case. With correlation between beams higher inside a node, see Figure 5.2, it is clearly plausible. This suggest that knowing when to make an intra-node handover and when to make an inter-node handover might be important for solving both problems.

Because of this scenario, latter models used the otherNode sample selection to train and test only on samples where another node was known to be better.

7.3 Scenario B - Impute Comparison

Goal

Investigate the usefulness of RSRP as a feature and the effects of different impute-strategies.

Setup

OtherNode sample selection is used to restrict training and testing to samples where it is known that another node is better.

In this scenario several different versions of RSRP is considered mainly using the MTR model. Only MO models were considered as they had the best run time (see Scenario C).

Result Summary

Unfortunately RSRP didn't help much at all. With the correct cross-validation it is not necessarily bad in regression but don't seem to have any positive effect on beam-hit-ratio or RSRP-difference. More comments on RSRP as a feature can be found in Section 8.2.2.

Setup

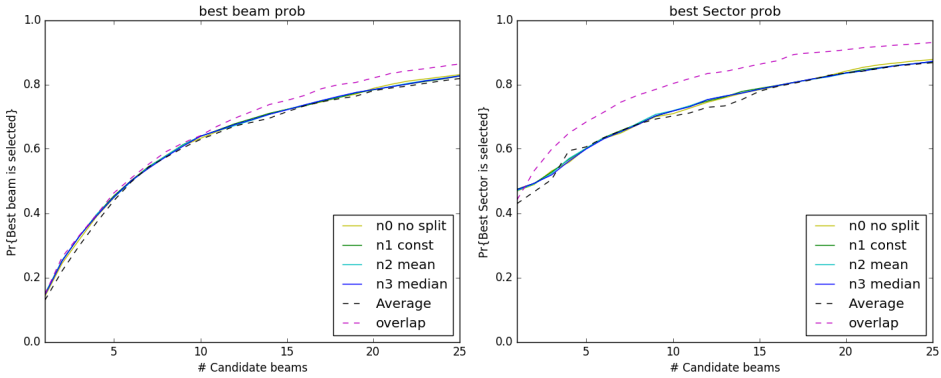
Table 7.5: *B - Scenario Settings*

Parameter	Setting	Comment
OtherNode	True	no restriction
Beam select	(8,0)	quite restricted
Train set size	3000	large, quite restricted
Test set size	3000	large, quite restricted

Table 7.6: *B - Model Setting*

#	Color	Source	Target	Features
n0	yellow	node	MO	index, RSRP (no split)
n1	green	node	MO	index, RSRP (constant)
n2	cyan	node	MO	index, RSRP (mean)
n3	blue	node	MO	index, RSRP (median)

Results MTR



(a) Beam-hit-ratio as a function of the number of candidate beams.

(b) Sector-hit-ratio as a function of the number of candidate beams.

Figure 7.2: *Beam and sector hit-ratio for scenario B regression models*

Both beam-hit-ratio, Figure 7.2a, and sector-hit-ratio, Figure 7.2b, looks similar to the Average-RSRP baseline model.

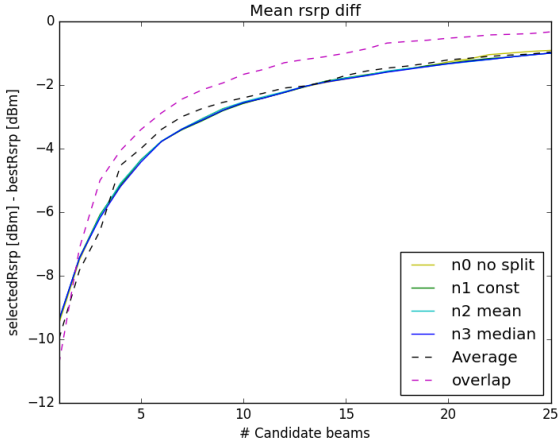


Figure 7.3: *Regression models. Mean RSRP-difference, as a function of the number of candidate beams.*

Mean RSRP-difference, Figure 7.3, seems to be on the low side, actually indicating a worse performance than the Average-RSRP baseline model.

A regression forest using only serving index would, except for the bagging procedure, reduce to the same as the Average-RSRP. Similarly does the Beam-Overlap

model corresponds to a classification forest using only serving index.

Being worse than the baseline in this case thus indicates that the learner performs worse with RSRP than without it.

7.4 Scenario C - Model Comparison

Goal

Investigate combinations of source and target options.

Setup

OtherNode sample selection is used to restrict training and testing to samples where it is known that another node is better.

Several different regression models, ST, FVV, MTR and PP are compared. As one of the merits of PP is that it is able to work with less information than the others, it only uses half of the RSRP-measurements in each sample, selected at random.

Only samples from one sector (eight beams) are used, as the long run time of some models made it unfeasible to run on whole nodes as in the other scenarios.

