University of Padua

Department of Information Engineering

Master Degree in ICT for Internet and Multimedia

Master's thesis

Machine Learning Techniques For In-Region Location Verification In Wireless Networks

Supervisor

Prof.

Stefano Tomasin

Candidate

Gabriele Ruvoletto

Academic year 2018/2019

Alla mia famiglia

Abstract

In-region location verification (IRLV) aims at verifying whether a user is inside a specific region, e.g., an authorized area where specific services can be provided. We proposed an IRLV system that verifies the position through the estimated features of the wireless channels connecting the device to the Base Stations (BSs) of the IRLV network. Since the channel feature statistics is usually not available we apply machine learning (ML) techniques for implement IRLV.

Also, we propose ML techniques that can be used by an attacker located outside a specific region in order to break the IRLV system and let the network believe that he is inside this region.

Contents

1 Introduction **9**

2 System Model **13**

2.1 RSS Channel Model 13

2.2 Angle Channel Model 15

2.3 Correlation Model 16

3 IRLV And Attacks By ML **19**

3.1 Artificial Neural Network (ANN) 22

3.1.1 Multilayer Perceptron 24

3.1.2 ANN With Spline Activation Function 31

3.1.3 Autoencoder 36

3.2 Support Vector Machine (SVM) 37

3.2.1 Support Vector Classification (SVC) 38

3.2.2 One Class SVM 39

4 Defense And Attack Methods **41**

4.1 Defence Strategies 41

4.1.1 Direct Method 42

4.1.2 Ranking Method 42

4.2 Attack Strategies 43

4.2.1 Passive Attacks 44

4.2.2 Active Attacks 45

5 Numerical Results **51**

5.1 Defense Implementations 53

5.2 Attack Implementations 60

6 Conclusions **65**

Appendices **67**

A Simulation Parameters **69**

List of Figures

2.1 Representation of the matrix Σ. 17

3.1 Block scheme for C. 21
3.2 Block scheme for R. 21
3.3 Block scheme for G. 21
3.4 Example of ANN with two hidden layers. 22
3.5 A simple representation of the neuron j of the layer t. 23
3.6 A simple MLP with 3 features in input, one value in output and only one hidden layer. 25
3.7 Bound for realizations with 5 BSs. 29
3.8 Results of different dimension networks with train set of $M_{\mathcal{S}}$=20000 samples. 32
3.9 Two plot of CR cubic interpolation. 34

4.1 Schematic division of the ordered data set returned by $\mathcal{R}$. 43
4.2 SIE model with $B = 4$. Each $\mathcal{R}$ is the same in all the graph. 43
4.3 Block scheme of RA. 45
4.4 Block scheme of PRA. 45
4.5 Block scheme of RandPA. 46
4.6 Block scheme of PTRA. 47
4.7 Block scheme of RTPA. 48
4.8 Block scheme of RPRA. 48
4.9 Block scheme of KTPA. 49
4.10 Block scheme of KPRA. 49

5.1 Map A with 10 BSs. 52
5.2 Attenuation map including path-loss and shadowing, with the BS positioned at the center. 53
5.3 Map of arrival angle at left and map of departure angle at right. 54
5.4 DET for MLP-IRLV with different number of angles features. 55
5.5 DET for MLP-IRLV with different attenuation features. 56
5.6 DET for MLP-IRLV with attenuation features, angles features and both types. . . . 56
5.7 DET for SVC-IRLV on P+S channel with different Kernels. 57
5.8 DET for SVC-IRLV on P+S+F channel with different Kernels. 57
5.9 DET of different ANN for defense P+S attenuation channel models. 59
5.10 Two trained Spline functions. 59
5.11 DET for SVC-IRLV and MLP-IRLV on data collected by MOMENTUM project [1]. 60
5.12 Comparison of passive attacks on MLP-IRLV. 61
5.13 Comparison of active attacks on MLP-IRLV and SVC-IRLV. 62
5.14 Comparison of passive attacks on MLP-IRLV with different value α_{ring}. 63
5.15 Comparison of active attacks on MLP-IRLV and SVC-IRLV with different value α_{ring}. 64

List of Tables

A.1 Value of parameters for RSS Channel model. 69
A.2 Value of parameters for Angles Model. 69
A.3 Value of parameters for the MLP applied on the generated values. 70
A.4 Value of parameters for the MLP applied on real values [1]. 70
A.5 Value of parameters for the ANN with Spline activation function. 70
A.6 Value of parameters of the implemented AE. 71

Chapter 1

Introduction

The location information is a set of measures that define the position of an object respect a reference system. The most widespread method to obtain the location information is by meas of Global Positioning System (GPS) [2], i.e., a satellite-based positioning system that uses the time of connection between an User Device (UD) and at least three satellites to calculate the UD position. The information obtained from the localization system can be used in many applications, such as networks of sensor [3], [4], and global position encryption [5]. The main problem of these applications is that, without verification, the location information can be easily manipulated by hardware/software alteration of positioning system or by spoofing the GPS signal from outside the UD. For these reasons it is necessary to use a system for location verification, able to verify if the location information has been tampered. The model that we will present aims to verify the UD position by wireless channel between these devices and different BSs, i.e., groups of antennas that are able to communicate wirelessly with the UDs. A simple model can be found in [6]. In this case the authors use the received signal strength (RSS) to estimate the distance, knowing the spatial attenuation law between the UDs and the BSs.

A system able to verify if a UD is in a specific position is called single location verification (SLV) system. Instead, a system able to understand if the UD is in a region or in another is called IRLV system.

In the past works SLV was implemented comparing the actual value obtained from the wireless communication with the value obtained by a legitimate user that was in the same position. An example of that can be found in [7], where the obtained information is the channel impulse re-

sponse, affected by noise with known statistics. Another example can be seen in [8], where the noise statistics are unknown and a learning strategy is implemented.

Also for IRLV some solution have been proposed. In [9] and [10], it is calculated the value of the distance of the UDs with respect to the BSs by repeated exchanges of packets. Instead, in [11] the value of ultrasound signals is used, because with this type of signals the RSS value is more sensitive to the position. In [4] a solution based on a geometric approach from the signal received by the UD was proposed.

The aforementioned solutions only consider a simple channel model without the effects of fading and shadowing, two important aspects that statistically modify the RSS value. Some other IRLV system [5], are implemented in specific environments that can not be generalized for different conditions. What we want to do is give a IRLV model that can be applied in the largest possible number of different conditions and environments.

Also, we proposed some new attacks at this model. Some attacks have been already done in [10], [12], where the attacker UD wants to identify a false location, or in [13],[14], where the attacker UD modifies the power of the signal sent to be localized by the IRLV in another position. Our implemented attacks try all the two strategies on our proposed IRLV.

In this thesis, we will use some ML algorithms on the features extracted from the wireless channel for implement IRLV and attacks on this, because, the models implemented in this way can be applied to as many situations as possible.

The system used in our test is presented in the Chapter 2. Instead, the ML algorithm and their application for the creation of the IRLV and attacks are described respectively in Chapter 3 and Chapter 4. Finally the numerical results are presented in the Chapter 5 and the conclusions in the Chapter 6.

In Chapter 3, the theoretical aspects has been used in "S. Tomasin, A. Brighente, F. Formaggio, and G. Ruvoletto, *Physical-layer location verification by ML*, book chapter of Machine Learning for Future Wireless Communications, Ed. Fa-Long Luo,J. Wiley Sons, Chichester, UK, 2019." Instead all attack models presented in Section 4.2 and the results presented in Section 5.2 have been published in "A. Brighente, F. Formaggio, G. Ruvoletto and S. Tomasin, *Ranking-Based*

Attacks to In-RegionLocation Verification Systems, IEEE International Workshop on Information Forensics and Security, Delft, Netherlands, 2019".

Chapter 2

System Model

We consider a wireless network with N_{BS} BSs spread on space, that cover a zone A. The BSs, with height h_{Bs}, communicate with the UDs that in our case, are devices with one or more antennas positioned in the space A.
We propose a IRLV system where the Server uses BSs to understand if a specific UD is trasmitting for a authentication zone A_0, sub region of A, or from the complementary region $A_1 = A \setminus A_0$. To do so, IRLV exploits the position dependency of the channel features between a specific UD to all the BSs. These features can be:

- Single RSS value at different BSs.
- Couple of angles (arrival and departure) at different BSs.

The signal sent by the UD to the BSs is a pilot signal with fixed power, known from the BSs, from which, observing the value of the signal at the receiver and the angles of arrival and departure, the features are extracted.
In the next subsection we define two channel models used to obtain the value of these features and a possible correlation model applicable at these channels models.

2.1 RSS Channel Model

In our implementation the value of RSS is calculated starting from a pilot signal sent by UDs to all the BSs through a noisy channel, modeled as in [15]. The value of the RSS is directly modified

by the attenuation effect on the channel and can be obtained, if the UD sent a pilot message with the power P_{tx}, at the n-th BS, as

$$P_{rc}^{(n)} = \frac{P_{tx}}{g^{(n)}}, \tag{2.1}$$

where $g^{(n)}$ is the value of channel gain including the effect of path-loss, shadowing, and fading. Assuming a Rayleigh model for fading we have

$$g^{(n)} \sim N(0, \alpha_n^2), \tag{2.2}$$

where $N(0, \alpha_n^2)$ is a Gaussian random variable with mean 0 and variance α_n^2. Since $g^{(n)}$ is a random variable, the fading effect is unpredictable and produces an attenuation that changes continuously over time. Every $g^{(n)}$ of the same pilot signal with respect to all the BSs on the space are collected in one only vector $\boldsymbol{g}$ of dimension N_{BS}.

The value α_n^2 is obtained, due to shadowing effect, from

$$(\alpha_n^2)_{dB} = (P_l^{(n)})_{dB} + s, \tag{2.3}$$

where s is the value in dB of the shadowing attenuation and $(P_l^{(n)})_{dB}$ is the path-loss in dB.

The shadowing attenuation is modeled in dB as a Gaussian distribution with zero mean and variance $\sigma_{s,dB}$. It is due to the reflection of the pilot signal on building and static objects positioned between the UD and the BS. For this reason, s depends on the position and has a similar value (for the same position) for long periods. The shadowing correlation between two signals from two different UDs at position $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ to the same BS, is founded by a correlation function. This function can be defined as

$$\text{corr}(\boldsymbol{p}_i, \boldsymbol{p}_j) = \sigma_{dev} e^{-\frac{l(\boldsymbol{p}_i, \boldsymbol{p}_j)}{d_c}} \tag{2.4}$$

where $\boldsymbol{p}_i$ and $\boldsymbol{p}_j$ is two generic position,σ_{dev} is a standard deviation, function $l : \mathbb{R}^2 \to \mathbb{R}$, starting from two points on the space, returns the Euclidean distance and d_c is the decorrelation distance. In our application of 2.4 the value of $\sigma_{dev} = \sigma_{s,dB}$.

The path-loss effect, instead, models the average attenuation of the signal over the transmit-receive distance d. It is different for Line of Sight (LoS) and non-LoS condition an can be written as [16]:

$$(P_{l,LoS}{}^{(n)})_{dB} = 20 \cdot \left(\frac{f 4\pi d_n}{c_{light}} \right) \tag{2.5}$$

$$(P_{l,non-LoS}^{(n)})_{dB} = 40(1 - 4 \cdot 10^{-3} h_{BS}^{(n)}) \log_{10} \left(\frac{d_n}{10^3} \right) - 18 \log_{10} h_{BS}^{(n)} + 21 \log_{10} \left(\frac{f}{10^6} \right) + 80, \tag{2.6}$$

where f is the carrier frequency, d_n is the distance between the UD and the n-th BS, and c_{light} is the light speed. This effect, on the implemented model, is deterministic and time invariant.

2.2 Angle Channel Model

The model for the calculation of the angles is based on [17]. Each UD transmits to all the BSs, sending a pilot signal. Each BS receives one or more signals for a single pilot signal due to partially reflections of this on obstacles. For every received reflected signal, we extract the first angle of reflection respect the signal sent by the UD (departure angle) and the last angle of the signal received by the BS (arrival angle). The propagation between all the other possible reflections of the signal can not be estimated.

We suppose to observe N_{ref} reflections for each of the signals received. The total number of observable angles, for a single signal, is $2 \cdot N_{ref}$. The formula for angles from UD in LoS with a BS is

$$\Phi_{LoS} = (\text{atan2d}(d)\text{mod}360) + c_{Los}\alpha_m, \tag{2.7}$$

where atan2d is four-quadrant inverse tangent in degrees, c_{LoS} is spread of angles for the LoS case and α_m is the fixed offset angle.

The values in NLoS, instead, are calculated as:

$$\Phi_{NLoS} = \Phi' + (\text{atan2d}(d)\text{mod}360) + c_{NLoS}\alpha_m, \tag{2.8}$$

where Φ' is a uniform random variable and c_{NLoS} is the spread of arrival angles for the NLoS case. The value Φ' is obtained from

$$\Phi' \sim 2 \cdot 10^{\mu_{lg}+\sigma_{lg}} U(-1, 1), \tag{2.9}$$

where μ_{lg} is the mean value of 10-base logarithm of angle spread and σ_{lg} is the standard deviation of 10-base logarithm of angle spread. These two values, c_{LoS} and c_{NLoS} are different for angles of arrival and departure.

All the uniform U(-1,1) variables of signals direct at the same BS are spatial correlated. To create a spatial correlation between uniform variables first we generated normal Gaussian variables $N(0, 1)$ and then we apply the spatial correlation between every point of the space as in (2.4) with $\sigma_{var} = 1$. Now we transform the spatial correlated normal Gaussian variables in uniform variables using the

transformation function defined as

$$TF(x) = \frac{1}{2}\left(1 + \text{erf}\left(\frac{x}{\sqrt{2}}\right)\right), \tag{2.10}$$

where erf is the error function, i.e,

$$\text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^x \, dx. \tag{2.11}$$

To define if a position is LoS or not respect a BS, we apply a probabilistic approach. The probability that a point in the map is in LoS respect a BS is

$$P_{LoS} = \begin{cases} 1, & \text{if } d \leq 18m \\ \frac{18}{d} + \exp\left(-\frac{d}{63}\right) \cdot \left(1 - \frac{18}{d}\right), & \text{if } d \geq 18m \end{cases} \tag{2.12}$$

The remaining part of the map are considered in NLoS respect the BS.

2.3 Correlation Model

One characteristic of the proposed channel models is that the random variables are spatial correlated with the random variables of the same channel. If we want correlated random variables with respect to random variables of the same position but belonging to the channels of others BSs, we need the following procedure. In order to obtain this correlation we need to collect in a vector $\boldsymbol{g}_{grv}$ the Gaussian random variables with respect to all the BSs for the same position. This vector is multiplied by the square root of the matrix of spatial correlation Σ. This matrix is obtained form (2.4) to all the combinations of BS positions, with $\sigma_{var} = 1$ and $\boldsymbol{p}_i$ with $i = 1, \ldots, N_{BS}$ that represent the positions of the BSs. The representation of the matrix Σ is seen in Fig. 2.1.

For simplicity if this effect of correlation is applied on the channel models the generated values are labeled as *correlated*. Instead, if this effect is omitted, the values are labeled as *uncorrelated*.

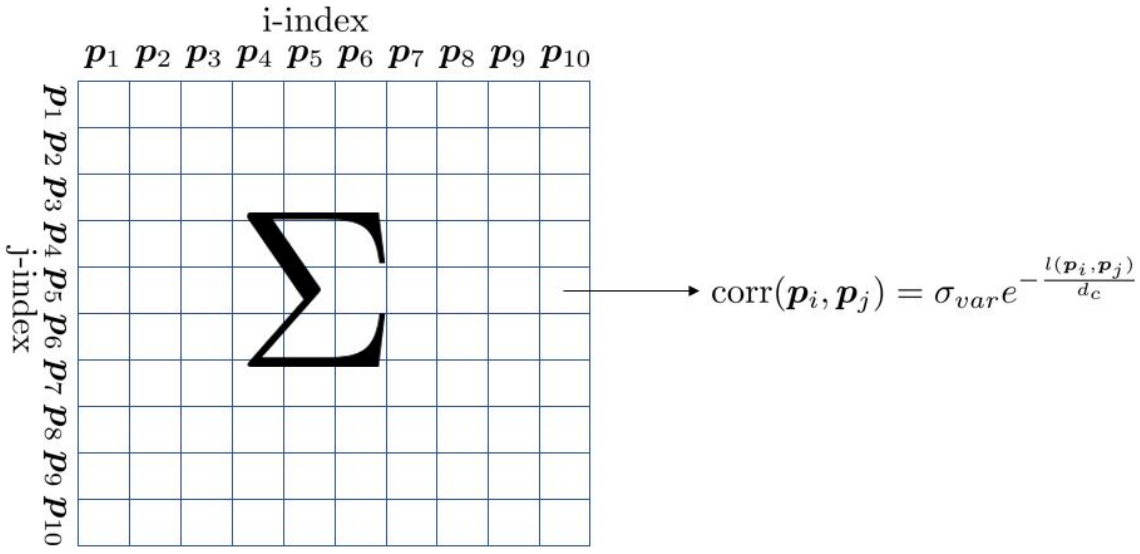


Figure 2.1: Representation of the matrix Σ.

Chapter 3

IRLV And Attacks By ML

The tasks that we want to implement are two, the binary classification and the ranking.

