INDOOR POSITIONING SYSTEM

A MINI PROJECT REPORT

18CSC305J - ARTIFICIAL INTELLIGENCE

Submitted by

BODDETI MUSILI NAIDU[RA2111003010624]
GUDALA PAVAN SURYA [RA2111003010661]
PERANDURU VITHESH VARUN[RA2111003010665]

Under the guidance of

Dr. G.Ramya

Assistant Professor, Department of Computer Science and Engineering

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

of

FACULTY OF ENGINEERING AND TECHNOLOGY

S.R.M. Nagar, Kattankulathur, Chengalpattu District

MAY 2024

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that Mini project report titled **"INDOOR POSITIONING SYSTEM"** is the Bonafide work of **BODDETI MUSILI NAIDU [RA2111003010624], GUDALA PAVAN SURYA [RA2111003010661], PERANDURU VITHESH VARUN [RA2111003010665]** who carried out the minor project under my supervision. Certified further, that to the best of my knowledge, the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Dr.G. Ramya

Assistant Professor

Department of Computing Technologies

ABSTRACT

KEYWORDS: Indoor Positioning, Received Signal Strength Indicator, Access Points, Feed Forward Neural Network

Indoor Positioning is a technique that helps us to get the knowledge about the location of a person or an object in indoor environments. This project is dedicated towards the implementation of one such Indoor Positioning technique which makes the use of Received Signal Strength Indicator values from different Access Points (Wi-Fi routers) and a Feed Forward Neural Network to predict the location of a person. As part of the project, the location of the person in one room and in three adjacent rooms is predicted. The error in the prediction is plotted against various positions of the Access Points. Considering any one of these positions of the Access Point, the variation in the error is observed for different number of layers and for different number of neurons per layer in the neural network, for different number of indoor locations, for different standard deviations of the shadowing noise etc. Finally, necessary conclusions are derived from the plots to see what can be an optimal prediction when Indoor Positioning is implemented in bigger indoor environments.

This report provides a detailed procedure of development of a model and its implementation for indoor positioning system using Bluetooth Low Energy. The main conclusion of this report is that BLE is a potential alternative to traditional localization techniques and it can be used very efficiently for indoor localization.

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS

MU – Mobile User

FNN – Feedforward Neural Network

AP – Access Point

ADE – Average Distance Error

SD – Standard Deviation

BLE – Bluetooth Low Energy

RSSI – Received Signal Strength Indicator

WPS – Wi-Fi based Positioning System

RFID – Radio Frequency Identification

IPS – Indoor Positioning System

GPS – Global Positioning System

CC2540EM – CC2540 Evaluation Modules

smartRF05EB – smartRF05 Evaluation Board

SOC_BB – System on Chip Battery Board

RF – Radio Frequency

PC – Personal Computer

USB – Universal Serial Bus

SPI – Serial Peripheral Interface

UART – Universal Asynchronous Receiver/Transmitter

csv – Comma Separated Values

TI – Texas Instruments

GPIO – General Purpose Input / Output

SRAM – Static Random Access Memory

IDE – Integrated Development Environment

GHz – Giga Hertz

Mbps – Megabits per second

KB – KiloBytes

AoA – Angle of Arrival

CHAPTER 1
INTRODUCTION

1.1 LOCALIZATION

Localization is the process of finding the exact position of an object in any environment. Depending on the environment under consideration, localization can be broadly classified into two categories:

- Outdoor localization
- Indoor localization / Indoor Positioning System (IPS)

One of the most popular localization techniques is GPS which is readily available now-a-days. The fundamental question therefore is, If GPS is a readily available, what is the need for new localization techniques?

1.2 SIGNIFICANCE OF INDOOR POSITIONING SYSTEM

GPS signals are unavailable in indoor environments such as railway stations, malls, airport etc. due to its large margin of error. This is where indoor localization comes into significance. IPS can be used in situations where GPS technologies fail. In indoor environments, where GPS signals are likely to be blocked, use of radio frequency signals is a very efficient method.

1.3 CURRENT TECHNOLOGIES FOR INDOOR POSITION SYSTEM

1.3.1 WI-FI BASED POSITIONING SYSTEM (WPS)

WPS is widely used technology for indoor localization. Wi-Fi modems and MAC address of the access point can be used to find the neighborhood of the object. The major drawback of such a method is

1.4 BLUETOOTH LOW ENERGY (BLE)

Bluetooth is a wireless technology that enables short range wireless communication among devices. Bluetooth Low Energy is a new specification of Bluetooth developed by Bluetooth Special Interest Group (SIG). BLE is version 4.0 of Bluetooth technology. It is also known as Bluetooth 4.0 or Bluetooth Smart. The major advantage of BLE over classic Bluetooth is its low power consumption and large battery life. The specialty of BLE is that the devices will be active only when data is getting transmitted and inactive otherwise. In this way, the device consumes less power.