Result Summary

The ST-approach performed slightly better than the MTR versions, but almost 100 times slower. The FVV-approach was bad, both performance and time wise. PP, which only uses half the amount of samples that the other models use, performed surprisingly good when looking at Sector-hit-ratio, RSRP-difference and run time while lagging behind considerably in beam-hit-ratio.

Setup

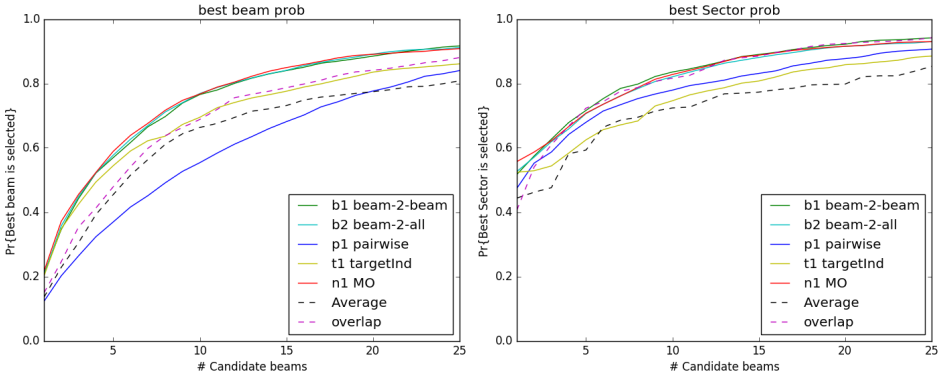
Table 7.7: *C - Scenario Settings*

Parameter	Setting	Comment
OtherNode	True	no restriction
Beam select	(8,0)	quite restricted
Train set size	1000	medium, very restricted
Test set size	2000	medium, quite restricted

Table 7.8: *C - Model Setting*

#	Color	Source	Target	Features
b1	green	beam	ST	CQI, distance, RSRP (no split)
b2	cyan	beam	MTR	CQI, distance, RSRP (no split)
p1	blue	sector	PP	index, CQI, distance, RSRP (median)
t1	yellow	sector	FVV	index, target index, CQI, distance, RSRP (median)
n1	red	sector	MTR	index, CQI, distance, RSRP (median)

Results



(a) *Beam-hit-ratio* ***(b)*** *Sector-hit-ratio*

Figure 7.4: *Beam and sector hit-ratio for different regression models*

Figure 7.4a shows the performance in a regression setting for some of the main source and target options. The ST and MTR approaches are best in terms of beam-hit-ratio, and will beat base line models (dotted lines). PP clearly suffers from the unfair training setup but manages to beat FVV in sector-hit-ratio. FVV show a sub-par performance despite using the same data as the other models.

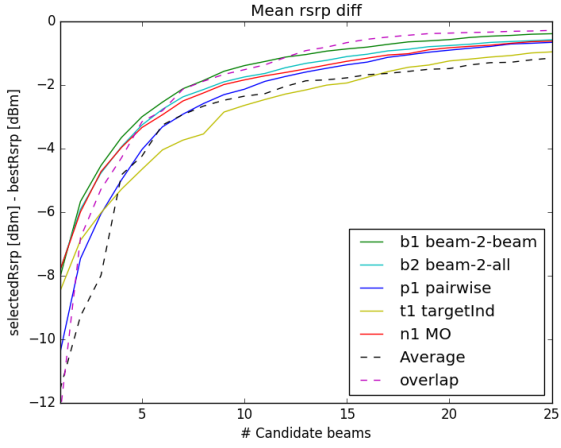


Figure 7.5: *Mean RSRP-difference as a function of the number of candidate beams.*

In terms of mean RSRP-difference the ST model takes a small lead. PP is good (considering its reduced data) while FVV-model still have trouble.

Table 7.9: *C - Regression run time (training+testing) and MSE-score*

#	Model	Run Time (min)	MSE	MSE top 25
b1	ST	1000	42.4	7.5
b2	MTR	12	42.1	7.6
p1	PP	8	N/A	N/A
t1	FVV	277	45.6	8.3
n1	MTR	12	41.9	7.7

Table 7.9, shows that the ST-approach and the FVV-approach are clearly inferior to the MO approaches in terms of computational complexity. With the amount of extra models ST and FVV need to be build (and cross-validate) this is hardly surprising, though the magnitude was larger than expected.

PP was surprisingly fast. The expectation was that it would be the slowest model, but with that expectation followed a less strict cross-validation procedure, see Section 6.2. The simpler cross-validation, combined with it being fed less data than the other models made it fastest of all these algorithms.

FVV slow run time is easily explained by the huge increase in samples, all which automatically have one feature more than the other models. It is also the only model with a notably worse MSE score.

7.5 Scenario D - MTR vs MCC

Goal

Investigate the performance difference between MTR and MCC.

Setup

OtherNode sample selection is used to restrict training and testing to samples where it is known that another node is better.