Binary Classification

The binary classification is a task that given a vector of feature in input returns a label that can be "1" or "0". In our case the two labels are associated to the two states as follows: 1 refers to the case wherein user is inside A_0, and 0 refers to the case wherein the user is outside of this zone. An error occurs when a signal from a UD inside A_0 is tagged with value 0 or vice versa.
A binary classifier $\mathcal{C}$ is an algorithm that starting from a feature vector returns a predicted label, defined as $\mathcal{C} : \mathbb{R}^m \to \{0, 1\}$, where m is the number of features in input, and represented in Fig. 3.1.
The set $\mathcal{D}$ is the set of available samples, composed by M_D sample of dimension m.
The two fundamental phases in the generation and evaluation of the binary classifier are:

1. Training phase: In this phase, the algorithm receives a set $\mathcal{S} \subset \mathcal{D}$ of $M_{\mathcal{S}}$ samples and their respective labels at the input and, by an optimization algorithm, it modifies its internal parameters. The data used are obtained from signals of specific UDs, for which the positions are known. The information in input *trains* the algorithm, because the algorithm tries to understand from these data the law of the analyzed binary classification problem. The set $\mathcal{S}$ is called training set.

2. Test phase: This is the prediction part, i.e., the labeling of $M_{\mathcal{E}}$ new samples in set $\mathcal{E} \subset \mathcal{D}$.

In this step the trained algorithm, from the previous step, receives new data from signals of UDs in different positions and assigns it one of the two possible labels. The set $\mathcal{E}$ is called test set.

The most important step is the first one, because in this step the binary classifier is optimized for the specific problem, by setting the free parameters in order to minimize the loss function, that measures the precision of the model.
Instead, to understand the performance of the ML algorithm, the Test phase it is necessary. Indeed, the labels predicted during the test phase are compared with the true labels. The prediction error can be distinguished into:

- False alarms, occurring, with probability P_{fa}, when a UD in the authorised zone is localised in the unauthorised zone, i.e., labelled "0" instead of "1".

- Misdetection, occurring, with probability P_{md}, when a UD in the unauthorised zone is localised in the authorised zone, i.e., labeled "1" instead "0".

Ranking

Ranking is a procedure to sorted the value of input respect some metrics, that works with the set $\mathcal{E}$ and $\mathcal{S}$. The ranker $\mathcal{R}$, starting from the vector of features in the set $\mathcal{E}$, returns a sorted array $\mathcal{L}$ of the same vectors from the most significant to the least one, i.e., from vectors that are give more information to the ones that give the least information those in $\mathcal{S}$. It is defined as $\mathcal{R} : \mathbb{R}^{M_{\mathcal{E}} \mathrm{x} m} \rightarrow \mathbb{R}^{M_{\mathcal{E}} \mathrm{x} m}$ and it is build by a trained ML model $\mathcal{G}$ applied to all the vectors in input of $\mathcal{R}$, that return the value of metric for sorting.
To train this internal algorithms $\mathcal{G}$ only the value in input is necessary. The definition of the model $\mathcal{G}$, the training phase and the test phase will be presented later, because they are different for every two implemented models. For simplicity we say that a data set trains $\mathcal{R}$ when it trains internally $\mathcal{G}$.
The general model $\mathcal{R}$ is showed in the Fig. 3.2, where the vectors of $\mathcal{E}$ is passed to the model $\mathcal{R}$ and it is returned the same vectors sorted by metric returned by $\mathcal{G}$. The representation of the $\mathcal{G}$ is provided in Fig. 3.3.

Now we will describe the mathematical structure of the algorithm. In the next two sections we

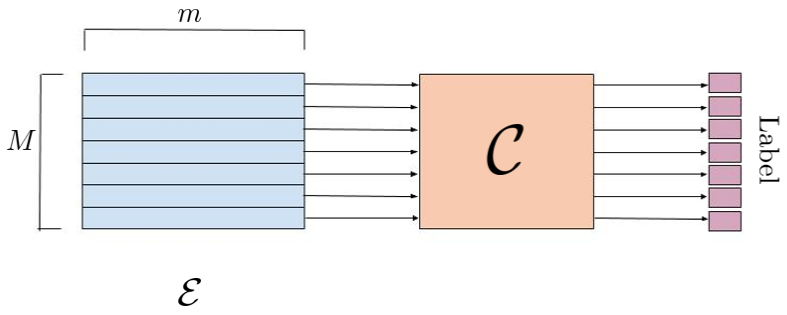


Figure 3.1: Block scheme for C.

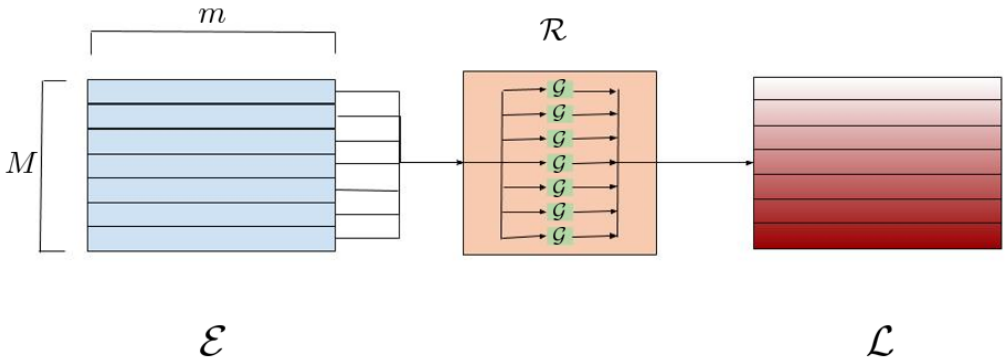


Figure 3.2: Block scheme for R.

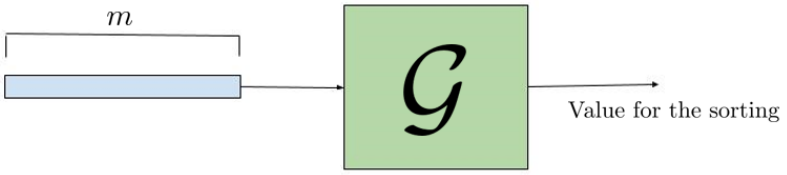


Figure 3.3: Block scheme for G.

define Artificial Neural Network(ANN) and Support Vector Machine (SVM) and their implementation to obtain good models for $\mathcal{C}$ and $\mathcal{G}$.

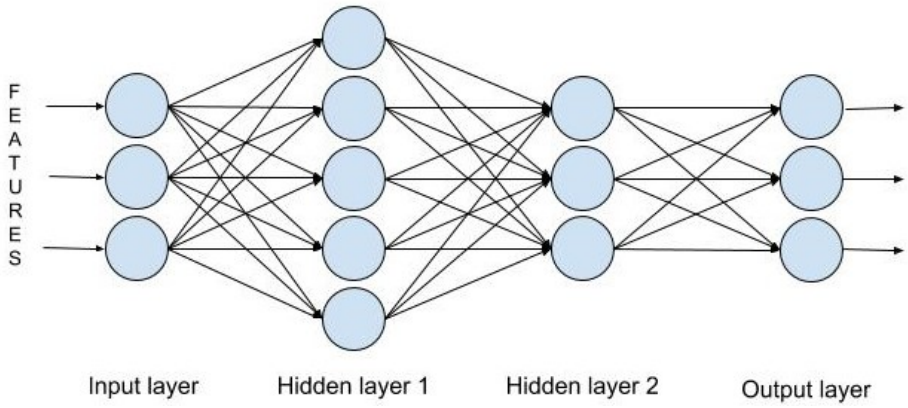


Figure 3.4: Example of ANN with two hidden layers.

3.1 Artificial Neural Network (ANN)

ANN is a system inspired by biological neural networks, composed by elements strictly connected to each others. This complex system can be descibed as a function

$$h : \mathbb{R}^m \to \mathbb{R}^o, \tag{3.1}$$

where o is the dimension of the output vector.

The ANN processes the input vector in different T stages, called layers, where in each one a sequence of calculations is performed. The layers are numbered as $t = 0, \ldots, T$, where layer 0 is the first layer, called input layer, the layer T is the last layer, called output layer, and all other $N_L = T - 1$ layers are called hidden layers.

Each layer can be seen as a union of L sub-parts called neurons. The neuron is the fundamental part of the ANN that receives the value in input for some neurons of the previews layer and sends the output value to the following layer or to the output of the ANN.

To simplify the comprehension of the cascade of functions in $h()$ we represent the ANN as a direct graph as in Fig 3.4, i.e., a graph composed by set of vertices V connected by one directional edges contained in the set E. Each node in the graph represents a neuron.

The set of nodes V can be decomposed to a union of disjoint subsets:

$$V = \bigcup_{t=0}^{T} V^{(t)}, \tag{3.2}$$

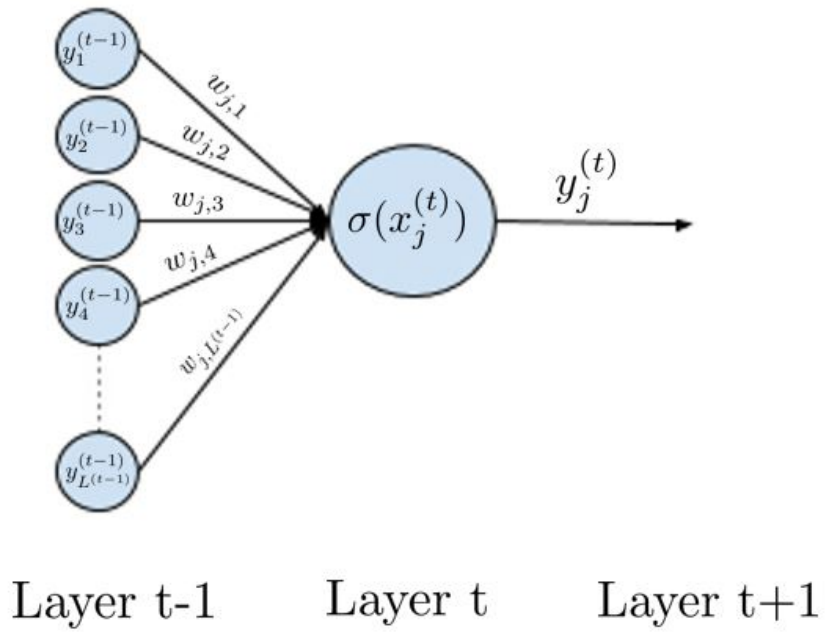


Figure 3.5: A simple representation of the neuron j of the layer t.

where $V^{(t)}$ is the set of nodes that belong to layer t.

Subset $V^{(t)}$ is defined as

$$V^{(t)} = \{v_1^{(t)}, v_2^{(t)}, \ldots, v_{L^{(t)}}^{(t)}\}, \tag{3.3}$$

where $v_i^{(t)}$ is the i-th node of the t-th layer and $L^{(t)}$ is the total number of nodes on the t-th layer. The set of edges E, instead, is composed by edges $e_{j,i}^{(t)}$ starting from the i-th node of the layer $t-1$ and reaching the j-th node of the layer t. For every element $e_{j,i}^{(t)}$ on E, there is a corresponding weight $w_{j,i}^{(t)}$.

All neurons of a layer, except for neurons in the input layer, implement the same function. In particular, the j-th neuron of the t-th layer $v_j^{(t)}$ tokes as input the weighted sum of the values from the incoming edges $e_{j,i}^{(t)}$ of the previous layer $t-1$, i.e.,

$$x_j^{(t)} = b + \sum_{i:(e_{j,i}^{(t)}) \in E} w_{j,i}^{(t)} \cdot y_i^{(t-1)}, \tag{3.4}$$

where $y_i^{(t-1)}$ is the value passed in output for the i-th node of the layer $t-1$ and b is the bias term.

Instead, for the input layer each neuron the value $x_j^{(t)}$ is the j-th feature of the vectors in input. The value $x_j^{(t)}$ is passed as input to the activation function, that can be different according to the

specific application of the ANN, called σ, defined as

$$\sigma(x_j^{(t)}) = y_j^{(t)}. \tag{3.5}$$

If the neuron is in the input layer the activation function is a linear function (LF) defined as:

$$l(x_j^{(0)}) = x_j^{(0)} = y_j^{(0)}. \tag{3.6}$$

If, instead, the neuron is in the other layers the activation function can be a linear or non linear function that will be selected for every applications. The complete operation of a standard neuron is shown in the Fig. 3.5.

During the training phase, the weights are selected in order to minimize a suitably chosen *loss function* over the training set $\mathcal{S}$.

3.1.1 Multilayer Perceptron

The Multilayer Perceptron (MLP) is a ANN for binary classification, with a static non-linear activation function and one single node at the output layer. This model can be seen as a function that receives a vector of m features and returns the probability that the vector belongs to the class "1". Then, $o = 1$ and the standard ANN function becomes $h() : R^m \rightarrow R$. A simple MLP structure is presented in Fig. 3.6.

The activation function used for neurons in both hidden an output layers is the sigmoid function defined as:

$$s(x) = \frac{1}{1 - e^{(-x)}}. \tag{3.7}$$

To know the label chosen by the MLP for a sample in input, it is necessary to take the output of the only node in the output layer and use it as input of a decision function (DF) that compares this value with a fixed threshold. The used DF is $d()$: $[0, 1] \rightarrow \{0, 1\}$ defined, as

$$d(x) = \begin{cases} 1, & \text{if } p \geq \lambda \\ 0, & \text{if } p < \lambda \end{cases} \tag{3.8}$$

where λ is the fixed threshold. This λ value can be modified to obtain desired P_{md} or P_{fa}, because setting the value λ the system is more suitable to use the label "1" or "0", increasing one error but reducing the other.

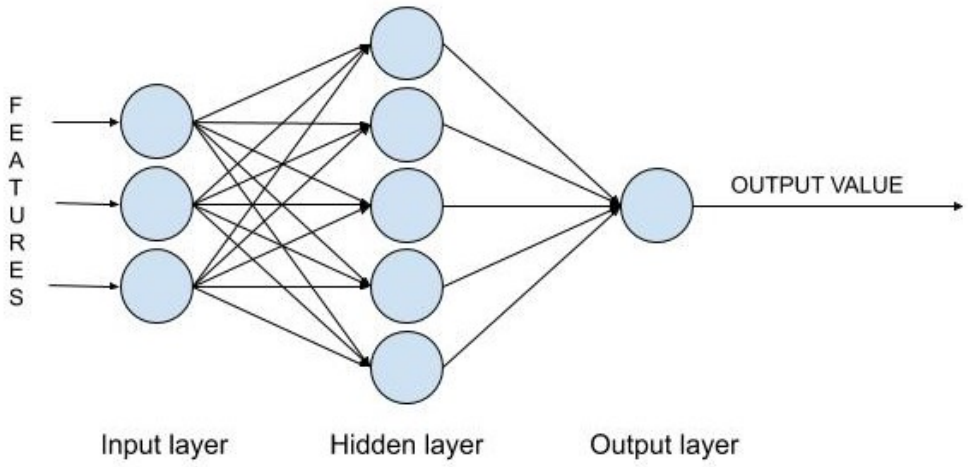


Figure 3.6: A simple MLP with 3 features in input, one value in output and only one hidden layer.

The loss function, i.e, a function capable of measuring the prediction error of the model for MLP is the Binary Cross Entropy (BCE), i.e.,

$$b(\boldsymbol{w}_{(\tilde{y},p)}) = -(\tilde{y} \cdot \log p + (1 - \tilde{y}) \cdot \log(1 - p)), \tag{3.9}$$

where $\tilde{y}$ is the true label of the samples (1 or 0), p is the output of $h()$, and $\boldsymbol{w}$ is the vector of the weights w of all the network.

The modification of function $h()$, by choice of the free parameters, the *weights*, is made by two optimization algorithms: Adam and limited-memory Broyden–Fletcher–Goldfarb–Shanno (LBFGS). These algorithms aim to reduce P_{fa} and P_{md}.

In the next sub-sections we will give the definition of the two optimization algorithms and then we will propose a theoretical study of MLP results, with respect to the training dimension $M_{\mathcal{S}}$.

Adam

The Adam algorithm derives its name from the procedure of *adaptive moment estimation* and it was presented for the first time in [18]. This iterative algorithm only requires first-order gradients of loss function but computes different learning rates, i.e., parameters that control the modifications of the internal parameters, in every iteration of the algorithm, using the first and the second moments of the gradients.

Adam is advantageous with respect to the most common learning algorithms, as the stochastic gradient descent (SGD), because it is:

- Simple to implement
- Computationally efficient
- Limited in memory requirements
- Effective on problems with a large number of training and variable parameters
- Robust in many non-convex optimisation problems

In order to implement Adam, we starts by calculating the loss function and its gradient for a sample in $\mathcal{S}$. The gradient $\boldsymbol{c}_\tau$, at the step τ of the algorithms with respect to $b(\boldsymbol{w}_{(\tau,\tilde{y},p)})$ is

$$\boldsymbol{c}_\tau = \bigtriangledown_{\boldsymbol{w}} b(\boldsymbol{w}_{(\tau-1,\tilde{y},p)}). \tag{3.10}$$

Then we calculate the first moment $\boldsymbol{\chi}_\tau$ and the raw second moment $\boldsymbol{\gamma}_\tau$, at the step τ, with the following operations:

$$\boldsymbol{\chi}_\tau = \beta_1 \cdot \boldsymbol{\chi}_{\tau-1} + (1-\beta_1) \cdot \boldsymbol{c}_\tau, \tag{3.11}$$

$$\boldsymbol{\gamma}_\tau = \beta_2 \cdot \boldsymbol{\gamma}_{\tau-1} + (1-\beta_2) \cdot \boldsymbol{c}_\tau^2, \tag{3.12}$$

where β_1 is the exponential decay rate for the first moment, β_2 is the exponential decay rate for the second moment and $\boldsymbol{c}_\tau^2$ is the vector having entries that are the square of the entries of $\boldsymbol{c}_\tau$.
Now we evaluate the modification of the first and second moments as

$$\hat{\boldsymbol{\chi}_\tau} = \boldsymbol{\chi}_\tau / (1-\beta_1^\tau), \tag{3.13}$$

$$\hat{\boldsymbol{\gamma}_\tau} = \boldsymbol{\gamma}_\tau / (1-\beta_2^\tau). \tag{3.14}$$

Finally the calculation of the new values of the *weights* are provides

$$\boldsymbol{w}_\tau = \boldsymbol{w}_{\tau-1} - \alpha_{L2} \cdot \hat{\boldsymbol{\chi}_\tau} / (\sqrt{\hat{\boldsymbol{\gamma}_\tau}} + \epsilon_{stability}), \tag{3.15}$$

where α_{L2} is the regularization term and $\epsilon_{stability}$ is a parameter for stability.
In order to make the modification of $\boldsymbol{w}$ more general and less tied to the single sample, we can take all the values $\hat{\boldsymbol{\chi}_\tau}$ and $\hat{\boldsymbol{\gamma}_\tau}$ for a specific subset of $\mathcal{S}$, called batch, and apply the average of these values on (3.15). The algorithms is iterated until the value of the loss function does not improve or the limit of iterations is reached.