1.5 CLASSICAL BLUETOOTH VS BLUETOOTH LOW ENERGY – A COMPARISON

SPECIFICATION	CLASSICAL BLUETOOTH	BLUETOOTH LOW ENERGY
Frequency Band	2.4 GHz	2.4 GHz
Data rate	1 to 3 Mbps	1 Mbps
Range	Up to 10 m	Up to 40 m
Power consumption	Low	Very low
Peak current consumption	<30 mA	<15 mA
Minimum time to send data	100 ms	3 ms
Battery life	Multiple weeks	Multiple months
Cost	Medium	Low
Accuracy	2 – 5 m	1 – 2 m
Developed for positioning	No	Yes

Table 1: Comparison of Classical Bluetooth and BLE

Wi-Fi based Indoor Positioning Systems (IPS) have emerged as the primary alternative to GPS for indoor localization due to the widespread availability of Wi-Fi network infrastructures in indoor environments, coupled with the ubiquity of Wi-Fi enabled consumer off-the-shelf (COTS) smartphones. There are three fundamental methods for computing the location of a Mobile User (MU) within indoor environments: (a) Time of Arrival (ToA)/Time Difference of Arrival (TDoA), (b) Angle of Arrival (AoA), and (c) Received Signal Strength (RSS). ToA/TDoA relies on measuring the time flight of signals, necessitating clock synchronization between Wireless Access Points (WAP) for implementation. AoA similarly relies on precise timing measurements and requires additional hardware for accurate implementation.

In contrast, RSS utilizes pattern matching algorithms to compute the location, with RSS fingerprinting being one of the most commonly used techniques. This method involves two phases: an offline phase where RSS from all APs is recorded at various locations, generating a location matrix with the AP fingerprint at each specific location; and an online phase where pattern matching algorithms compare the observed RSS by the MU with that recorded in the location matrix to calculate the MU's location.

In the current implementation, a Feedforward Neural Network (FNN) is utilized, employing optimization algorithms and backpropagation techniques to estimate the MU's location.

1.6 MERITS AND DEMERITS OF A NEURAL NETWORK USED FOR INDOOR POSITIONING

When there is noise in the indoor environment, it causes an error in the measurement of RSS. This affects the accuracy of the positioning performed by the FNN. On the other side, in a noiseless environment, an FNN for IPS is credited for the following:

- Powerful learning and adaptive capabilities
- Ability to map the nonlinearity between RSS signals and the position coordinates of an object
- Parallel distributed processing
- Data fusion
- Multivariable structure

CHAPTER-2
LITERATURE SURVEY

HARDWARE AND SETUP

2.1 HARDWARE COMPONENTS

Main components used are

- CC2540 evaluation modules
- smartRF05 evaluation board
- System on Chip Battery Board (SOC_BB)

2.1.1 CC2540 module / Beacons

CC2540 is a BLE module developed by Texas Instruments specifically for BLE applications. They are also called BEACONS. CC2540 system-on-chip has a high-performance and low-power 8051 microcontroller with an in-system-programmable flash memory of 128/256 KB, a SRAM memory of 8 KB, 21 GPIO ports, and other powerful features. CC2540 modules are connected via Bluetooth and they transmit and receive the radio frequency signals. They have an antenna connection for amplification of RF signals.