Previously regression models have been compared mainly with other regression models. In this scenario the most promising/realistic models of this thesis are compared.

The results in earlier Scenarios shows that MO-models offer the best runtime to performance trade-off among these models. It was also shown in scenario B that RSRP was not a very useful feature. In this scenario MTR and MCC models are put against each other only using serving beam index, CQI and distance to serving BS.

Result Summary

In general models built with MCC are better than MTR. Models built for all beams in the node perform slightly better than when there is one model per serving beam.

Setup

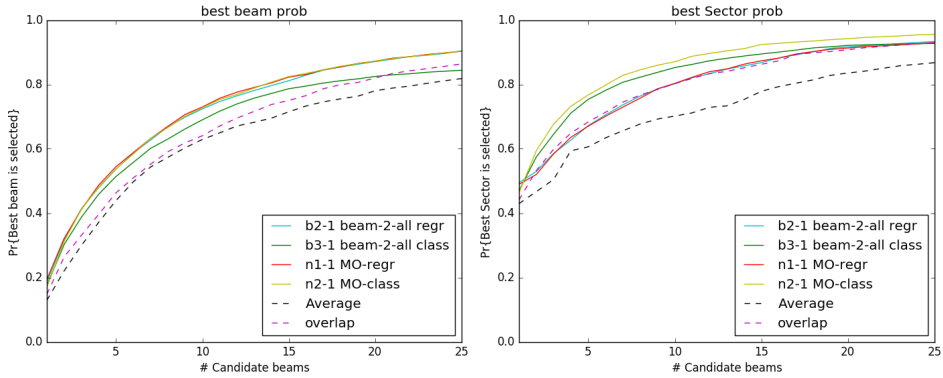
Table 7.10: *D - Scenario Settings*

Parameter	Setting	Comment
OtherNode	True	no restriction
Beam select	(8,0)	quite restricted
Train set size	3000	medium, quite restricted
Test set size	3000	medium, quite restricted

Table 7.11: *D - Model Setting*

#	Color	Source	Target	Features
b2	cyan	beam	MTR	CQI, distance
b3	green	beam	MCC	CQI, distance
n1	red	node	MTR	index, CQI, distance
n2	yellow	node	MCC	index, CQI, distance

Results



(a) *Beam-hit-ratio as a function of the number of candidate beams.*

(b) *Sector-hit-ratio as a function of the number of candidate beams.*

Figure 7.6: *Beam and sector hit-ratio for scenario D models*

The b3 model, using beam as source and MCC to predict for all targets, seems to perform slightly worse than the others in beam-hit-ratio, Figure 7.6a.

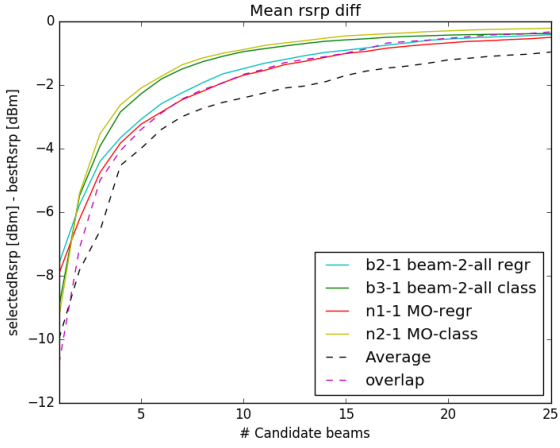


Figure 7.7: *Mean RSRP-difference as a function of the number of candidate beams.*

MCC based models seems to have an easier time both regarding sector-hit-rate and mean RSRP-difference.

Run time for all these models are quite similar: 20 to 35 minutes, depending on how many samples are used for training.

7.6 Scenario E - Feature Importance

Goal

Compute and compare feature importance for all features, including those with high reality-cost, and get an estimate of performance gain using those extra features.

Setup

OtherNode sample selection is used to restrict training and testing to samples where it is known that another node is better.

Previously, no "expensive" feature have been considered, but here those constraints are loosened somewhat. New features are x and y coordinate (noisy) and time spent in the serving beam.

Result Summary

The regression models achieve slightly higher beam-hit-ratio, while classification has a slightly better mean RSRP-difference.

Feature importance in regression is dominated by serving index and distance. Without noise added to position, x and y together sums up to over 80% of the importance. Feature importance in classification is not as clear, though distance and position are rated highly. In both regression and classification UE speed is given zero importance and CQI is generally given a low score.