LBFGS

The limited-memory BFGS is a modified version of BFGS, that uses a limited amount of memory. Indeed, L-BFGS stores only a small number of vectors, i.e., the most recent m gradients of the loss function. The LBFGS works better with a few number of samples with respect to the Adam and SGD optimizer, but needs more storage and time for computation.

The algorithm starts with an initial estimation of the weight vector, $\boldsymbol{w}_0$ (random values), and after some iterations, returns the the best possible vector $\boldsymbol{w}$ that minimizes the value of the loss function.

The first step, of the iterative procedures, is the calculation of the approximate Newton's chis direction, i.e., the direction to change the values of the vector $\boldsymbol{w}$ to minimize the loss function, obtained, for a generic step τ, as

$$\boldsymbol{d}_\tau = -\boldsymbol{H}_\tau \boldsymbol{c}_\tau, \tag{3.16}$$

where $\boldsymbol{H}_\tau$ is the inverse of the Hessian matrix. The method adopted to calculation of the value of $\boldsymbol{d}_\tau$ is the called two-loop recursion method [19].

For the correct functioning of the next step, we need to save the last values of

$$\boldsymbol{s}_\tau = \boldsymbol{w}_{\tau+1} - \boldsymbol{w}_\tau, \tag{3.17}$$

$$\boldsymbol{f}_\tau = \boldsymbol{c}_{\tau+1} - \boldsymbol{c}_\tau, \tag{3.18}$$

where $\boldsymbol{s}_\tau$ is the variation of the $\boldsymbol{w}$ on the last iteration and $\boldsymbol{f}_\tau$ is the gradient variation in the last iteration. It is also necessary to initialize the approximation of the inverse matrix as $\boldsymbol{H}_0^0 = \frac{\boldsymbol{s}_\tau^T \cdot \boldsymbol{f}_{\tau-1}}{\boldsymbol{f}_{\tau-1}^T \cdot \boldsymbol{f}_{\tau-1}}$ and define $\rho_\tau = \frac{1}{\boldsymbol{f}_\tau^T \boldsymbol{s}_\tau}$.

Now, let us define a sequence of vectors $\boldsymbol{q}_{\tau-m}, \ldots, \boldsymbol{q}_\tau$. With the rule $\boldsymbol{q}_\tau = \boldsymbol{c}_\tau$ it is possible to set the first element and from this obtain all the remaining value of $\boldsymbol{q}_i$ with $i = t-1, \ldots, \tau - m$, as

$$\boldsymbol{q}_i = \boldsymbol{q}_{i+1} - \beta_i' \boldsymbol{f}_i, \tag{3.19}$$

where $\beta_i' = \rho_i \boldsymbol{s}_i^T \boldsymbol{q}_{i+1}$.

At this point we compute another sequence of vectors $\boldsymbol{z}_{\tau-m}, \ldots, \boldsymbol{z}_\tau$ that can be found with another recursive procedure. Indeed, starting from $\boldsymbol{z}_{\tau-m} = -\boldsymbol{H}_\tau^0 \boldsymbol{q}_{\tau-m}$ it is possible to obtain all the values with $i = \tau, \ldots, \tau - m + 1$, as

$$\boldsymbol{z_{i+1}} = \boldsymbol{z}_i + (\beta_i' - \beta_i'')\boldsymbol{s}_i, \tag{3.20}$$

where $\beta_i'' = \rho_i \boldsymbol{f}_i^T \boldsymbol{z}_i$. The last computed value, $\boldsymbol{z}_\tau$, is the searched descent direction used to compute $\boldsymbol{w}_{\tau+1} = \boldsymbol{w}_\tau - \boldsymbol{z}_\tau$.

This procedure is repeated until either the value of the loss function does not improve or until the algorithm has used all samples in $\mathcal{S}$.

Fundamental Learning Algorithm

Before the implementation of IRLV by MLP, we need to understand the effects due to the ANN dimension and the training set dimension on the accuracy of the model. Indeed, if these dimensions grow the accuracy increases but also the complexity and the convergence time, i.e., the time that the optimization algorithm needs to converge, increases.

These theoretical study can be applied strictly at this particular case of ANN because it is required that the network has the task of binary classification and the sigmoid as activation function.

First of all, we recall some theoretical bounds, based on the fundamental theorem of statistic learning (FTSL) [20], for the dimension of the training set. Then, we analyze the effect on performance due to the ANN dimension.

The following theorems define all the essential concepts to understand the FTSL, starting from simple definitions:

- A hypothesis class H is a set which contains all the possible learning functions $h()$ that can be generated by a learning algorithm as MLP.

- A hypothesis class H is *agnostic PAC learnable* if there exists a function $m_H : (0,1)^2 \to \mathbb{N}$ and a dimension of train set $M_{\mathcal{S}} \geq m_H(\epsilon, \delta)$, for every ϵ, $\delta \in (0,1)$ and for every distribution D over pairs of input-output objects, that, with probability of at least 1-δ, has error probability on the test set bounded as

$$L_D(h) \leq \min_{h' \in H} L_D(h') + \epsilon, \tag{3.21}$$

  where $L_D(h)$ is the probability error on the test set.

- Let H be a hypothesis class from the set of objects X to a set of binary labels $Y = \{0,1\}$ and $C = \{c_1, \ldots, c_{\tilde{m}}\} \subset X$. The restriction H_C of H to C is

$$H_C = \{[h(c_1), \ldots, h(c_{\tilde{m}})] : h \in H\}. \tag{3.22}$$

- Let H be a hypothesis class. The *growfunction* $\tau_H : \mathbb{N} \to \mathbb{N}$ is

$$\tau_H(\tilde{m}) : \max_{X \subset X : |C| = \tilde{m}} |H_C|. \tag{3.23}$$

  This function returns a value that represents the maximum number of functions $h()$ on the restriction H_c with a fixed dimension of the set C.

- The hypothesis class H *shatters* the set $C \subset X$ if the restriction H_C contains all the $2^{|C|}$ functions from C to $\{0, 1\}$.

- The VC-dimension $VCdim(H)$ of the hypothesis class H is the largest size of a set $C \subset X$ that can be shattered by H. This value measures the learning capacity of the space of functions H.

Theorem 1 (Fundamental Theorem of Statistic Learning) *Let H be a hypothesis class of functions from a domain X to $\{0, 1\}$ (as all the possible $\mathcal{C}$ implemented by MLP) and let the loss function be the 0-1 loss (as the binary cross-entropy). Assume that $VCdim(H) = d_{vc} < \infty$.*

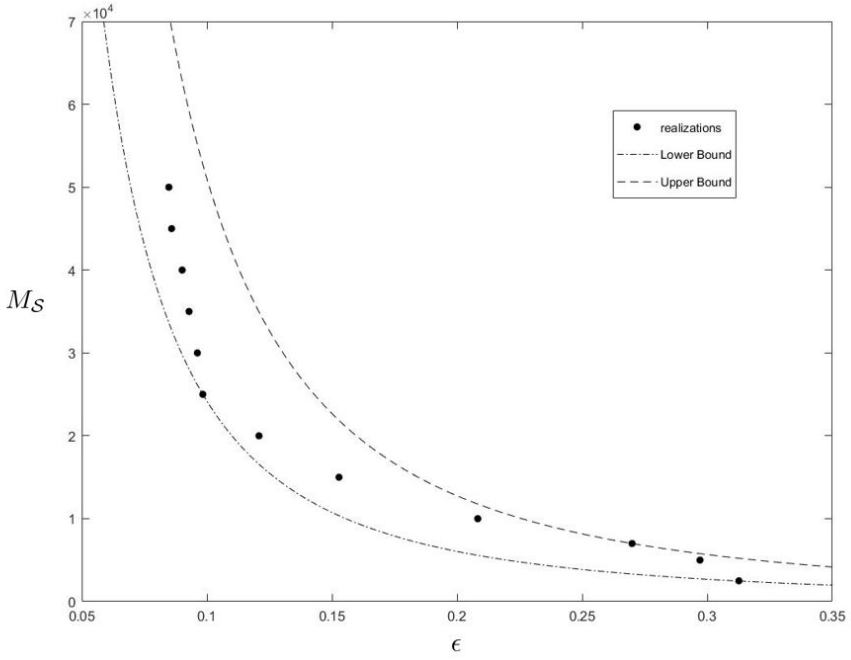


Figure 3.7: Bound for realizations with 5 BSs.

Then, there are absolute constants C_1 and C_2 such that H is agnostic PAC learnable with sample complexity:

$$C_1 \cdot \frac{d_{vc} + \log(1/\delta)}{\epsilon^2} \leq m_H(\epsilon, \delta) \leq C_2 \cdot \frac{d_{vc} + \log(1/\delta)}{\epsilon^2}. \tag{3.24}$$

This equation provides a bound on the training set size, according to the intrinsic characteristics of the considered learning method (through d_{vc}) and the desired accuracy of the test (through δ and ϵ). In particular, considering the class of functions identified by a MLP with E connections and a single output neuron, its VC-dimension grows as $\mathcal{O}(E \log E)$, with E going to infinity [21]. Fig. 3.7 shows the bounds of (3.24) for a scenario with $N_{\text{BS}} = 5$ BSs, each collecting attenuation values at a single carrier frequency $f = 1800$ MHz and for channels with shadowing and without fading. The MLP comprises $N_L = 2$ hidden layers with $L^{(i)} = 5$ for $i = 0$ and 1 . The VC-dimension is $d_{vc} = 2500$, and we set $\delta = 0.01$. The C_1 and C_2 values have been chosen such that their relative distance is the minimum guaranteeing that realization points are inside the bounds. Each dot represents the average value of ϵ obtained over 100 realizations of the attenuation maps for a fixed value of $M_{\mathcal{S}}$. We notice that the error performance saturation is well represented by the bound curves. The results give a first intuition on the possible outcome of the system performance in terms of classification error probability, comprising both P_{fa} and P_{md} errors, for a given size of the training set. We notice that, after a certain point, increasing the training set size does not improve the performance in terms of classification error. This is due to the size of the ANN (in terms of number of neurons and layers), which is not sufficient to deal with the problem complexity.

Another observation that can be done starting from (3.24) is that the dimension of the MLP can modify the accuracy of the learning algorithm and the convergence time. The two negative effects on wrong selection of the dimension of the MLP can be

- If we choose a too small MLP, after some time the learning on the training set does not improve the accuracy of the network.
- If we choose a too big MLP, the computation complexity becomes high and the algorithm can not easily select a learning function close to the excellent.

The first effect is due to the finite dimension of the set H, that contains few functions $h()$ where the closest to the optimal one $h_{optimal}$, i.e., the function that perfect represents the problem analysed decreasing the error to zero, has an error still high.

The second effect is due to the large dimension of H, that contains an enormous number of functions $h()$ difficult to be selected, and the direct proportional increase of VC-dimension and the MLP dimension.

To select the correct dimension of MLP, the only way is observing the results from practical applications. Fig. 3.8 shows the value of P_{fa} and P_{md} for different MLP dimensions in the previously described condition, with $N_{BS} = 5$ and $N_{BS} = 10$. As it seen, both in the case 5BSs, Fig. 3.8a, and 10BSs, Fig. 3.8b, the best performance is obtained with three hidden layers of one hundred nodes each.

3.1.2 ANN With Spline Activation Function

The MLP described in the previous section, is based on ANN, with a non-linear activation function for the hidden layer node and a modification of the weight parameters by one of the two algorithms. Another way to generate the learning function $h()$ is exposed below. We propose a model for $\mathcal{C}$ with the same structure of MLP and then $d()$, but with a particular activation function, that it is not fixed during the learning phase. This variable activation function is a parametric Spline function.

Spline function

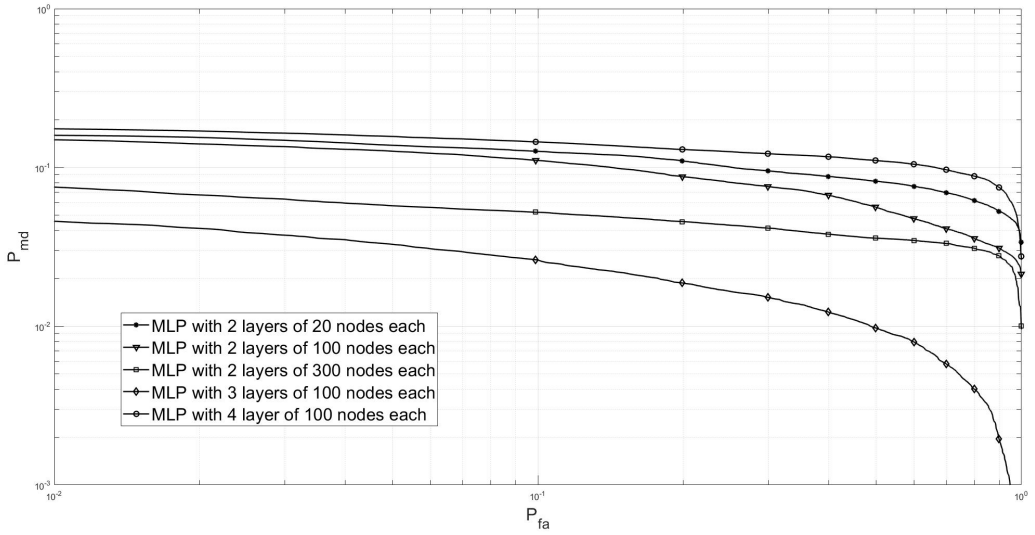
The Spline function is a parametric function with a fixed number of free parameters. These parameters are saved on a Lookup Table (LUT) with $N+1$ parameters and they are used, with a suitable interpolation scheme, to generate the function. First of all, for the correct presentation of the concepts, we will define the LUT, initialized during the Preprocessing phase, and the interpolation scheme.

After this, we will explain the procedure for the Forward Propagation and the Backward Propagation, i.e, two procedure for finding the output value of the algorithm and for the optimization of the free parameters.

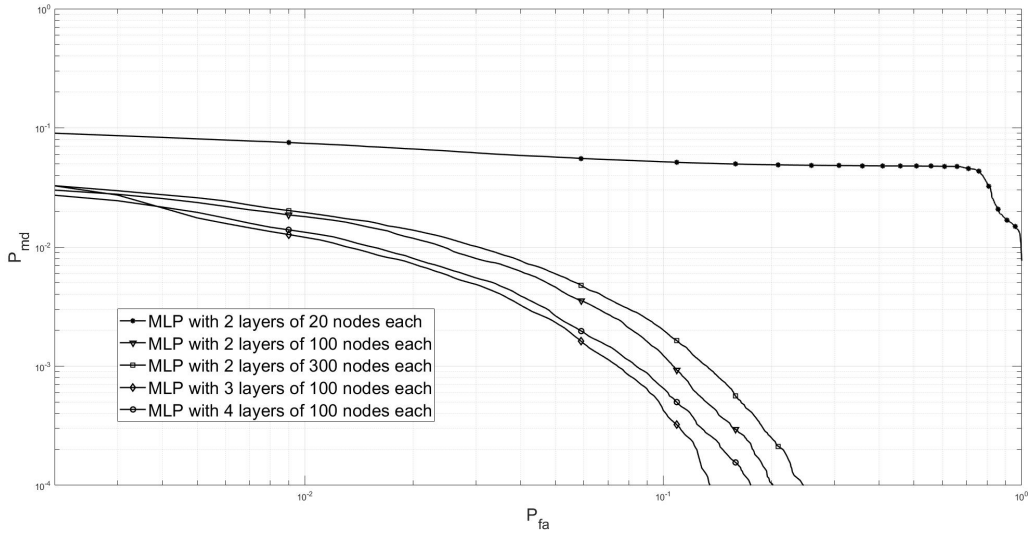
LUT definitions

The LUT is composed, for the k-th node in the t-layer, by an array of points $\{\boldsymbol{Q}_{k,0}^{(t)}, \ldots, \boldsymbol{Q}_{k,N}^{(t)}\}$. Every point is defined as:

$$\boldsymbol{Q}_{k,h} = [q_{k,x,h}^{(t)}, q_{k,y,h}^{(t)}]^T, \tag{3.25}$$



(a) $N_{BS} = 5$



(b) $N_{BS} = 10$

Figure 3.8: Results of different dimension networks with train set of $M_{\mathcal{S}}$=20000 samples.

where $q^{(t)}_{k,x,h}$ and $q^{(t)}_{k,y,h}$ are the positions respect the two axis. The points are saved on the LUT such that $q^{(t)}_{k,x,0} < q^{(t)}_{k,x,1} < \ldots, < q^{(t)}_{k,x,N}$.

The first two and the last two points on the LUT are fixed at the start of the algorithm and they are never be modified. The other elements are modified during the learning phase.