Setup

Table 7.12: *E - Scenario Settings*

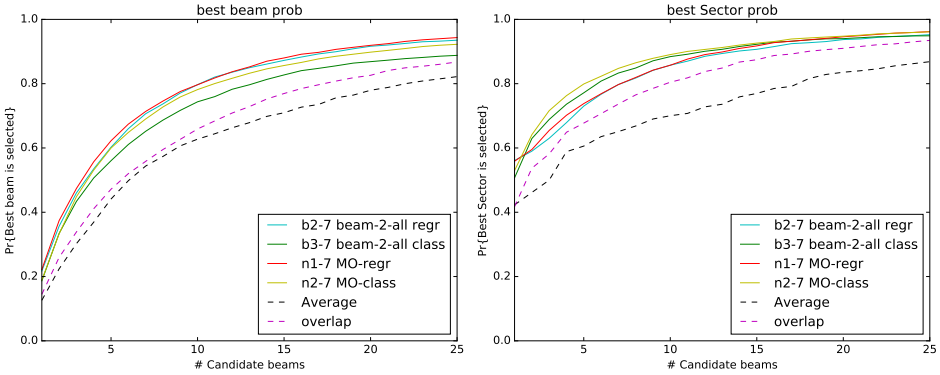
Parameter	Setting	Comment
OtherNode	True	no restriction
Beam select	(8,0)	quite restricted
Train set size	3000	medium, quite restricted
Test set size	3000	medium, quite restricted

Table 7.13: *E - Model Setting*

#	Color	Source	Target	Features
b2	cyan	beam	MTR	CQI, distance, dwell time, x-position, y-position, UE-speed
b3	green	beam	MCC	CQI, distance, dwell time, x-position, y-position, UE-speed
n1	red	node	MTR	CQI, distance, dwell time, index, x-position, y-position, UE-speed
n2	yellow	node	MCC	CQI, distance, dwell time, index, x-position, y-position, UE-speed

Results

First, similar graphs like in scenario D is presented. Then the feature importance of model n1 and n2 are presented. Good to remember is that this is the same models and setup as scenario D only allowing more features.



(a) *Beam-hit-ratio, as a function of the number of candidate beams.*

(b) *Sector-hit-ratio, as a function of the number of candidate beams.*

Figure 7.8: *Beam and sector hit-ratio for scenario E models*

With the extra features, node-MTR reaches the highest beam-hit-ratio and clearly outperforms the other models. The easiest way in this framework to increase performance is to lower the noise on position/distance.

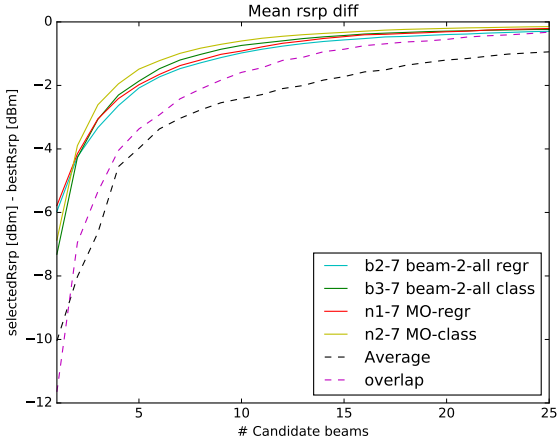


Figure 7.9: *Mean RSRP-difference, as a function of the number of candidate beams.*

With more feature all models get a better RSRP-difference. The node-MTR (n1)

manages to bridge the gap to node-MCC (n2). Much of this can probably be explained by diminishing returns, it gets more and more difficult to improve the model the closer it gets to perfect performance. As all models seems to level out on the same level, those last samples are probably very difficult to get right.

Table 7.14: *E - feature importance for n1 model with noise added to position*

Feature	**Importance (%)**
Serving beam index	37
Distance	23
x-position	15
y-position	13
CQI	6
Dwell time	5
UE speed	0.00

To get a feeling for the impact of the noise, the feature importance was also computed for a model with no added noise (unfortunately from a different run where previous serving beam index was used as a feature too, which in this particular setup is artificially similar to the true best beam index).

Table 7.15: *E - feature importance for n0 model without noise added to position*

Feature	**Importance (%)**
x-position	47
y-position	35
Serving beam index	13
Distance	4
CQI	1
Previous serving beam index	0
Dwell time	0
UE speed	0

As seen in the Table 7.15, the model makes many splits on position almost ignoring other features completely. This might be because of information overlap - position giving same information as distance, but for some reason the splits happens on position. It could also be an indication of how important position is in the underlying channel model used in the simulator.

Table 7.16: *E - feature importance for n0 model with noise added to position*

Feature	Importance (%)
Distance	24
y-position	19
x-position	18
Dwell time	16
Serving beam index	13
CQI	10
UE speed	0.00

Comments

Perhaps the most exciting, but also most unrealistic scenario. It shows considerably better performance for both MTR and MCC when compared to Scenario D. It however comes with the cost of many expensive features.

With a better positional estimate MTR can be very good indeed - almost ignoring all other features. Unfortunately it seems difficult to reach a state of "machine learning equal magic", predicting the correct beam 95+% of the time, even when such perfect position information is used. Best, at ten sent beams, is roughly 90% beam-hit-rate.