Usage of the Spline function

To understand the solid mathematics behind the spline functions [22] and the use of these as activation functions of the neurons, we need to define the following variables:

- Δx: Distance from two consecutive $q_{k,x,h}^{(t)}$ values of the same LUT.
- $z_k^{(t)}$: Internal dummy variable obtained by the following equation:

$$z_k^{(t)} = \frac{x_k^{(t)}}{\Delta x} + \frac{N-2}{2}. \tag{3.26}$$

- $i_k^{(t)}$: Index of the interval between two consecutive $q_{k,x}^{(t)}$, i.e., $[q_{k,x,i}^{(t)}, q_{k,x,i+1}^{(t)}[$, for the k-th node of the t-layer, calculated by:

$$i_k^{(t)} = \begin{cases} \lfloor z_k^{(t)} \rfloor, & \text{if } q_{k,x,0}^{(t)} < z_k^{(t)} < q_{k,x,N-3}^{(t)} \\ N-3, & \text{if } z_k^{(t)} \geq N-3 \\ 0, & \text{if } z_k^{(t)} \leq 0. \end{cases} \tag{3.27}$$

- $u_k^{(t)}$: Position of the value $z_k^{(t)}$ on the interval $[q_{k,x,i}^{(t)}, q_{k,x,i+1}^{(t)}[$ scaled between 0 and 1, for the k-th node of the t-layer, calculated by:

$$u_k^{(t)} = \begin{cases} z_k^{(t)} - i_k^{(t)}, & \text{if } q_{k,x,0}^{(t)} < z_k^{(t)} < q_{k,x,N-3}^{(t)} \\ 1, & \text{if } z_k^{(t)} \geq N-3 \\ 0, & \text{if } z_k^{(t)} \leq 0. \end{cases} \tag{3.28}$$

.

Once defined these variables, we describe the techniques for interpolation. Many choices are possible but we decide, for its low computational overhead, to use one specific cubic polynomial strategy, called Catmull Rom (CR) cubic spline basis [23]. This interpolation scheme is described,

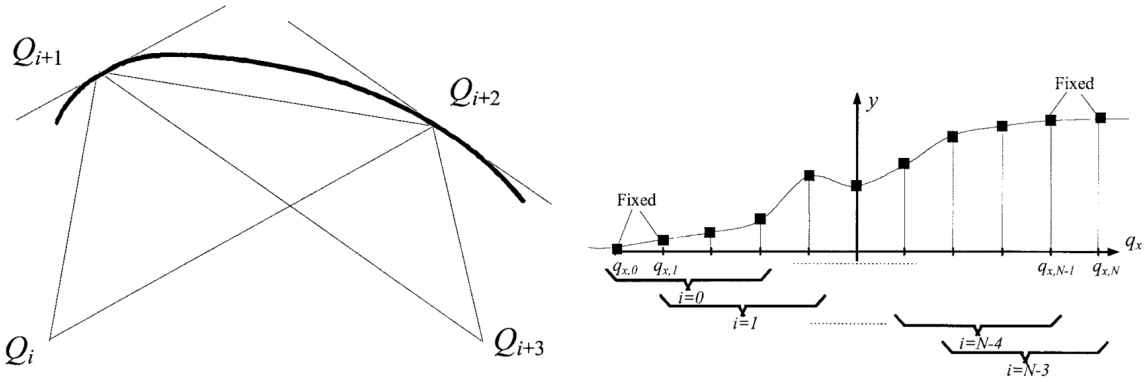
for generic node k on the layer t, by the following polynomials:

$$\begin{aligned} C_{k,0}(u) &= \frac{1}{2}(-u^3 + 2u^2 - u), \\ C_{k,1}(u) &= \frac{1}{2}(3u^3 - 5u^2 + 2), \\ C_{k,2}(u) &= \frac{1}{2}(-3u^3 + 4u^2 + u), \\ C_{k,3}(u) &= \frac{1}{2}(u^3 - u^2). \end{aligned} \tag{3.29}$$

From this, the value of Spline, for the the k-th neuron in the layer t, can be calculated as:

$$y_k^{(t)} = \sum_{m=0}^{3} q_{k,y,(i_k^{(t)}+m)}^{(t)} \cdot C_{k,m}(u_k^{(t)}). \tag{3.30}$$

This equation makes it possible to understand that starting from an index $i_k^{(t)}$, founded by (3.27), the values on the array $[Q_{k,i_k^{(t)}}^{(t)}, Q_{k,i_k^{(t)}+1}^{(t)}, Q_{k,i_k^{(t)}+2}^{(t)}, Q_{k,i_k^{(t)}+3}^{(t)}]$ are used for the generation of $y_k^{(t)}$. Fig. 3.9, from [24], shows the interpolation with the CR polynomials.



(a) The interpolation of 4 points with CR polynomials.

(b) A simple example of a spline function generated by a CR polynomials.

Figure 3.9: Two plot of CR cubic interpolation.

Preprocessing

In this first step the values of the LUT are saved for every node on hidden layers. The value for the control points $Q_{k,i}^{(t)}$ are obtained from the quantization of the sigmoid function with step equal to Δx. This quantization starts from the value $-\tilde{x}$ and arrives to $+\tilde{x}$, by passing through $N + 1 = \frac{\tilde{x}}{\Delta x} \cdot 2 + 1$ points.

Forward Propagation

By forward propagation the output value for a generic neuron is calculated from the input value. We will focus only on what happens in the k-th neuron in the t-th layer. The procedure for all the other nodes of the hidden layer is similar.

The forward propagation can be divided in three steps:

1. The value received by the neuron, $x_k^{(t)}$, is used to find $z_k^{(t)}$ by (3.26).
2. From $z_k^{(t)}$, we obtain the value of $u_k^{(t)}$ and $i_k^{(t)}$ by (3.28) and (3.27).
3. These $u_k^{(t)}$ and $i_k^{(t)}$ are used for the generation of the output $y_k^{(t)}$ by (3.30).

The values of $u_k^{(t)}$ and $i_k^{(t)}$ are saved for every node and iteration, because these values are necessary for the back propagation step.

Back Propagation

In this step, from the comparison between the output of the ANN and the desired value, the free parameters are optimized to reduce the error on the prediction.

We describe only the practical formulation of the algorithm for back propagation, very similar to SGD. First of all we define the parameters and the value used for this algorithm:

- μ_w: Learning rate for the weight.
- μ_q: Learning rate for the control points $\boldsymbol{Q}_{k,h}$.
- $e_k^{(t)}[\tau]$: Internal error variable computed as

$$e_k^{(t)}[\tau] = \begin{cases} d_k[\tau] - y_k^{(t)}[\tau], & \text{if } t = T \\ \sum_{p=1}^{L^{(t+1)}} \delta_p^{(t+1)}[\tau] \cdot w_{p,k}^{(l+1)}[\tau], & \text{otherwise ,} \end{cases} \tag{3.31}$$

  where $d_k[\tau]$ is the desired value in output of the ANN.

- $\delta_k^{(t)}$: Weighted gradient used for steepest descent:

$$\delta_k^{(t)}[\tau] = e_k^{(t)} \cdot \left(\frac{\partial y_k^{(t)}[\tau]}{\partial s_k^{(t)}[\tau]} \right) = \begin{cases} \frac{q_{k,y,1}^{(t)} - q_{k,y,0}^{(t)}}{\Delta x}, & \text{if } i_k^{(y)} = 0 \\ \frac{q_{k,y,N}^{(t)} - q_{k,y,N-1}^{(t)}}{\Delta x}, & \text{if } i_k^{(y)} = N - 3 \\ \frac{\partial y_k^{(t)}[\tau]}{\partial u_k^{(t)}[\tau]} \cdot \frac{1}{\Delta x}, & \text{otherwise,} \end{cases} \tag{3.32}$$

where can be calculated with the formula:

$$\frac{\partial y_k^{(t)}[\tau]}{\partial u_k^{(t)}[\tau]} = \sum_{m=0}^{3} q^{(t)}_{k,y,(i_k^{(t)}+m)} \cdot C'_{k,m}(u_k^{(t)}), \tag{3.33}$$

and $C'_{k,m}(u)$ for $m = 0, \ldots, 4$, are the derivative of CR polynomials, i.e.,

$$\begin{aligned} C'_{k,0}(u) &= \frac{1}{2} \cdot (-3u^2 + 4u - 1) \\ C'_{k,1}(u) &= \frac{1}{2} \cdot (9u^2 - 10u) \\ C'_{k,2}(u) &= \frac{1}{2} \cdot (-9u^2 + 8u + 1) \\ C'_{k,3}(u) &= \frac{1}{2} \cdot (3u^2 - 2u) \end{aligned} \tag{3.34}$$

.

Now it's possible to update the weights and the four control points, for the step $\tau + 1$,as:

$$w_{k,j}^{(t)}[\tau+1] = \begin{cases} w_k^{(t)}[\tau] + \mu_w \cdot \delta_k^{(t)}[\tau], & \text{if } j = 0 \\ w_k^{(t)}[\tau] + \mu_w \cdot \delta_k^{(t)}[\tau] \cdot x_j^{(t)}, & \text{otherwise,} \end{cases} \tag{3.35}$$

$$q_{k,y,(i_k^{(t)}+m)}[\tau+1] = q_{k,y,(i_k^{(t)}+m)}[\tau] + \mu_q \cdot e_k^{(t)} \cdot C_{k,m}(u_k^{(t)}) \text{ for } m = 0, \ldots, 3. \tag{3.36}$$

The algorithm is iterated until all training samples have been used.

3.1.3 Autoencoder

The AE is a non-recurrent ANN used for the implementation of the model $\mathcal{G}$. It has input and output layers of the same size, trained to reconstruct the input vector at the output. AE can be seen as function $h() : \mathbb{R}^m \to \mathbb{R}^m$.

The AE is used in image computation and denoising [25], where it is are composed by two mirrored parts (encoding part and decoding part) with a small layer in the axis of symmetry. Fig. 3.10a shows a possible of structure of AE for denoising. This structure provides fewer but richer features going to the center (reducing the number of neurons), and after to reconstruct the vector of features in input from this small set.

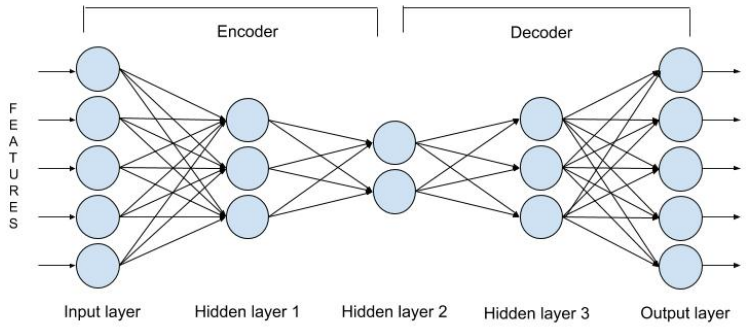
In our implemented cases, the dimension of the feature vector in $\mathcal{D}$ is too small to apply a filter of this type. The best solution is obtained by using a particular AE called Vanilla [26]. This ANN is composed by three or more layers. The input and output layers are of equal size, instead the

central layers have a larger size, equal for every hidden layers. The structure of a generic Vanilla AE can be seen in Fig. 3.10b.

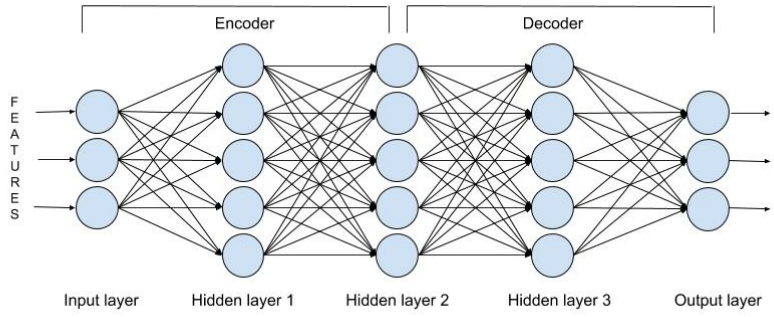
The loss function used for the AE is the mean square error (MSE),

$$MSE(\boldsymbol{x}^{(0)}, \boldsymbol{y}^{(T)}) = \frac{\sum_{i=1}^{L^{(T)}} (x_i^{(0)} - y_i^{(T)})^2}{L^{(T)}}. \tag{3.37}$$

The metric for ranking $\mathcal{R}$ is the value of the MSE between the input and the output. The optimization algorithm used for reduce the value of the loss function during the training is Adam.



(a) AE denoise model with 3 hidden layer and five input features



(b) Vanilla AE with 3 hidden layer and three feature in input and output

3.2 Support Vector Machine (SVM)

SVM [27] is on ML algorithm, widely adopted by the scientific community, that can be used for both classification and ranking. This algorithm constructs as hyperplane or set of hyperplanes in

multi-dimensional space that divide the samples in two or more classes. This is possible because each input vector of m features can be seen as a Cartesian point in an m-dimensional space. In our implementation only one hyperplane is used and it is described by the vector $\boldsymbol{w}$. The goal of the algorithm is to obtain the best separation hyperplane that has the largest distance to the nearest point of any class. The optimization problem to find the optimal value $\boldsymbol{w}$ is different depending on the implemented task.

The operations performed by an SVM can be described as

$$\tilde{t}(\boldsymbol{x^{(0)}}) = \boldsymbol{w}^T \phi(\boldsymbol{x^{(0)}}) + b, \tag{3.38}$$

where $\phi : \mathbb{R}^m \to \mathbb{R}^m$ is a fixed *feature-space transformation function* where $\boldsymbol{x}^{(0)}$ is m-size feature vector, $\boldsymbol{w} \in \mathbb{R}^m$ is the weight vector and b is a bias parameter. In (3.38) function $\phi(\cdot)$ is fixed, while vector $\boldsymbol{w}$ is properly chosen according to the specific modeled problem. This equation returns the soft output, i.e., the distance between the points and the hyper plane need to be applied to a DF, depending on the specific task performed.

$\phi(\boldsymbol{x}^{(0)})$ is typically implicit defined by the kernel Υ. The kernel can be expressed as $\Upsilon(\boldsymbol{x}_i^{(0)}, \boldsymbol{x}_j^{(0)}) = \phi(\boldsymbol{x}_i^{(0)})^T \phi(\boldsymbol{x}_j^{(0)})$, where $\boldsymbol{x}_i^{(0)}, \boldsymbol{x}_j^{(0)}$ are two input vectors, amd it can be various but we consider three options:

- Linear: $\Upsilon(\boldsymbol{x}_i^{(0)}, \boldsymbol{x}_j^{(0)}) = (x_i^{(0)})^T x_j^{(0)}$.
- 3rd grade polynomial: $\Upsilon(\boldsymbol{x}_i^{(0)}, \boldsymbol{x}_j^{(0)}) = (\boldsymbol{x}_i^{(0)} \boldsymbol{x}_j^{(0)} + 1)^3$
- Radial basis function (RBF) kernel: $\Upsilon^{(0)}(\boldsymbol{x}_i^{(0)}, \boldsymbol{x}_j^{(0)}) = \exp\left(-\frac{||\boldsymbol{x}_i^{(0)} - \boldsymbol{x}_j^{(0)}||^2}{2\sigma_{RBF}^2}\right)$ where σ_{RBF} is the standard deviation.

3.2.1 Support Vector Classification (SVC)

SVC is a procedure that uses SVM for the implementation of $\mathcal{C}$. This algorithms use the function $\tilde{t}()$ to obtain the value of the class for the feature vector in input. To do so it is necessary to apply at the value returned by $\tilde{t}()$ a DF. The optimization problem for the binary classification, on the set $\mathcal{S}$, can be written as:

$$\min_{\boldsymbol{w},b} \quad \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C_{hp}\frac{1}{2}\sum_{i=1}^{k_t}\xi_i \tag{3.39a}$$

$$\text{subject to} \quad \xi_i \geq 1 - \tilde{y}_i[\tilde{t}(\boldsymbol{x}_i^{(0)})], \quad i = 1, \ldots, M_{\mathcal{S}}, \tag{3.39b}$$

where C_{hp} is a hyper-parameter, $\tilde{y}_i$ is the for modified label for the element i on the set of vector in input, that is equal to 1 if the label is "1" and it is equal to -1 if the label is "0". $\tilde{y}_i$ for all the element of $\mathcal{S}$ are collected in $\tilde{\boldsymbol{y}}$. Inequalities in the constraints lead to a quadratic programming problem, typically solved in its dual version. Therefore it is possible to write the problem as:

$$\min_{\kappa} \quad \frac{1}{2}\boldsymbol{\kappa}^{\boldsymbol{T}}\tilde{K}\boldsymbol{\kappa} + \mathbf{1}^T\boldsymbol{\kappa} \tag{3.40a}$$

$$\text{subject to} \quad \tilde{\boldsymbol{y}}^T\boldsymbol{\kappa} = 0 \quad 0 \leq \kappa_i \leq C_{hp}, \quad i = 1, \cdots, M_{\mathcal{S}}, \tag{3.40b}$$

where $\mathbf{1}$ is a vector of $M_{\mathcal{S}}$ ones, $\boldsymbol{\kappa}$ is a vector of $M_{\mathcal{S}}$ elements κ_1, and $\tilde{K}$ is a $M_{\mathcal{S}}x\, M_{\mathcal{S}}$ matrix where every element

$$\tilde{K}_{i,j} = \tilde{y}_i\tilde{y}_j\Upsilon^{(0)}(\boldsymbol{x}_i^{(0)}, \boldsymbol{x}_j^{(0)}). \tag{3.41}$$

Solving the problem (3.40) we obtain the optimal $\boldsymbol{w}$, that satisfies:

$$\boldsymbol{w} = \sum_{i=1}^{M_{\mathcal{S}}} \tilde{y}_i\kappa_i\phi(\boldsymbol{x}_i^{(0)}). \tag{3.42}$$

For the analyzed case the DF is:

$$\text{sgn}(\tilde{t}(\boldsymbol{x}_j^{(0)})) = \text{sgn}\left(\sum_{i=1}^{M_{\mathcal{S}}} \tilde{y}_i\kappa_i\tilde{K}(\boldsymbol{x}_i, \boldsymbol{x}_j) + b\right). \tag{3.43}$$

where the sign function is defined as

$$\text{sgn(x)} = \begin{cases} 0, & \text{if } x \leq 0 \\ 1, & \text{if } x > 0. \end{cases} \tag{3.44}$$

3.2.2 One Class SVM

One-Class SVM (OSVM) is a method that can be used for $\mathcal{G}$ based on SVM, that aims at finding the optimal hyperplane that divides samples in two classes. Unlike SVC, where the values of sample with their respective labels are known, in OSVM the two classes are created to find the best hyperplanes that better divide the vector of features on the space in two class without knowing correct pair vector-label. After the hyperplane is found, function $\tilde{t}()$ returns the value of the distance for a sample respect the hyperplane.