An common factor for models with more features is the large performance improvement on first-beam prediction, *i.e.* $n = 1$. These successful models also present a much wider range of possible top-1 outputs. A simple model will almost always predict the same ten beams as the top-1 beam while a complex model might have 40 different top-1 suggestions. It is possible that most of the added performance of these complex models are added to the top-1 case and the rest of the curve following along. If so, future learning tasks on this topic could very well focus on top-1 performance, confident that a good result there will give a good result when sending five beams.

7.7 Summary

These scenarios indicates the following results:

A The learner learns the majority "class" of the samples (intra /inter node handover), and though beam selection can be used to mitigate that it has some costs.

B RSRP seems to be too noisy, or have little to do with inter node handover. Simple imputation scheme do not help.

C In regression, the ST model seems best but extremely slow. MTR is only a small distance behind (which might even change by just adding more data), while several times faster. PP might be an interesting choice if ever trying the same thing with more restrictions on data collection.

D MCC is better than MTR, both preferably built with as many beams as possible as sources (the discrete serving beam indexes seems to not be a problem)

E With more positional information, performance increases. MCC seems still to have the lead.

The best models were the MCC model from scenario D and E, with their MTR counter-parts close behind. Below, in Table 7.17-7.19, the results for the best models are displayed from a different point of view than before. Rather than answer the question "how good is the model if allowed these many candidate beams?" they tries to answer "how many beams are needed to achieve a certain performance level?". Note that these numbers should be taken with a grain of salt. They need to be considered in proportion of the total number of beams and compared with each other.

In these tables two different MCC models, MCC and MCCextra, two different MTR models, MTR and MTRextra, and the Beam-Overlap baseline model are compared. The MCC and MTR are the same models as in Scenario D, while MCCextra and MTRextra are the same models (only considering models with node as the source) as in Scenario E. The main difference between the normal and extra models are the number of features they use (most importantly position or not).

Table 7.17: *Number of candidates beams required to achieve a certain beam-hit-ratio*

%	MCC	MTR	MCCextra	MTRextra	Beam-Overlap
50	5	5	4	4	6
60	6	6	5	5	8
70	9	9	8	7	13
80	14	14	11	10	17
90	25	25	21	20	N/A
100	N/A	N/A	N/A	N/A	N/A

Table 7.18: *Number of candidates beams required to achieve a certain sector-hit-ratio*

%	MCC	MTR	MCCextra	MTRextra	Beam-Overlap
50	2	1	1	1	2
60	3	4	2	3	4
70	4	6	3	4	6
80	6	10	5	7	10
90	13	19	13	14	17
100	N/A	N/A	N/A	N/A	N/A

Table 7.19: *Number of candidates beams required to achieve a certain minimum mean RSRP-difference*

dBm diff	MCC	MTR	MCCextra	MTRextra	Beam-Overlap
5	3	3	2	2	3
4	3	4	2	3	4
3	4	6	3	4	6
2	5	9	4	5	9
1	10	16	7	11	16

8 Discussion

In this chapter the thesis and its methodology are critically examined and ideas for the future presented.

8.1 Overall

The overall results were not as good as was initially expected. RSRP-difference looks good, whereas beam-hit-ratio was worse than expected. RSRP as a feature was a disappointment and, with few other features to use, it was difficult to get above the baseline models. Though, to be fair, it should be noted that sector/node-hit-ratio generally is quite good. If one is fine with correcting the beam placement after switching node, then good sector-hit-ratio might be enough.

Other than performance estimates, the thesis increases the understanding of the beam selection problem and how it relates to machine learning. A comparison between ST-models, FVV-models and MO-models resulted in a performance-tie between ST and MO, with the latter being much more computationally efficient. Future work along this line could focus on the multi-target nature and how to address that more effectively.

Relaxing the constraints on features, *i.e.* allowing distance and possibly even position, improves the results a lot. Future work in this area could focus on enabling the use of positional features (*i.e.* distance and position).

Another area which could use improvement is more clearly defined problem boundaries:

- 1) what kind of samples should performance be measured on?
- 2) how are these samples generated?

- 3) how is performance measured?
- 4) what features are (cheaply) available to the learner?

This thesis included the last two questions from the start and stumbled into the first two along the way, but failed to acknowledge their importance. Comparing to the desired thesis outcome, from Chapter 1:

"This study on candidate beam selection should result in:

- an method of transforming the problem to supervised learning
- suggestion on features and their importance
- performance metrics and comparison to simpler methods
- insight into the trade-off between number of candidates beams and probability of finding the best beam."

Transforming the beam selection problem to machine learning can be done in several different ways as shown in the thesis. The most straight forward is a multiple-output learning problem then used for pointwise ranking. Features used were mainly beam index, RSRP, CQI and distance. Not much feature engineering was considered, only RSRP was investigated in terms of splitting and imputing, which perhaps is something that could be investigated further.

Performance metrics that fit the problem and made it easy to translate to radio communication performance was introduced. These could be further refined and their connection to machine learning metrics further studied. Average-RSRP and Beam-Overlap model were used as baseline models. Both baseline models performed quite well when considering their extremely low computational cost.

The number of candidate beams needed for adequate performance were a bit higher than the initial expectations, almost requiring 10-15 beams (10% of all beams) to be reliable (assuming 90-95% beam-hit-ratio combined with a low average RSRP-difference counts as reliable).

8.2 Method

A surprising amount of time went into pre-processing, cross-validation and metrics. The lesson learned is that all of these steps matters, and should be considered very seriously early on.

8.2.1 Studied System

The system studied is interesting, but simulated. It is clearly so that the extent these results are applicable in the real world very much depend on a lot of details in the simulations. It is possible (maybe even likely) that real-world performance ís worse than these results (in absolute performance both for learned models and base line) as several measurements have been idealized in a way not possible in the real world (*e.g.* RSRP, position and distance noise). On the other hand, a real world scenario might offer different kinds of patterns, movement along

roads etc., which might be possible to make use of. For example, speed was always considered as "unimportant" according to the random forest importance measurement. This could be because of the underlying propagation model used in the simulator, or maybe speed is meaningless when it comes to making RSRP-predictions.

The use of otherNode sample selection and the way that the current simulator changes beams was probably not ideal. It connects to question two above (see Section 8.1). This combined way of generating handover-samples is a sub-optimal construction which results in potentially confusing data set. In many of those samples, serving beam RSRP is not that far away from the best beam. Roughly 75% of the samples have a RSRP-difference between the serving and best beam of less than 3 dB. On average there are only 4 beams that are better than the current serving beam when using otherNode sample selection.

Handover zones and selection statistics have not been investigated that much. My feeling is that this kind of learner will pick one or two good beams from each sector that act as "handover points", which all samples going to that node are handed over to. This might be due to the layout of the map, simply these combinations of beams being the strongest for this kind of samples. If the new node is capable of quickly reassigning the incoming UE to a new beam this might not be a problem. Otherwise some beams will be used very much and some not at all. Though it feels like the purpose of many beams disappear if not all of them are used equally much. In the case of some beams always being better than other beams some beams could clearly be removed, or all beams could be more focused in area.

8.2.2 Data

From a machine learning perspective the use of a simulator gave an almost unlimited access to data, which was convenient. Of course this data wasn't without its own faults. It is for example difficult to estimate the impact of the dummy beam switch procedure and non-i.i.d. samples. A clear definition of conditions for interesting samples, a more random data collection and only use instant features could perhaps help with that.

Problem Imbalance

Imbalanced machine learning problems are difficult to solve, as the models will favor the majority class and the metrics might look good even though the algorithm performs poorly on important minority classes.

The beam selection problem studied is imbalanced in several ways. By the very definition of the problem it is known that only a couple of beams will be good. Thus training a MCC-model for one node will result in an imbalanced machine learning problem where some classes have zero positive samples and other classes very many positive samples. This imbalance is inherent to the problem, but makes machine learning much more difficult to practice and evaluate.

The way simulations and source beams are handled is also important. If, by

chance or design, there are more samples from some serving beams, these beams' candidates will be considered more important to the algorithm. The imbalance makes the learner favor popular beams over other, meaning they will be chosen first not because they are good but because they are used a lot. Ideally, input to the algorithm should have an equal number of samples per possible serving beam, and (if possible) an equal number of samples for each possible best beam.

The natural imbalance of the problem was first seen as something potentially good, as it is information about the problem and thus maybe possible to use in some way. This was in hindsight quite naive and something that should have been investigated in much more detail. In traditional multiple-output machine learning, only one candidate beam would be allowed. Performance metrics in the $n = 1$ -case clearly show that the learner struggles.

By allowing n candidate beams to be sent, much of this imbalanced is hidden and performance seems good anyway. However, traditional machine learning tools and metrics are much more difficult to use and make sense of.

RSRP

RSRP was a disappointment as a feature and when looking into why, more problem with it was noticed. In the thesis it is assumed that 1) RSRP is estimated using a perfect channel estimation 2) is represented as a float and 3) that all values below -120 dBm are reported as -120. Point 1 and 2 are true in the simulator. The -120 dBm threshold comes from minimum strength required for data transmission in LTE. In LTE, RSRP measurements are reported in the range -140 dBm to -44 dBm with 1 dB resolution (total of 98 levels) [2, Section 9.1.4].

All these factors combined makes LTE-like RSRP almost a whole new feature, probably with more noise and even less information than the version used in the thesis.

It could perhaps be considered feasible and beneficial to measure more than one beam and then run the learning algorithm, if the number of extra beams are few. However, quick tests of this concept (in the thesis framework) did not yield any better result.