The objective function for OSVM, given a vector of training $\boldsymbol{x}^{(0)}$, is

$$\min_{\boldsymbol{w},\xi,\tilde{\xi}} \quad \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} - \tilde{\xi} + \frac{1}{vM_{\mathcal{S}}}\sum_{i=1}^{k_t}\xi_i \tag{3.45a}$$

$$\text{subject to} \quad \boldsymbol{w}^T \phi(\boldsymbol{x}_i^{(0)}) \geq \tilde{\xi} - \xi_i, \text{with } \xi_i \geq 0, i = 1, \ldots, M_{\mathcal{S}}, \tag{3.45b}$$

where $\tilde{\xi}$ is an upper bound on the fraction of training errors and v is regularization parameters. The dual problem can be expressed also as

$$\min_{\kappa} \quad \frac{1}{2} \boldsymbol{\kappa}^{\boldsymbol{T}} \tilde{K} \boldsymbol{\kappa} + \mathbf{1}^T \boldsymbol{\kappa} \tag{3.46a}$$

$$\text{subject to} \quad 0 \leq \kappa_i \leq 1, \quad i = 1, \cdots, M_{\mathcal{S}}, \quad \mathbf{1}^T \boldsymbol{\kappa} = v M_{\mathcal{S}}. \tag{3.46b}$$

The solution of problem (3.46) is the optimal value $\boldsymbol{w}$. The DF for OSVM is

$$\text{sgn} \left(\sum_{i=1}^{M_{\mathcal{S}}} \kappa_i \tilde{K}(\boldsymbol{x}_i^{(0)}, \boldsymbol{x}_j^{(0)}) - \tilde{\xi} \right) \tag{3.47}$$

The metric used for the ranking $\mathcal{R}$ is the distance sample-hyperplanes, i.e., the soft output.

Chapter 4

Defense And Attack Methods

In the last chapters we exposed the ML techniques and the channel models that will be used for tests. Now we will describe the strategy and the combination of algorithms for both IRLV and attacks.

In the assumed situation, UDs try to authenticate themselves in the system by sending a pilot signal to the BSs. Each BS, from signal received by its antennas, is able to obtain the values of attenuation and the angles of departure and arrival.

We remember that the UD is accepted if the pilot signal are recognized as coming from the region A_0, otherwise the authentication request of the UD is rejected. The attacker (AT) wants, by sending a pilot signal from the unauthorized zone A_1, to be recognized as an UD in the authorized zone. The defender has the possibility to move around in space and send various signals to BSs from different locations, from both in A_0 and A_1, to estimate the value of the features for these spaces. The attacker, instead, does not have access to zone A_0 but he can know the value of features with respect to the signal from the zone A_1.

4.1 Defence Strategies

The defender moves in the space, sending pilot signals to every BSs and collecting per each time the value of features. These vectors of information are labeled with the correct label, because the defender knows from which of the two zones the pilot signal comes.

The two strategies for the implementation of defense method are:

- Direct method: In this method the training set is directly applied as input to ML algorithms for binary classification.

- Ranking method: The training set is first applied as input to a ranking algorithms, and after the skimming of the repetitive and similar sample, the remaining set is passed to a binary classification algorithm.

4.1.1 Direct Method

In this type of defense we apply directly the collected data set, without pre-processing, to $\mathcal{C}$. During this learning, we observe that many data in the training are useless and repeated several times. For this reason we pre-process them before application of the binary classification algorithm.

4.1.2 Ranking Method

The data pre-processing is a procedure of data analysis, having the purpose of modifying or reduce the dimension of the input data set. We propose to apply a strong dimension reduction with the use of ranking methods by an iterative procedure called Sample Important Extractor (SIE). This algorithm is based on a Ranker $\mathcal{R}$ that it is applied at different parts of the data set.
The SIE starts dividing the data set $\mathcal{E}$ in B parts, every part containing the same number of samples. The first sub-set, obtained by separation, is used for training $\mathcal{R}$. The same sub-set is also passed to input at $\mathcal{R}$ that returns the same samples sorted by $\mathcal{G}$ metric.
This ordered set is divided into 4 parts. Only the first and the third are taken, instead, the two remaining parts are dropped as in Fig. 4.1. This is done, because from a data set of random values, to obtain a better set for the training of $\mathcal{C}$, we need to extract the most significant values (first part) and similar but not to much (third part), with respect to the previous vectors of features used for training $\mathcal{R}$.

The remaining ordered sub-set is added to a new set $\mathcal{E}'$, that will be the training set of $\mathcal{C}$, and it is used for a second training of $\mathcal{R}$. Now, new part of $\mathcal{E}$ is used as input of $\mathcal{R}$. The resulting set is halved as shown in Fig. 4.1, passed as training for $\mathcal{R}$, and added at the set $\mathcal{E}'$. This procedure is iterated until all the sub-sets in $\mathcal{E}$ are used. The iterative procedure is represented in the Fig 4.2.
All the values used for training $\mathcal{R}$ are collected and used for training $\mathcal{C}$. Model $\mathcal{C}$, trained with the

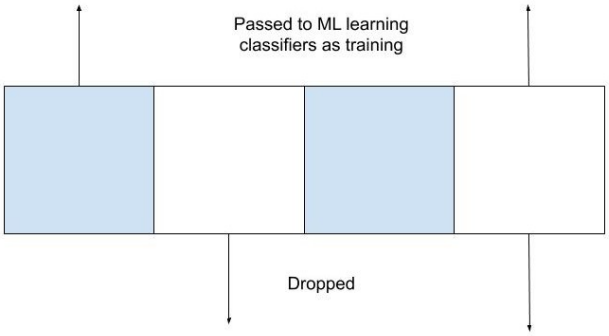


Figure 4.1: Schematic division of the ordered data set returned by $\mathcal{R}$.

set $\mathcal{E}'$, has better results respect the model trained with $\mathcal{E}$, because we remove many values that are too similar and do not give much information.

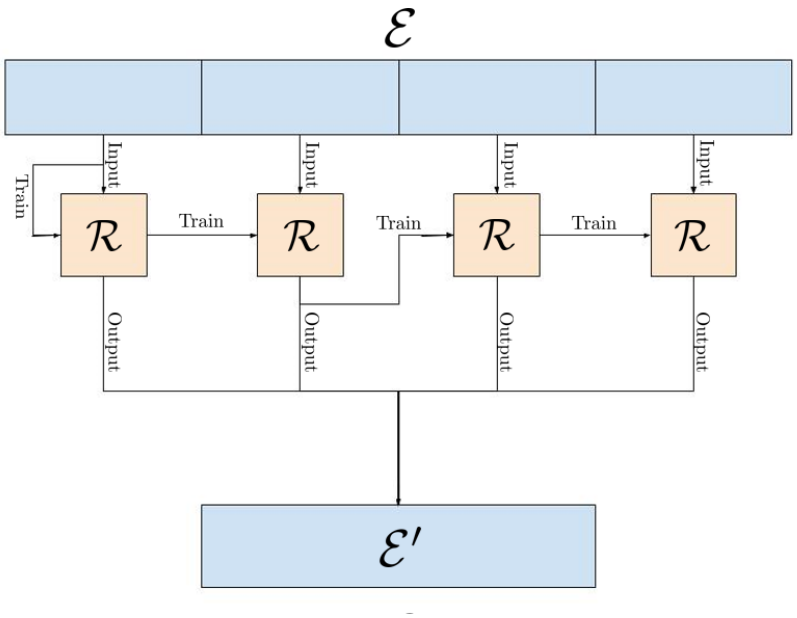


Figure 4.2: SIE model with $B = 4$. Each $\mathcal{R}$ is the same in all the graph.

4.2 Attack Strategies

The AT sends pilot signals and wants to be authenticated by $\mathcal{C}$ as an user in region A_0.

For the analysis of the attacks we consider systems where the only attenuation is used as feature.

The attacker have two main strategies:

- Passive attack: With the passive attack strategy each position of the attacker inside A_1 defines a different attack, i.e., the AT selects a physical location where he performs the attack. This attack strategy does not require special hardware with respect to the standard UD.
- Active attack: This type of attack strategy provides that the AT directly selects the attenuation with respect to each BS, irrespective of his physical position. This strategy requires that the AT is equipped with directional antennas, so that he can induce different estimated attenuation values to each BS.

The AT is free to attack with any number of attenuation vectors. However, note that a sequence of wrong attacks could be detected by the BSs that can adopt suitable countermeasures or raise warnings. Moreover, active attacks are more power consuming, representing a cost for the AT. Therefore, it is in the best interest of the AT to find effective attenuation values. This holds for both passive and active strategies, therefore we assume the attacker can employ an $\mathcal{R}$ model, to better select attenuation vectors suitable for attacks.

4.2.1 Passive Attacks

With passive attacks the AT uses attenuation vectors only from positions in A_1, without forging new attenuation vectors. We introduce two passive attacks, namely the random attack (RA), and the passive ranking attack (PRA).

- Random attack (RA): The first and most simple attack provides that the AT attacks the IRLV system from positions chosen uniformly at random inside A_1. The block scheme is shown in Fig. 4.4, where $\mathcal{E}$ is a set of features read at the BSs corresponding to uniformly randomly selected positions in A_1.
- Passive ranking attack (PRA): In this attack type, instead, the AT gives more priority to attenuation in $\mathcal{E}$ with better chances to be accepted by the IRLV system, therefore he selects the positions from which the attacks are performed. To this end, he selects the next point for the attack, using an ML ranking approach. In particular, he trains the model $\mathcal{R}$ with training set $\mathcal{S}$, another set of features collected from pilot signal sent from A_1. Then, he applies, at the ranker $\mathcal{R}$, all elements in $\mathcal{E}$. The output of the ranking procedure is stored in a list $\mathcal{L}$

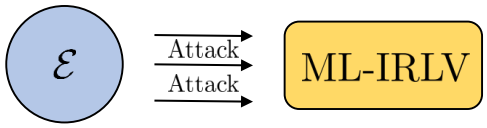


Figure 4.3: Block scheme of RA.

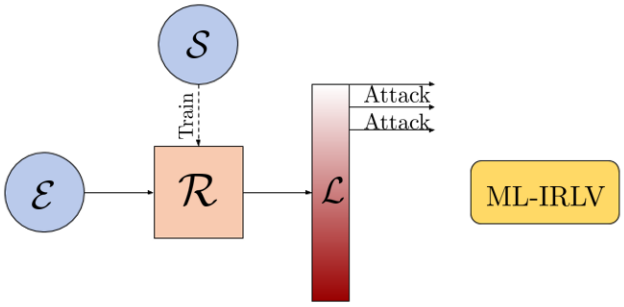


Figure 4.4: Block scheme of PRA.

ordered from the best to the worst ranking score. Attacks are then performed following the order of $\mathcal{L}$. The block scheme of this type of attack is shown in Fig. 4.4.

4.2.2 Active Attacks

With active attacks, the AT generates new attenuation vectors that are not necessarily already available inside A_1.

First, we define the perturbation function $\mathcal{P}(\boldsymbol{g})$ that generates a new features values, not necessarily corresponding to a physical position, around a given attenuation vector $(\boldsymbol{g})_{\text{dB}}$ as

$$\mathcal{P}(\boldsymbol{g}) = (\boldsymbol{g})_{\text{dB}} + \tilde{s}, \quad \tilde{s} \sim \mathcal{N}(\mathbf{0}_{N_{BS}}, \sigma_{\tilde{s}}^2 \boldsymbol{I}_{\boldsymbol{N_{BS}}}), \tag{4.1}$$

where $\boldsymbol{I}_{N_{BS}}$ and $\mathbf{0}_{N_{BS}}$ denote the identity matrix of size N_{BS} and the all-zero vector of size N_{BS}, respectively. The motivation behind $\mathcal{P}$ comes from the channel model by (2.3). Given a vector $\boldsymbol{g}$, with $\mathcal{P}(\boldsymbol{g})$ we are *perturbing* the shadowing component in the attempt to find an attenuation vector closer to the values obtained from pilot signal sent by an UD in the region A_0. Note that adding the random component $\tilde{s}$ to an already existing attenuation vector $\boldsymbol{g}$, is better than just generating a completely random point, because we are creating a new object which has a true path loss value.

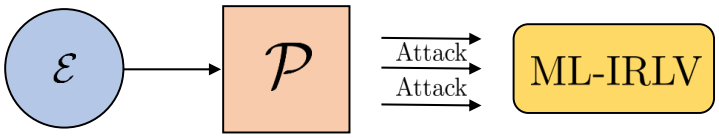


Figure 4.5: Block scheme of RandPA.

Another function that we need to introduce is the k-means block $\mathcal{K}$. $\mathcal{K}$ receives the set $\mathcal{E}$ and returns the value of the best $N_{centroid}$ centroids that minimize the intra-class variance, i.e., the sum of squared distances from each data point being clustered to its cluster center. Clustering is the task of grouping a set of objects so that objects in the same group are more similar to each other, and more different with respect to those in other groups.

This function is applied on our data set $\mathcal{E}$ to obtain representing points of a large space of values, reducing the size of the data set without lose too much information. We now introduce five active attack strategies, namely the Random perturbed attack (RandPA), the Perturb-then-rank attack (PTRA), the Rank-then-perturbed attack (RTPA), Rank-perturb-rank attack (RPRA), Kmeans-then-perturb attack (KTPA) and Kmeans-perturb-rank attack (KPRA).

RandPA

In RandPA, the AT selects $\mathcal{E}$ as in RA, and then applies $\mathcal{P}$ to generate new points. In particular, we apply $\mathcal{P}$ on each $\boldsymbol{g} \in \mathcal{E}$, and we denote this operation as $\mathcal{P}(\mathcal{E})$.

The AT then picks random points from $\mathcal{P}(\mathcal{E})$ and performs the attack, as shown in Fig. 4.5. Note that in this case the AT needs to explore $\mathcal{A}_1$ only to collect attenuation vectors $\mathcal{E}$, while he can performs the attacks anywhere (by properly using multiple transmit antennas) since the attack attenuation vectors are not associated to specific positions.

PTRA

The second active attack is similar to PRA, but the generation function $\mathcal{P}$ with ranking machine $\mathcal{R}$ selects good attack points from the perturbed attenuation vectors. The block diagram of this attack is shown in Fig. 4.6.

In the first step $\mathcal{R}$ is trained with $\mathcal{S}$, as in the RPA. Now the value in $\mathcal{E}$ are perturbed with the function $\mathcal{P}$, and the resulting values are ranked with the trained $\mathcal{R}$. The result of the ranking is

stored in a set $\mathcal{L}$.

However, this time the AT attacks starting from points with worst ranking, indicated in Fig. 4.6 by attack arrows located at the bottom of $\mathcal{L}$, contrary to Fig. 4.4 in which attack arrows are located at the top of $\mathcal{L}$. This is done because the perturbation generates new values that may be too far even from the set of possible sample in $\mathcal{E}$ and with this selection we return near at the possible values. Values too different from those in $\mathcal{E}$ are easily recognized as external from A_0 and labeled by C with the value "0".

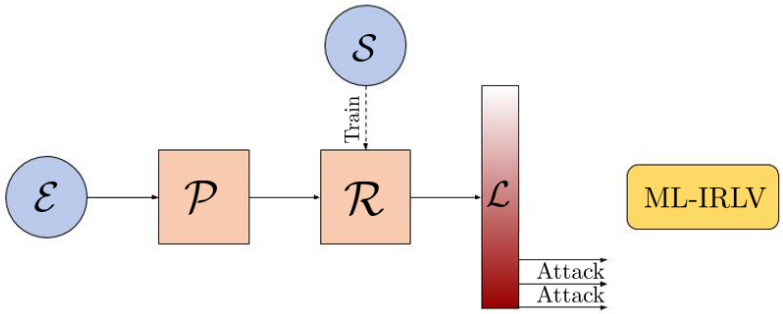


Figure 4.6: Block scheme of PTRA.

RTPA

In RTPA, the AT applied the same algorithm of the PTRA on $\mathcal{E}$, but in inverse order. In this attacks, rather than perturbing (by $\mathcal{P}$), directly $\mathcal{E}$ like in the RPA, the AT applies $\mathcal{P}$ only on the points of $\mathcal{E}$ with best ranking. Therefore, he first computes the ranking, and at this point, the block diagram is similar to Fig. 4.4 with the difference that elements of $\mathcal{L}$ with best score are not used for the attack, but rather as input to $\mathcal{P}$. This is done because starting from good values, a permutation P can making them even better for attacks. The complete attack model can be seen in Fig. 4.7.