The only positive benefit of RSRP in this thesis was in Scenario A where it helped the learner to decide on whether to make inter or intra node handovers. Possibly it is a great feature when asking the question: inter, intra or no handover at all?

8.2.3 Learning Models

Random Forest

The random forest model offered a lot of "nice-to-have" in terms of feature preprocessing and feature importance. Both should not be underestimated, but it is possible that it comes with a cost of decreased accuracy. Though not the focus of this study, other machine learning models should also be investigated in order to find the one that fits the problem best.

Learning Framework

The idea to first score each beam, then rank and then select beams feels like a good problem break down, though some improvements ought to be possible to make.

The focus on inter node handovers was added quite late, first wanting an algorithm able to preform in all nodes and on all samples. Because of that the software design was built around running the regression for all beams, even beam from the own node. In a strict inter-node handover case this is unnecessary, and will possibly mislead the learner to spend resources on those beams. In the current implementation source node beams are discarded in the beam selection, which might be too late. A better software structure would allow the learner to separate source beams and target beams completely. This would allow the algorithm to spend its focus on predicting beams that it actually is allowed to used later on.

The beam selection might also be improved on. Instead of blindly picking the top n beams, a learner could try to look at the number of beams originating from the same sector/node among the top 30-40 predictions and base its choice of beams on that instead. It could for example pick the three most likely sectors and activate all beams in those sectors.

Metrics

The regression metric used, MSE, is simple, but in hindsight not ideal. All errors are not equal in severity and shouldn't be treated equally. An error of -10 dB on a bad beam (*e.g.* estimated -120 dBm and target was -110 dBm), is not so much of a problem as an error of +10 dB (*e.g.* estimated -100 dBm and target was -110 dBm). In the former case the beam will be ignored, while in the latter case it might end up as a (quite bad) candidate. Similarly, a large negative error on a really good beam means that it will never be considered as a candidate.

Another error model than MSE could be used, for example punishing errors differently depending on the true RSRP of the beam. However, such an error will be more difficult to analyze and to find an optimal solution for.

In classification, metrics also struggles with many imbalanced classes. Perhaps a class concatenation could be done. The objective would be to reduce the number of targets/classes by concatenating two or more destination beams into one target/class. This could either be done by concatenating beams close in index, or based on Beam-Overlap.

Always looking for the best possible beam/sector was also partly a mistake. Though easier than defining a more loose metric, it is very difficult to achieve. One possible way to make it easier would be to define an RSRP-interval around the best beam, and consider all beams inside that interval as good enough.

Cross-Validation

Scikit-learn computes its cross-validation as an average of the score in each fold, more or less a macro-average (see Section 4.3.3).

This is not the only way to compute cross-validation metrics. In [11], Forman and Scholz study the F1-metric and the effect of cross-validation. They conclude that it with very imbalanced classes, or when the proportion of true/negative samples vary greatly between folds, is best to compute F1-score using the sum of tp, fp, fn across the folds, which is a micro-average instead. They also point out that such situations are much more likely to occur in multi-output classification with many targets or multi-class classification with many classes.

8.2.4 Software

Structure

The division of software into three separate parts was not made by design, but rather by surrounding circumstances. Not tampering more than necessary with simulator code was decided early on, and felt like a good decision even in hindsight. The choice of using python for machine learning was also made quite early based upon Ericsson supervisor's experience with scikit-learn as an easy to use library. The third part, in Matlab, was created as the simulator output was in .mat-format and some useful mat-scripts provided by Ericsson could be used to repackage it. The resulting design enabled a reduction of unnecessary data in the python part by repackaging it in the matlab scripts. However, some computations and choices were made in the matlab part that in hindsight were more suitable for the python part. This resulted in issues were errors were noticed in python, which originated from matlab scripts, demanding both scripts to be run again. This worked fine for a while, but when the data amount increased each of the three parts took large amounts of time: one to three days for the simulator, one-two hours for the matlab-script and one to 48 hours for the python script. A complete re-run of all involved components and evaluation of results could easily take a week, which in hindsight was too inefficient.

The matlab script today serves the following functions: combining many simulator logs into one without running out of ram or disc space, only loading necessary data in python to avoid running out of memory while training, feature engineering. If the last one is moved to the python script and the matlab part redesign to focus on the first two and an added condition of being simple and fast, then things would have been much easier.

Scikit-learn

Scikit-learn offered an easy road into machine learning. It provides many examples and quite good guides for how to use the standard functions. However, as mentioned earlier in the thesis, its implementation of random forest did lack in some details. Details which most problems can do without, but that with this setup would have been nice to have. It is not that I think that the overall results would change much correcting these things (categorical features, missing feature values), but as I have been wrong before regarding which details that matters it would have been nice to be able to exclude these particular details from the list of possible errors.

8.3 Future Work

Regarding the choice of learner I would probably use something capable of predicting multiple targets from one feature, but with more accuracy than a forest. This is of course made more difficult by the categorical features.

More important than algorithms is however features and pre-processing. The imbalance needs to be investigated, as it probably is one of the main reasons to why few candidate beams fares so poorly. Being able to use some more good features ought to help too. Enabling positional features, either straight up coordinates or distance or something similar, should improve the result of any learner.

One idea is to run prediction algorithms on data collected in LTE-networks. In theory everything presented here should be possible to do in such a network too with the added benefit from real-world data, noise and performance. With the frequent CRS signals to act as ground truth it should be almost as easy as in the simulator to get the data needed.

Important, and maybe more doable, is to setup a (machine learned?) algorithm that determines if to make a handover or not, and after that possibly an algorithm deciding whether to make a node-handover or not.

Alternative Methods

During my work I stumbled upon some different ideas. Two of them modeled the problem as an imputation task. Suppose that you are given a couple of RSRP-measurments arranged in a matrix with many "missing values". The goal of imputation is to estimate all the "missing values". One proposed solution is similar to the Netflix prize, using truncated SVD to approximate the missing RSRP values. The other uses the missForest algorithm to iteratively predict the missing data.

Bibliography

[1] 3GPP. Spatial channel model for multiple input multiple output (mimo) simulations (ts 25.996), . URL https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=1382. Cited on page 39.

[2] 3GPP. Evolved universal terrestrial radio access (e-utra); requirements for support of radio resource management (ts 36.133), . URL https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2420. Cited on pages 31 and 66.

[3] 3GPP. Evolved universal terrestrial radio access (e-utra); physical layer; measurements (ts 36.214), . URL https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2428. Cited on pages 11 and 12.

[4] 3GPP. Evolved universal terrestrial radio access (e-utra); radio resource control (ts 36.331), . URL https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2440. Cited on page 6.

[5] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006. ISBN 0387310738. Cited on page 15.

[6] Hanen Borchani, Gherardo Varando, Concha Bielza, and Pedro Larrañaga. A survey on multi-output regression. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 5(5):216–233, 2015. Cited on pages 19, 20, and 21.

[7] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001. Cited on page 22.

[8] Erik Dahlman, Stefan Parkvall, and Johan Skold. *4G: LTE/LTE-advanced for mobile broadband*. Academic press, 2013. Cited on page 6.

[9] Ericsson. 5g energy performance, Apr 2015. URL http://www.ericsson.com/res/docs/whitepapers/wp-5g-energy-performance.pdf. Cited on page 1.

[10] Ericsson. 5g radio access, Feb 2015. URL http://www.ericsson.com/res/docs/whitepapers/wp-5g.pdf. Cited on pages 5 and 7.

[11] George Forman and Martin Scholz. Apples-to-apples in cross-validation studies: pitfalls in classifier performance measurement. *ACM SIGKDD Explorations Newsletter*, 12(1):49–57, 2010. Cited on page 68.

[12] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*. Springer series in statistics Springer, Berlin, 1 edition, 2001. Cited on pages 17, 19, 23, 24, and 25.

[13] Eyke Hüllermeier, Johannes Fürnkranz, Weiwei Cheng, and Klaus Brinker. Label ranking by learning pairwise preferences. *Artificial Intelligence*, 172 (16):1897–1916, 2008. Cited on page 26.

[14] Sotiris B Kotsiantis, I Zaharakis, and P Pintelas. Supervised machine learning: A review of classification techniques, 2007. Cited on page 24.

[15] Erik G Larsson, Ove Edfors, Fredrik Tufvesson, and Thomas L Marzetta. Massive mimo for next generation wireless systems. *IEEE Communications Magazine*, 52(2):186–195, 2014. Cited on pages 5 and 7.

[16] Henrik Linusson. Multi-output random forests. 2013. Cited on pages 19 and 25.

[17] Tie-Yan Liu. *Learning to rank for information retrieval*. Springer Science & Business Media, 2011. Cited on page 26.

[18] Gilles Louppe. *Understanding Random Forests: From Theory to Practice*. PhD thesis, University of Liege, Belgium, 10 2014. arXiv:1407.7502. Cited on pages 15, 22, 24, and 25.

[19] Stefan. Ericsson Research Parkvall. Release 14 – the start of 5g standardization. URL https://www.ericsson.com/research-blog/lte/release-14-the-start-of-5g-standardization/. Cited on page 5.

[20] Marina Sokolova and Guy Lapalme. A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, 45(4):427–437, 2009. Cited on pages 15 and 19.

[21] Eleftherios Spyromitros-Xioufis, Grigorios Tsoumakas, William Groves, and Ioannis Vlahavas. Multi-target regression via input space expansion: treating targets as inputs. *Machine Learning*, pages 1–44, 2012. Cited on pages 19 and 20.

[22] Daniel J Stekhoven and Peter Bühlmann. Missforest—non-parametric miss-

ing value imputation for mixed-type data. *Bioinformatics*, 28(1):112–118, 2012. Cited on page 19.