RPRA

RPRA combines the ranking, used two times, and perturbation. As shown in the Fig. 4.8, first the ranking system $\mathcal{R}$ is trained with the data set $\mathcal{S}$ and applied at the data set $\mathcal{E}$. Now only

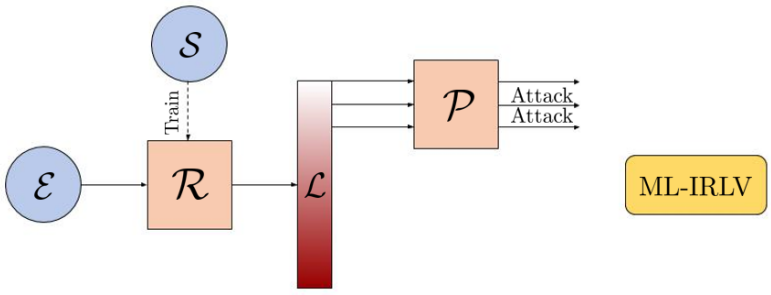


Figure 4.7: Block scheme of RTPA.

attenuation vectors of $\mathcal{E}$ with higher ranking are perturbed by $\mathcal{P}$. The result is passed another time to the ranker $\mathcal{R}$ that saves all the values in the ordered set $\mathcal{L}'$. The AT attacks with the value in the set $\mathcal{L}'$ starting from the one with the worst ranking as in PTRA. This strategy has the best results because it takes full advantage of the potential of the models $\mathcal{R}$ and $\mathcal{P}$ because the ranking values are already good for a possible attacks. By applying $\mathcal{P}$ it is generates sample that may be too far from the value of the region A_0. Reusing the trained $\mathcal{R}$ on these generated values, the AT provide values closer to those physically achievable.

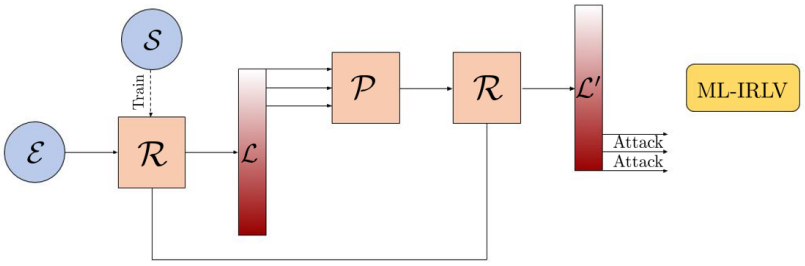


Figure 4.8: Block scheme of RPRA.

KTPA

This technique mixes clustering and perturbation procedures. First model $\mathcal{K}$, is applied on $\mathcal{E}$, to obtain the best value of cluster centers, i.e, the values that best represent the space A_1. Then, the

cluster centers values are permuted by $\mathcal{P}$ and the attacker try some of these generated values for attacks.

This procedure is done because, taking the cluster centers, AT removes the possibility that an external value, which has a low chance of being accepted by $\mathcal{C}$, is taken to be permuted and used for attack. The scheme of this attack can be see in Fig.4.9.

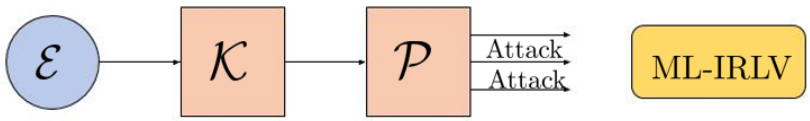


Figure 4.9: Block scheme of KTPA.

KPRA

In KPRA, the procedure applied is similar from the one used to RPRA. The only difference is that the first $\mathcal{P}$ is substituted with a $\mathcal{K}$. This is done because selecting the first part of the array $\mathcal{L}$ as in RPRA, we remove some values from section of the space, instead with $\mathcal{K}$ the value of these areas are only agglomerated into a single value and not completely rejected.

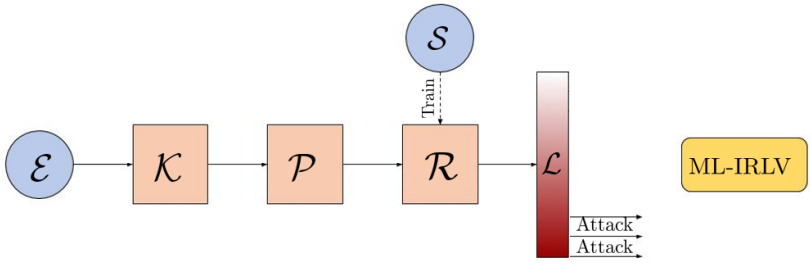


Figure 4.10: Block scheme of KPRA.

Chapter 5

Numerical Results

In this chapter, we present the performance of the proposed IRLV under the various methods. The data set $\mathcal{E}$ and $\mathcal{S}$ are created by simulation with the one or both the channel models described in Chapter 2. In all the cases, we consider a carrier frequency of $f = 2.12$ GHz, and $h_{BS}^{(n)} = 15$ m for all the $N_{BS} = 10$ BSs. We consider, where spatial correlation is assumed, a decorrelation distance $d_c = 75$ m for the RSS channel model and of $d_c = 50$ for the angle channel model.

The regular UD transmits the pilot signal with power $P_{tx} = 0$ dBm. Zone A is a square of $d_l = 525$ m on each side. The space is sampled every 0.5 meters, thus the space map is divided in a 1050x1050 matrix with 1.102.500 points.

The BSs are positioned as in Fig. 5.1. On the corners there are four buildings of square base with side $d_e = 255$ m that are considered when shadowing is added. The authentication area A_0 is positioned in one of these buildings as in Fig. 5.1. A_0 is a square of dimension $\tilde{\beta}_1 = 150$ m and $\tilde{\beta}_2 = 150$ m, at horizontal distance $d_1 = 50$ m and vertical distance $d_2 = 50$ m with respect to the left corner of A.

Around zone A_0, we create a Ring zone R, where the defender can access to estimate the features from the points in this space, while the AT has not this possibility. It is parametrized by the *normalized incremental area* $\alpha_{ring} \in \{0, 1\}$ such that its side has dimension $\beta_{ring}^2 = (1 + \alpha)\tilde{\beta}_1^2$. In other words, $\alpha = 0$ means there is no ring, while with $\alpha = 1$ the ring side is $\sqrt{2}\tilde{\beta}$.

To each point in the space, the features with respect to a pilot signal sent from that position can be assigned. To understand if a point is in LoS or NLoS we use these two approaches. For the RSS model a point is in NLoS if the signal sent by the UD to the BS crosses buildings. Instead, if the

signal travels between buildings, on a road, the point from which it stated is considered in LoS. For the Angle model the choice is different, because for every point of A the LoS condition is set according to (2.12). A realization of attenuation map (only considering the shadowing and path loss effect) with respects to a generic BS positioned in the center of A can be seen in Fig. 5.2.

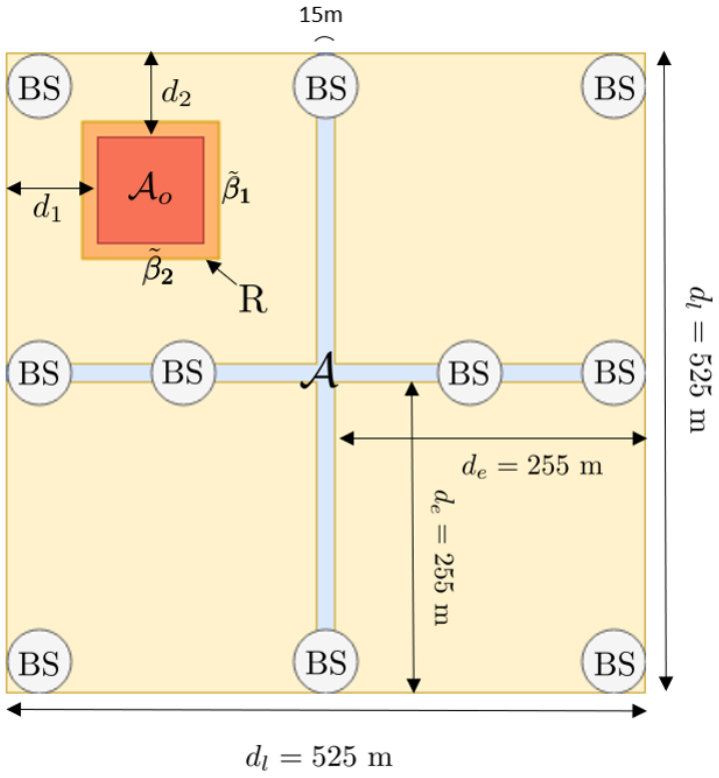


Figure 5.1: Map A with 10 BSs.

Instead, a possible map of angle of departure and arrival with respect to a generic BS positioned as the center of A can be seen in Fig. 5.3.

In the following section, first we analyze the performance of IRLV on the channel models previously presented and on the real attenuation features obtained from [1]. Then we analyze, only using the RSS simulation channel, the attack models on two defenses: IRLV implemented with MLP (MLP-IRLV) and IRLV implemented with SVC (SVC-IRLV). The ranking for the defense is implemented with the AE instead the AT use, for ranking, also OSVM.
The parameter values for the model implemented are tabulated in Appendix A.

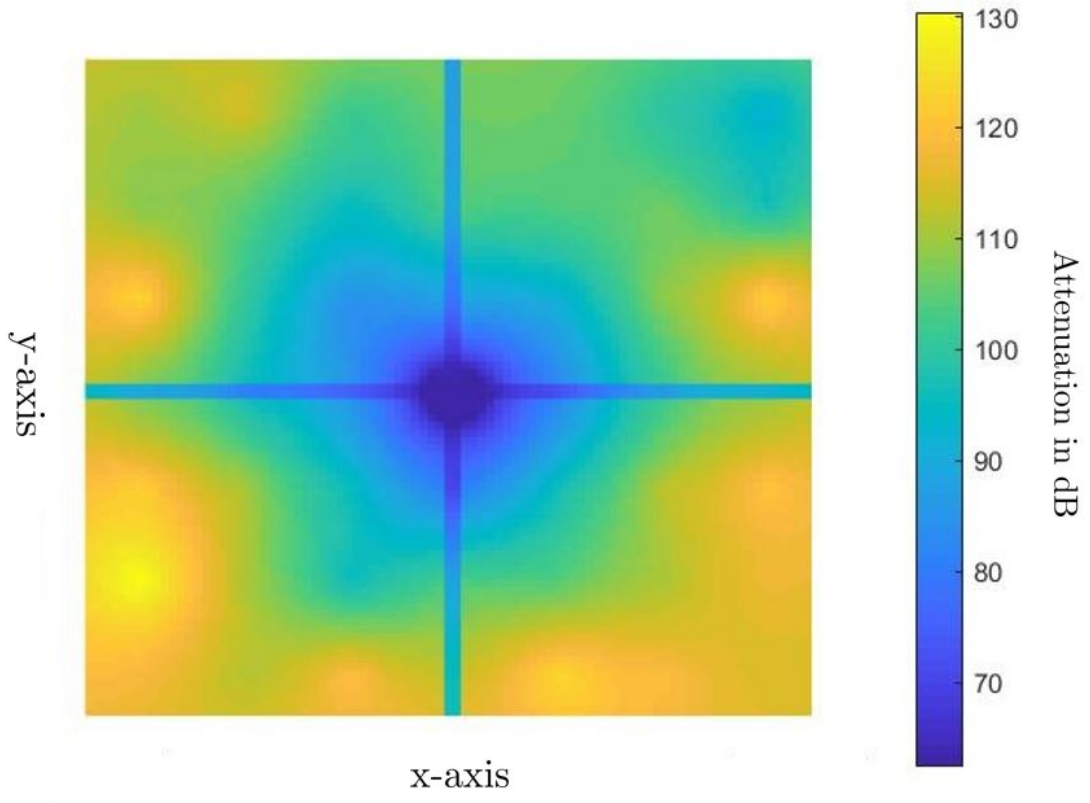


Figure 5.2: Attenuation map including path-loss and shadowing, with the BS positioned at the center.

5.1 Defense Implementations

All the models of defense are trained on a generated data set $\mathcal{S}$ with $M_{\mathcal{S}} = 320000$ samples and they are tested on a test set $\mathcal{E}$ with $M_{\mathcal{E}} = 100000$ samples. Each performance result is obtained by an average of 20 results in 20 environment realizations.

In order to compare the various defense methods we use the detection error (DET) where in the x-axis there is the P_{fa} in log scale, and in the y-axis there is the P_{md} in log scale.

The first implemented system is the MLP-IRLV. We use a library of Keras [28] for the implementation of the MLP. This model is the first chosen because it is simple to implement, fast to learn and in different applications of binary classification [29],[30] it gives good results.

The implemented MLP is composed by an input layer with $L^{(0)} = 10, 20 \cdot N_{ref}, 10 + 20 \cdot N_{ref}$. The

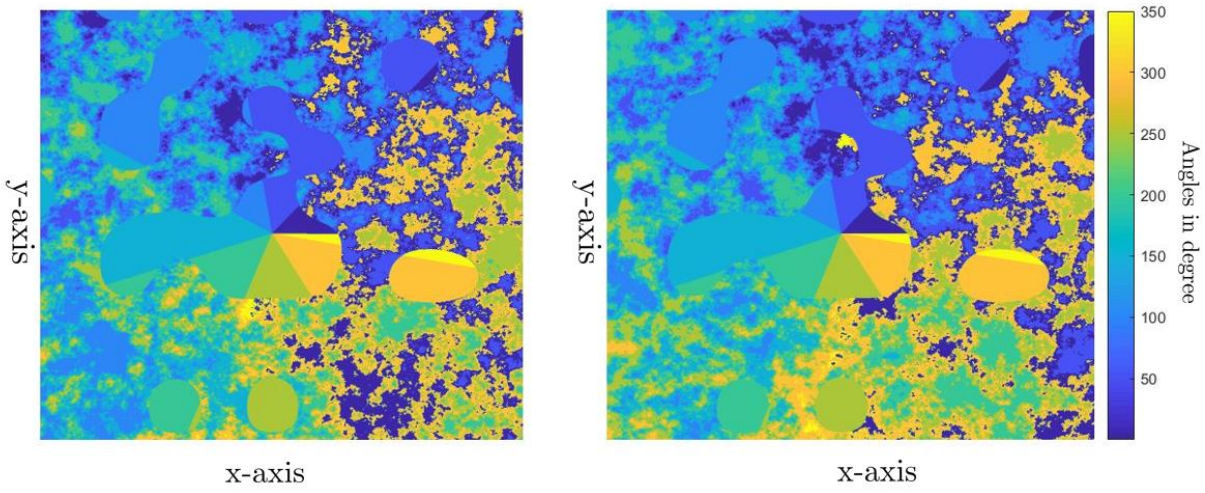


Figure 5.3: Map of arrival angle at left and map of departure angle at right.

dimension is 10 if we consider only the attenuation features, it is $20 \cdot N_{ref}$ if we consider only the angle features, and $10 + 20 \cdot N_{ref}$ if we use both types of feature together. The hidden layer is composed by three layers of one hundred nodes each. This choice of dimension had already been explained in the Section 3.1.1.

First, we test the model with $20 \cdot N_{ref}$ input nodes and try different value of N_{ref}, with uncorrelated and correlated features. The result observed with $N_{ref} = 1, 2$, and 3 can be seen in Fig.5.4. From this figure is possible to see that the best solution for the defense is obtained with $N_{ref} = 3$. This result could be expected because observing multiple angles of reflection of the same signal is easier to understand where it comes from.

The second test is done on the attenuation features, uncorrelated and correlated. In the Fig. 5.5, we plot the performance of the MLP-IRLV with the simulated samples in different conditions: when the only effect of path loss and shadowing (P+S) is applied (2.3) and when the path loss, shadowing and fading (P+S+F) is used (2.2). The best result is obtained when the fading effect is omitted, because the introduction of so much randomness with fading reduces the performance of the MLP. To improve the result in presence of fading we proposed to observe $N_{fading} = 10$ features from pilot signals coming from the same position in different times. We average these values, trying to reduce the randomness of fading. As we seen in the Fig. 5.5 the performance improves, but the

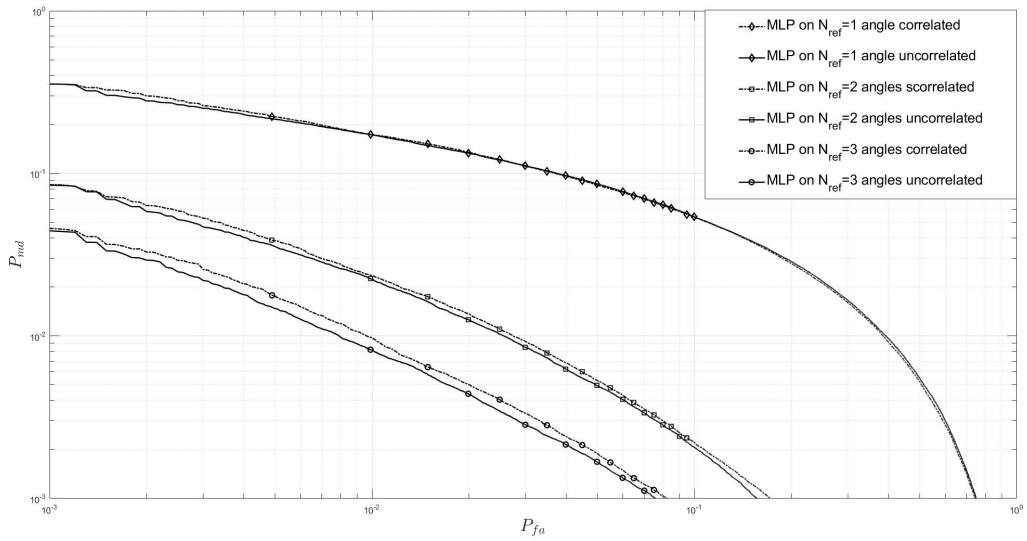


Figure 5.4: DET for MLP-IRLV with different number of angles features.

result is still far from that obtained with P+S. Unfortunately, this strategy is discarded because the time for collecting of N_{fading} different realizations with different fading values is excessive in practical applications.

The performance obtained on the features with the random variables correlation is a bit worst with respect to the case of uncorrelated features. This difference of results can be seen in Fig. 5.4 and 5.5. For simplicity, in all the next implementations, since the performance variation is not so significant, we consider the only the uncorrelated features.
Now, we show that the best results are obtained with MLP using both features of angles and attenuation together as shown in Fig. 5.6. Observing this plot, we decide, to use only the attenuation features, because the addition of the angles features greatly increase the complexity of the training without provides a significant improvement performance.
The second strategy used for implementation of IRLV is the SVC approach. In Fig. 5.7 we present the result of the SVC-IRLVs where training and testing is done on different Kernels, with effect of P+S in Fig. 5.7 and P+S+F in Fig. 5.8.

The SVC algorithm is implemented with sklearn.svm.SVC [31]. The best results are obtained by SVC with RBF Kernel and value $C_{hp} = 100000$. The performance of IRLV is suitable for our

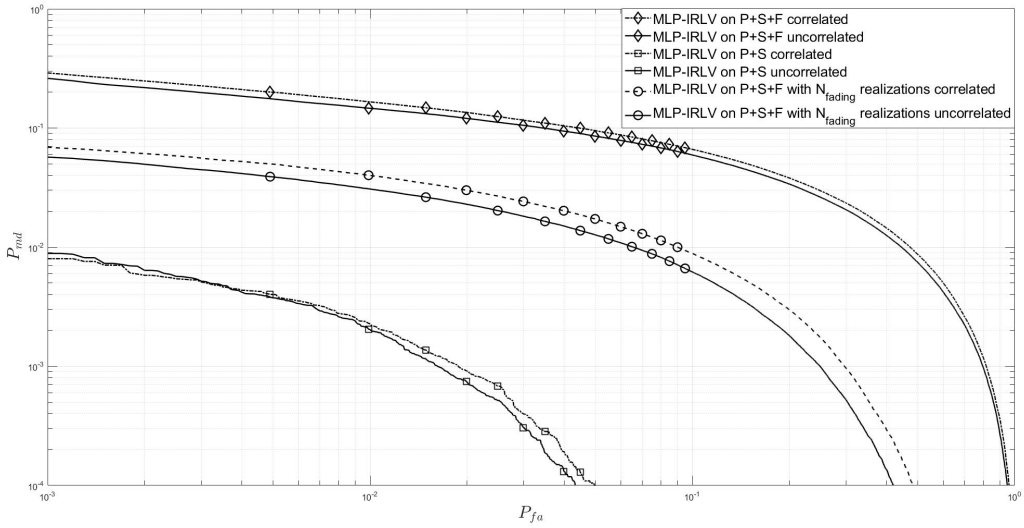


Figure 5.5: DET for MLP-IRLV with different attenuation features.

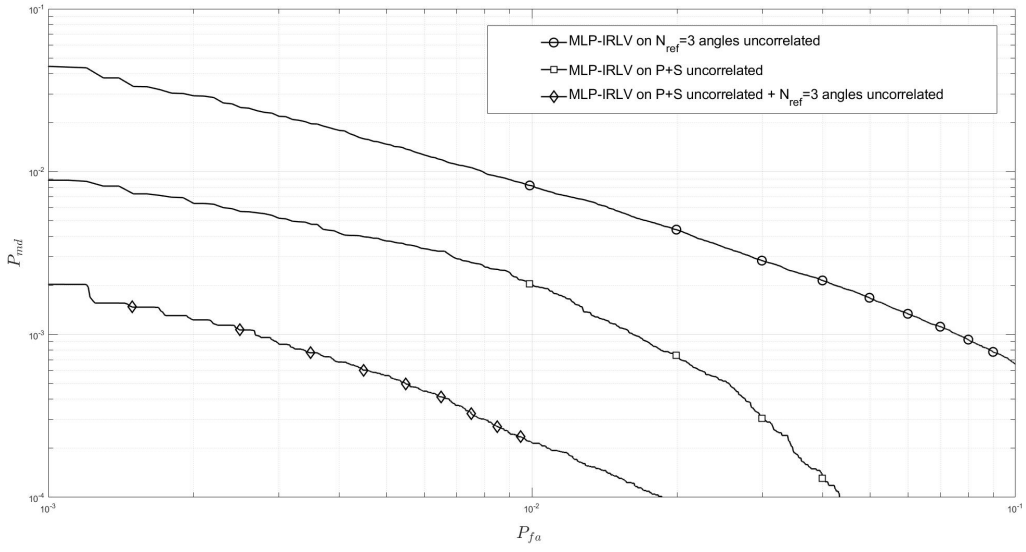


Figure 5.6: DET for MLP-IRLV with attenuation features, angles features and both types.

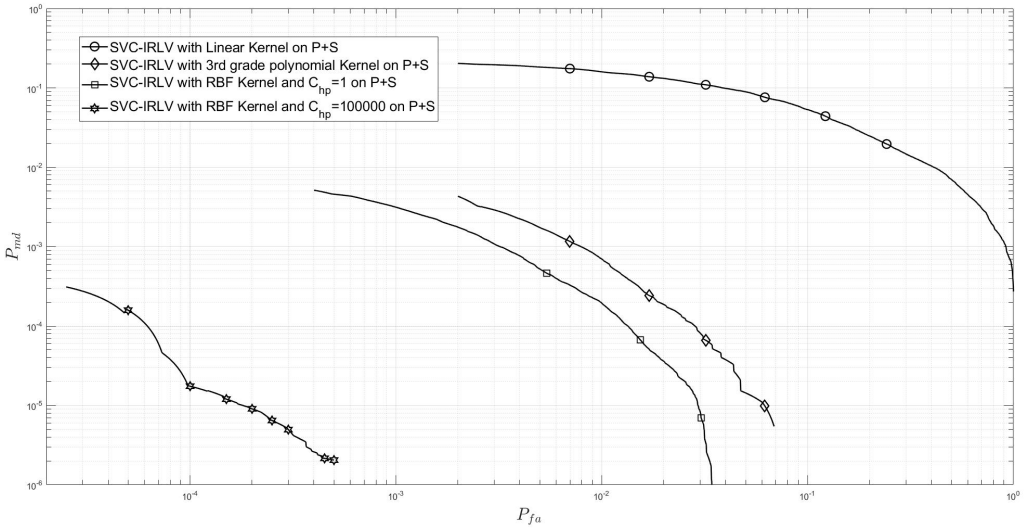


Figure 5.7: DET for SVC-IRLV on P+S channel with different Kernels.

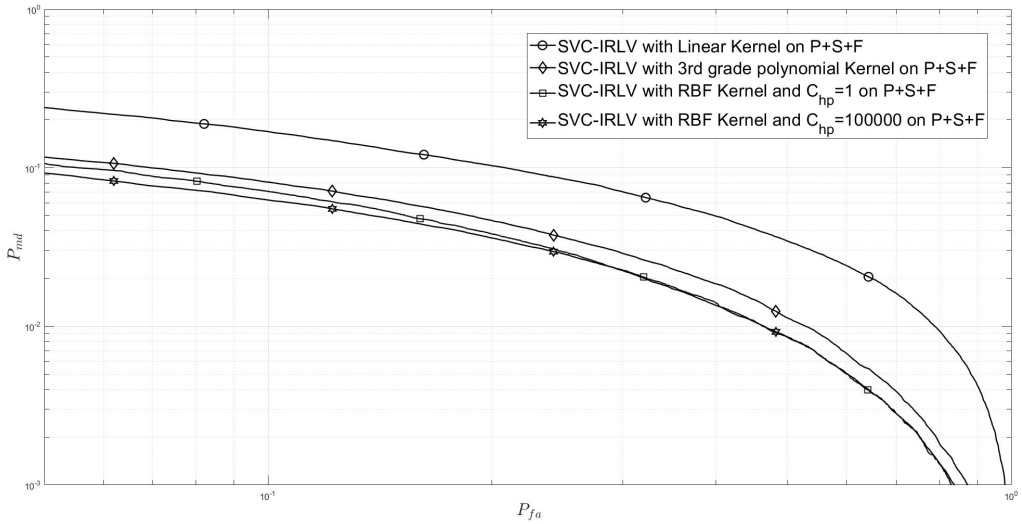


Figure 5.8: DET for SVC-IRLV on P+S+F channel with different Kernels.

problem and is able to correct classify the sample with an accuracy, in P+S+F case, of 10^{-6}. This is the best solution that we are able to obtain.

The performance of the defense implemented on P+S+F with MLP and SVC have similar results. Instead, observing the strong difference between SVC-IRLV and the MLP-IRLV in the P+S case, we asked ourselves what strategies could be used to improve the performance of the ANN. First, we tried to create IRLV by ANN with spline activation function model, called (ANN with Spline)-IRLV. In Fig. 5.9 we can see the performance of this algorithm. Instead, in Fig. 5.10 we can see two trained spline activation functions, implemented according to [22]. The obtained performance is better then the simple MLP but still not comparable to the best SVC. A motivation for this bad result it can be found in the difficulty of the ANN to properly use information obtained from the samples. Analyzing the samples used for training, we noticed that there are many similar values. For this reason we tested the SI , with $B = 10$ implementation as described in Section 4.1.2. The performance of this IRLV, called (SIR+ANN)-IRLV, can be seen in Fig. 5.9, where it is compared with the MLP-IRLV and (ANN with Spline)-IRLV. The performance is comparable to the result obtained with the (ANN with Spline)-IRLV.

In conclusion, our implementations through ANN have failed to achieve the performance of the best SVC but we must not forget that the SVM methods cannot be applied in all conditions because it is complex and requires a large memory.

Also, we have tested the proposed MLP-IRLV and SVC-IRLV solutions on real data collected by the MOMENTUM project [1] at Alexanderplatz in Berlin (Germany). The value of attenuations at the frequency of the global system for mobile communications (GSM) have been measured for several BSs in a square area of side 450 m, on a measurement grid of 50 m. We have considered 10 attenuation maps, corresponding to 10 BS positions (all in meters) $\boldsymbol{p}_1 = (2500, 2500)$, $\boldsymbol{p}_2 = (500, 4000)$, $\boldsymbol{p}_3 = (4000, 4000)$, $\boldsymbol{p}_4 = (500, 500)$, $\boldsymbol{p}_5 = (4000, 500)$, $\boldsymbol{p}_6 = (100, 4500)$, $\boldsymbol{p}_7 = (1000, 400)$, $\boldsymbol{p}_8 = (4000, 500)$, $\boldsymbol{p}_9 = (4300, 4000)$, $\boldsymbol{p}_{10} = (4500, 500)$. The authentication zone has been positioned in the lower-right corner, corresponding to, following the same notation of Fig. 5.1 $d_1 = 3000$ m, $d_2 = 1500$ m, and $\tilde{\beta}_1 = \tilde{\beta}_2 = 1000$ m. In this case, we have a single realization of any channel effect P+S+F per location, for a total of 8464 realizations, 5000 of which have been used for training and the rest for testing. Fig. 5.11 shows the DET for both MLP and SVC. Moreover, we notice that SVC and MLP achieve approximately the same performance.

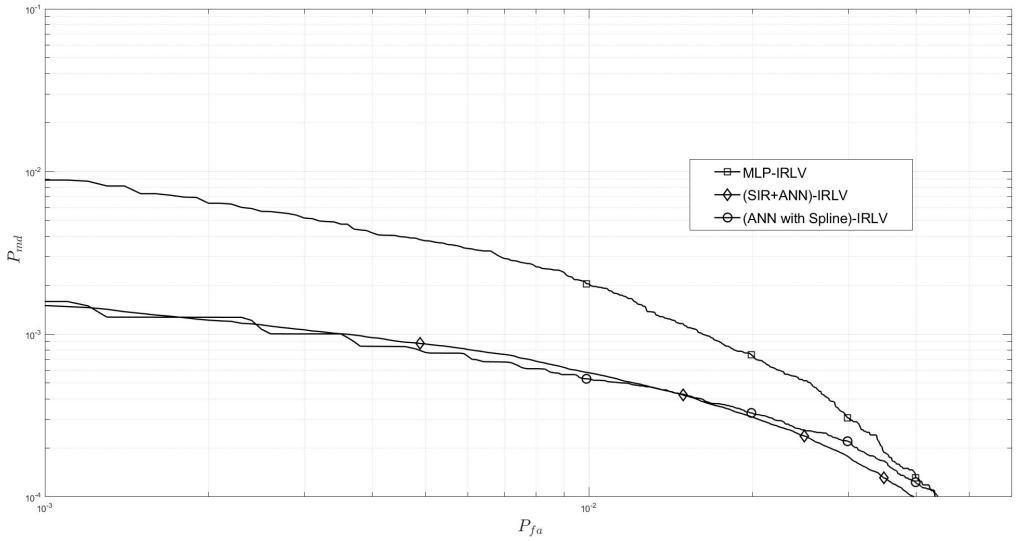
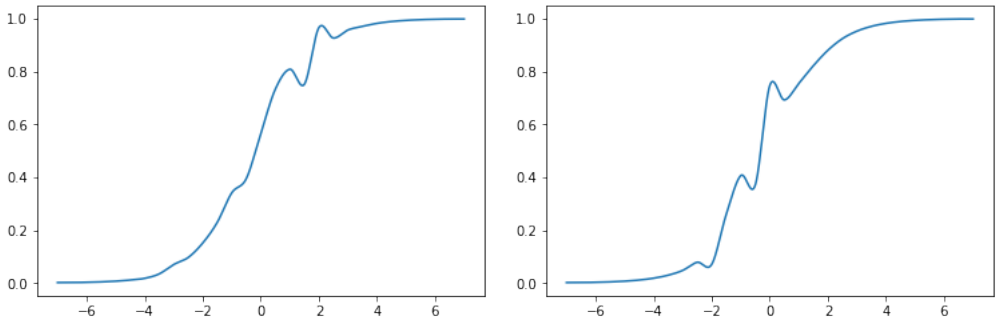


Figure 5.9: DET of different ANN for defense P+S attenuation channel models.



(a) (b)

Figure 5.10: Two trained Spline functions.

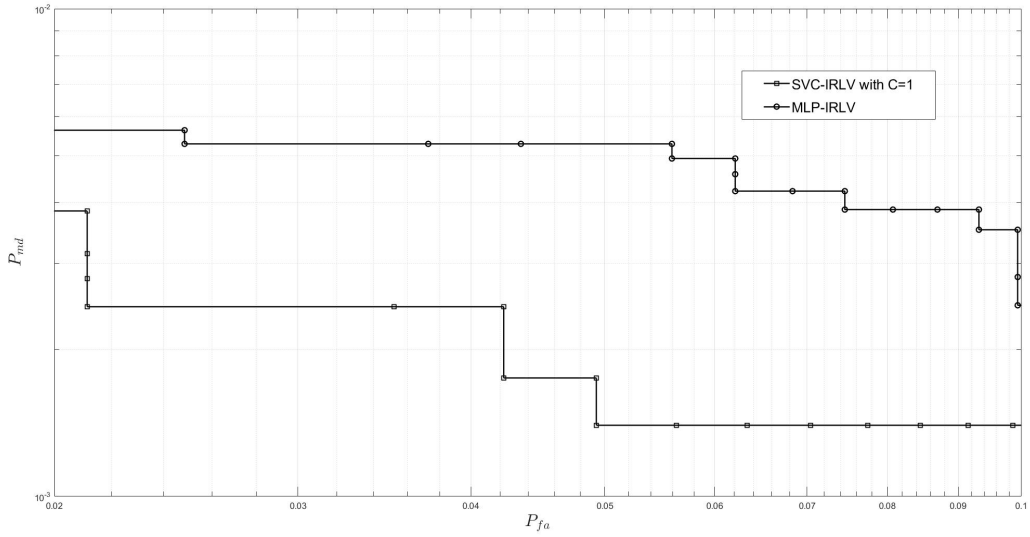


Figure 5.11: DET for SVC-IRLV and MLP-IRLV on data collected by MOMENTUM project [1].

5.2 Attack Implementations

In this section the attack models are tested against the MLP-IRLV and SVC-IRLV. The features used for the attack and the defense model is only obtained by RSS channel model that use A+S, because the defense models that used RSS channel model with A+S+F are too simple to be attacked and do not allow a good comparison between the various attacks methods. The dimension of the set $\mathcal{S}$ is $M_{\mathcal{S}} = 50000$ and the dimension of the set $\mathcal{E}$ is $M_{\mathcal{E}} = 10000$.

To compare the efficiency of the different attacks we need a comparable metric. We use the success probability (SP) within n_{attack} attacks, i.e., the probability that within n_{attack} attacks at least one is successful, which is denoted as $P_s(n_{attack})$. Note that the SP corresponds to the misdetection probability of the IRLV, under the various attack strategies. Each plot is the mean of one hundred attacks on different channel simulations.

The first tested attack is the most simple, i.e., the RA. Fig. 5.12 shows the $P_s(n_{attack})$ with respect to the number of attacks for the RA method. The SVC-IRLV reaches the best performance and therefore is omitted, instead the MLP-IRLV leaves many possibilities of success.

Instead of this simple and random selection of samples used we proposed the PRA method, where the sample are selected with a Ranking method. This solution gives a better result with respect to

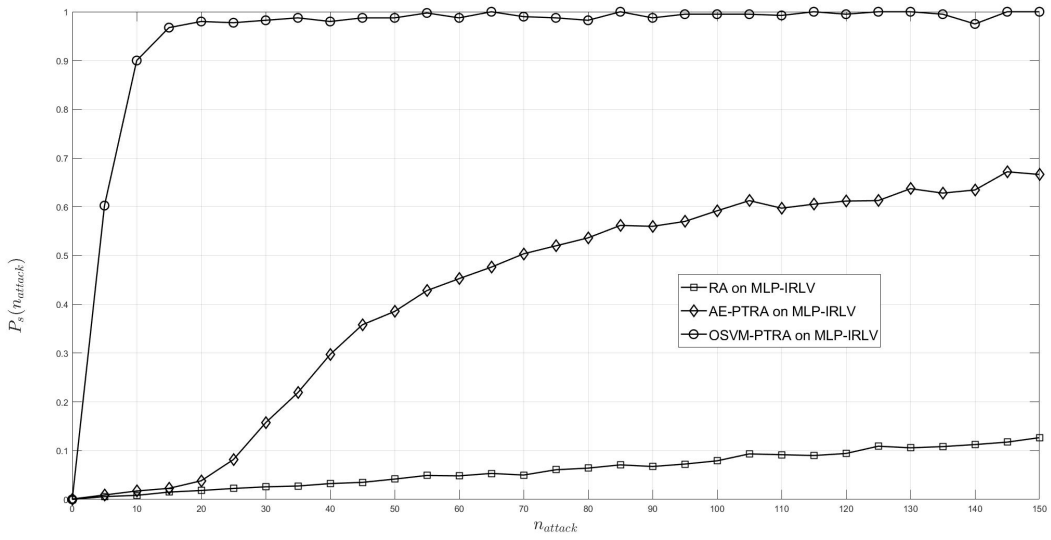
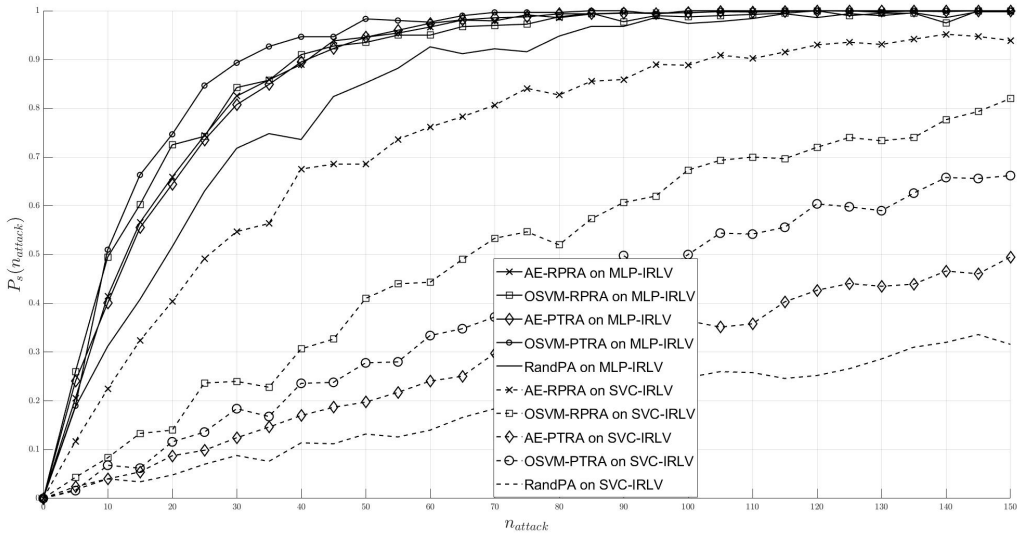


Figure 5.12: Comparison of passive attacks on MLP-IRLV.

the RA method for the MLP-IRLV. Instead, the SVC-IRLV is still impenetrable. Fig. 5.12 shows the result of the PRA against MLP-IRLV.

The attacks exposed above are passive, i.e., based on the selection of existing attenuation features. The models presented in Section 4.2.2 generate attenuation values that can not be obtained from the pilot signals in region A.
First we test PTRA in Fig. 5.13, where we see that this attack strategy outperforms all the passive attacks under both MLP and SVC defenses. It was the first attacks that have success on SVC defense. Continuing on this active attack strategy we tried to optimize the creation of the attack samples. We have implemented and tested all different active attacks as PTRA, RPRA, KTPA and KPRA, where elementary parts $\mathcal{R}$, $\mathcal{P}$, and $\mathcal{K}$ are mixed in various ways. The PS for every method of attacks is shown in the Fig. 5.13. The best result is obtained with RPRA.

The $P_s(n_{attack})$ with these attacks is very high, with IRLV implemented with both MLP and SVC. To improve the defense with respect to these attacks we added a ring with $\alpha_{ring} \neq 0$. We now investigate the effects of the ring R in reducing the effectiveness of the attacks. Fig. 5.14 shows the SP after 50 attacks for various passive-attack strategies, as a function of the normalized



(a)

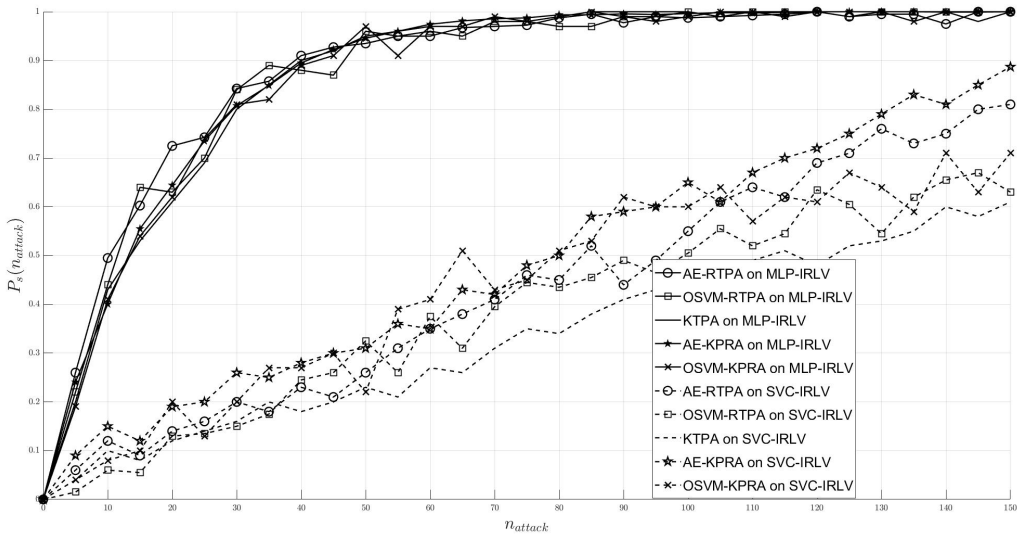


Figure 5.13: Comparison of active attacks on MLP-IRLV and SVC-IRLV.

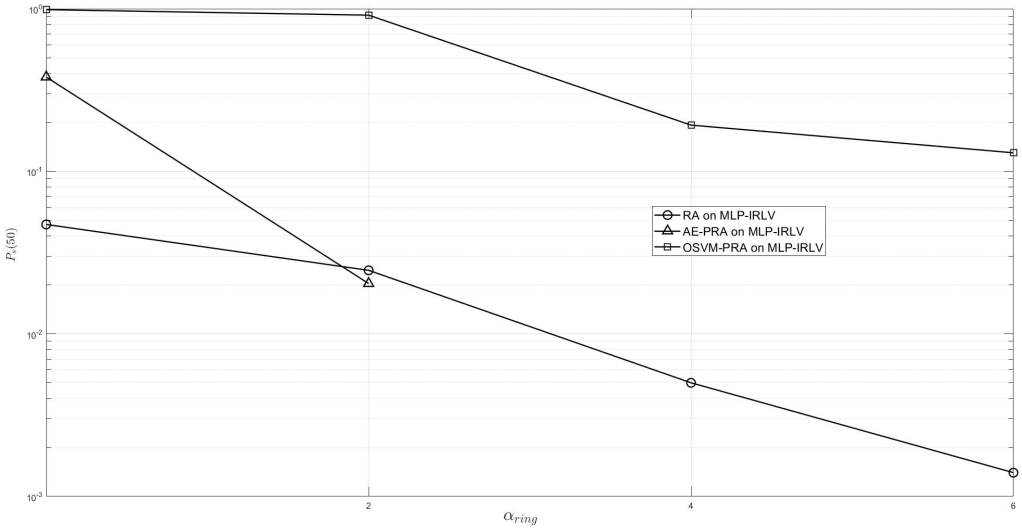


Figure 5.14: Comparison of passive attacks on MLP-IRLV with different value α_{ring}.

incremental area of the ring $\alpha_{ring} \in [0, 1]$. We note that $P_s(50)$ is rapidly decreasing with α_{ring}. Therefore, we conclude that the most effective attack positions are those close to A_0, which clearly have attenuation vectors similar to points inside A_0.

We have already seen that active attack strategies without ring are more effective than passive ones. We note that now the value of α must be larger than in the case of passive attacks, in order to achieve the same $P_s(50)$. Indeed, even with SVC-IRLV and $\alpha = 1$, $P_s(50)$ barely goes below 10^{-2}. Therefore, active attacks are really effective, also in the presence of a ring.

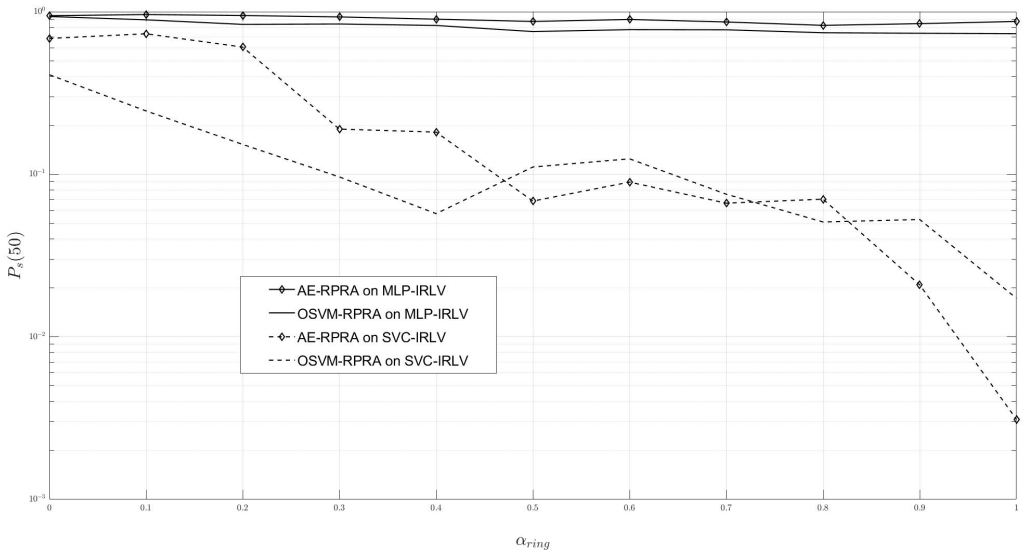


Figure 5.15: Comparison of active attacks on MLP-IRLV and SVC-IRLV with different value α_{ring}.

Chapter 6

Conclusions

In this thesis, we have proposed innovative solutions for IRLV in wireless networks that exploit the features of the channels between the UD whose location must be verified by a trusted network of BSs. By observing that in typical situations the channel statistics are not available for IRLV.
We have proposed ML-based solutions, using binary classification and ranking. We evaluated the performance of ML algorithms in terms of DEF curve on simulated propagation scenario and experimental data, as we can see in the Section 5.1. For both the cases the best results are obtained with the SVC-IRLV.
Also, we have considered various methods for attacking an IRLV system based on ML. All novel techniques significantly reduce the time of the first successful attack with respect to the existing random strategy. It is confirmed that SVC-IRLV is effective against passive attacks, whereas being more vulnerable to active attacks. The presence of a ring around the A_0 significantly improves the effectiveness of IRLV. Indeed, the MLP-IRLV is vulnerable at all types of attacks but becomes more resistant to passive attacks when the ring is added.
The concept algorithm we considered open new research opportunities in the ML framework, e.g., the ANN architecture can be further optimized (in the number of neurons, layers shape, and activation function type) for a better adaptation to different IRLV scenarios.

Appendices

Appendix A

Simulation Parameters

Table A.1: Value of parameters for RSS Channel model.

Parameter	Value
$\sigma_{s,dB}$	8 dB
f	2.12 GHz
d_c	75 m

Table A.2: Value of parameters for Angles Model.

Parameter	Value for arrival angle	Value for departure angle
c_{LoS}	17	3
c_{NLoS}	22	10
α_m	0.0447	0.0447
μ_{lg}	-0.08 · log10(1+f)+1.81	-0.23·log10(1+f)+1.53
σ_{lg}	0.05 · log10(1+f)+0.3	0.11·log10(1+f)+0.33

Table A.3: Value of parameters for the MLP applied on the generated values.

Parameter	Value
N_L	3
$L^{(1)}, L^{(2)}, L^{(3)}$	100
Optimizer	Adam
β_1	0.99
β_2	0.99
α_{L2}	0.0001
$\epsilon_{stability}$	0.001

Table A.4: Value of parameters for the MLP applied on real values [1].

Parameter	Value
N_L	3
$L^{(1)}, L^{(2)}, L^{(3)}$	500
Optimizer	LBFGS

Table A.5: Value of parameters for the ANN with Spline activation function.

Parameter	Value
N_L	1
$L^{(1)}$	100
N	14
Δ_x	0.5
$\tilde{x}$	7
μ_w	0.01
μ_q	0.005

Table A.6: Value of parameters of the implemented AE.

Parameter	Value
N_L	3
$L^{(1)}, L^{(2)}, L^{(3)}$	100
Optimizer	Adam
β_1	0.99
β_2	0.99
$\alpha L2$	0.0001
$\epsilon_{stability}$	0.001

Bibliography

[1] A. Eisenblätter, A. Fügenschuh, E. E. Fledderus, H.-F. Geerdes, B. Heideck, D. Junglas, T. Koch, T. Kürner, and A. Martin, "Mathematical methods for automatic optimisation of umts radio networks," 2003.

[2] U. S. C. Guard, "Global positioning system standard positioning service specification, 2nd edition," 1995.

[3] Y. Zeng, J. Cao, J. Hong, S. Zhang, and L. Xie, "Secure localization and location verification in wireless sensor networks: a survey," *The Journal of Supercomputing*, vol. 64, pp. 685–701, 2010.

[4] G. Caparra, M. Centenaro, N. Laurenti, and S. Tomasin, "Optimization of anchor nodes' usage for location verification systems," *2017 International Conference on Localization and GNSS (ICL-GNSS)*, pp. 1–6, 2017.

[5] T. P. T. Ioannides and G. Gibbons, "Know vulnerabilities of global navigation satellite systems, status, and potential mitigation techniques," *Proc. IEEE, vol. 104, no. 6, pp. 1174-1194*, Jun. 2016.

[6] C. Li, F. H. Chen, Y. Zhan, and L. Wang, "Security verification of location estimate in wireless sensor networks," *2010 6th International Conference on Wireless Communications Networking and Mobile Computing (WiCOM)*, pp. 1–4, 2010.

[7] P. Baracca, N. Laurenti, and S. Tomasin, "Physical layer authentication over mimo fading wiretap channels," *IEEE Transactions on Wireless Communications*, vol. 11, pp. 2564–2573, 2012.

[8] L. Xiao, H. Zhang, G. Han, G. Liu, and W. Zhuang, "Phy-layer spoofing detection with reinforcement learning in wireless networks," *IEEE Transactions on Vehicular Technology*, vol. 65, pp. 10037–10047, 2016.

[9] S. Brands and D. Chaum, "Distance-bounding protocols (extended abstract)," 1993.

[10] D. Singelée and B. Preneel, "Location verification using secure distance bounding protocols," *IEEE International Conference on Mobile Adhoc and Sensor Systems Conference, 2005.*, pp. 7 pp.–840, 2005.

[11] N. Sastry, U. Shankar, and D. A. Wagner, "Secure verification of location claims," 2003.

[12] J.-H. Song, V. W. S. Wong, and V. C. M. Leung, "Secure location verification for vehicular ad-hoc networks," *IEEE GLOBECOM 2008 - 2008 IEEE Global Telecommunications Conference*, pp. 1–5, 2008.

[13] A. Vora and M. Nesterenko, "Secure location verification using radio broadcast," 2004.

[14] S. Yan, I. Nevat, G. W. Peters, and R. A. Malaney, "Location verification systems under spatially correlated shadowing," *IEEE Transactions on Wireless Communications*, vol. 15, pp. 4132–4144, 2016.

[15] "Lte; evolved universal terrestrial radio access (e- utra); radio frequency (rf) system scenarios," *3GPP, TR 36.942 version 15.0.0 Release 15*, Jul 2018.

[16] B. Nevio and M. Zorzi, *"Principles of Communications Networks and Systems"*. John Wiley and Sons Inc, 1. edizione, 2011.

[17] "5g; study on channel model for frequencies from 0.5 to 100 ghz," *3GPP, TR 38.901 version 15.0.0 Release 15*, Jul 2018.

[18] D. P. Kingma and J. Ba *3rd International Conference on Learning Representations, "ICLR" , San Diego, CA, USA*, May 7-9, 2015.

[19] J. Nocedal, "Updating quasi-newton matrices with limited storage," *Math. Comp. 35, 773-782*, 1980.

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

[21] S. Shalev-Shwartz and S. Ben-David, *Understanding Machine Learning*. Cambridge University Press, 2014.

[22] N. E. N. W. Ahlberg, J. Harold and J. L., *The Theory of Splines and Their Applications*. New York: Academic Press, 1967.

[23] E. Catmull and R. ROM, *A class of local intepolating splines*. in Computer-Aided Geometric Design, R. E. Barmhill and R. F. Riensenfeld, Eds. New York: Academic Press, 1974.

[24] F. P. Stefano Guarnieri and A. Uncini, *Multilayer feedforward networks with adaptive spline activation function*. IEEE Transactions on Neural Networks, 1999.

[25] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *ICML*, 2008.

[26] "Deep inside: Autoencoders." `https://towardsdatascience.com/deep-inside-autoencoders-7e41f319999f`.

[27] C.-C. Chang and C.-J. Lin, "Libsvm: A library for support vector machines," *ACM TIST*, vol. 2, pp. 27:1–27:27, 2005.

[28] "Keras: The python deep learning library." `https://keras.io`.

[29] M. D. Matino, S. Fanelli, and M. Protasi, "An efficient algorithm for classification of patterns using mlp-networks," *IEEE International Conference of Neural Networks*, 1993.

[30] G. Latif, M. M. Butt, A. H. Khan, M. O. Butt, and J. F. Al-Asad, "Automatic multimodal brain image classification using mlp and 3d glioma tumor reconstruction," *2017 9th IEEE-GCC Conference and Exhibition (GCCCE)*, pp. 1–9, 2017.

[31] "C-support vector classification." `https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html`.