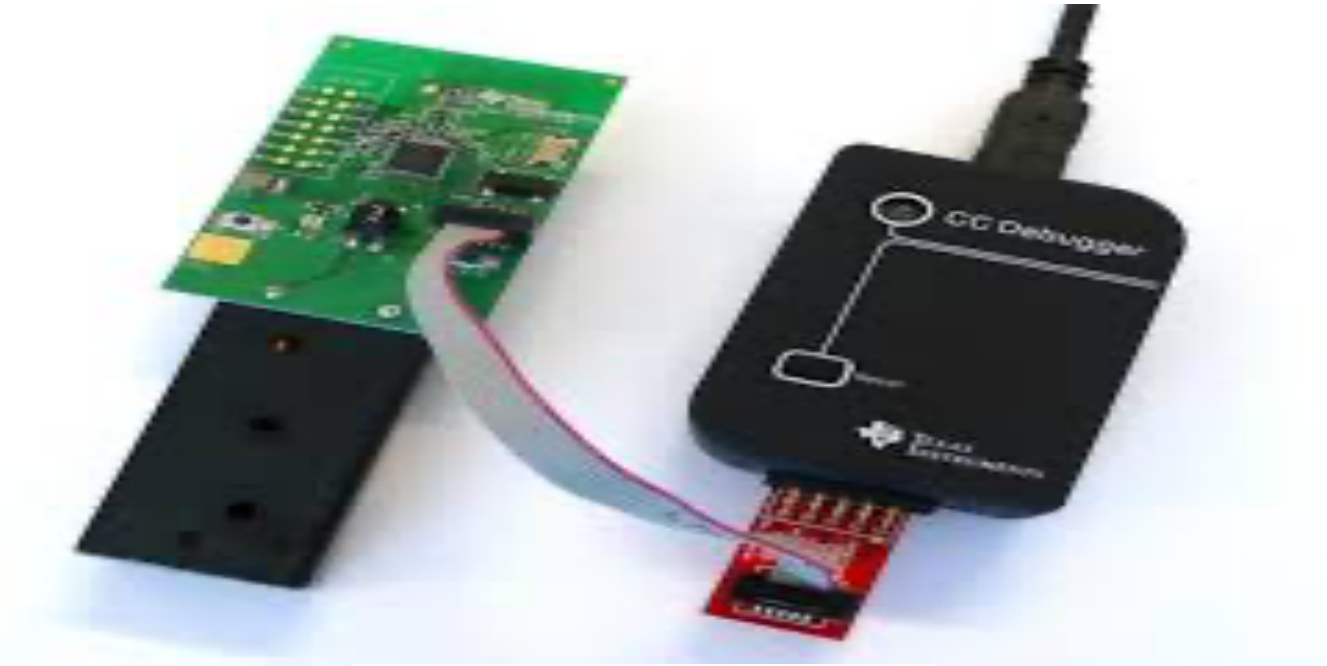

Figure-1 : CC2540 Module

2.1.2 smartRF05 evaluation board

smartRF05 evaluation board is used as a medium of communication between the beacons and the PC. Two major functions of smartRF05EB are to:

- Download the program into the beacon from the IDE.
- Serial communication between the beacon and PC

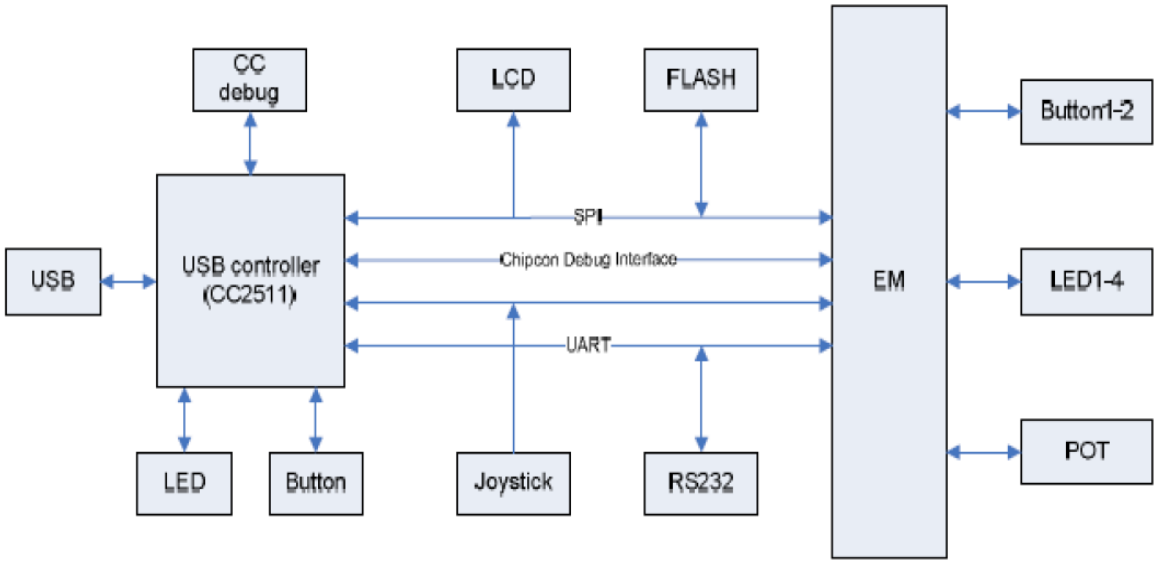

Figure – 2: smartRF05EB block diagram

The board can communicate with the PC in four possible ways:

- USB controller and USB connection to PC
- Debug interface
- SPI communication
- UART RS-232 communication

  In this case, UART serial communication is used using RS-232 interface and DB-9 serial cable. USB cable is used to power the CC2540EM through smartRF05EB.

CHAPTER 3
ARCHITECTURE SURVEY
DATA REPRESENTATION FOR INDOOR POSITIONING

3.1 When there is a single room

3.1.1 Setup environment

Consider a cubical room with one of its edges as 10 units. There are no restrictions in selecting the shape of the room and the room of any shape is perfectly suitable for carrying down the simulation. It is just that the cube has uniform dimensions and perfect symmetry in all the directions that led us to narrow down to this shape. The cubical room with different faces of it marked from 1 to 6 is shown below.

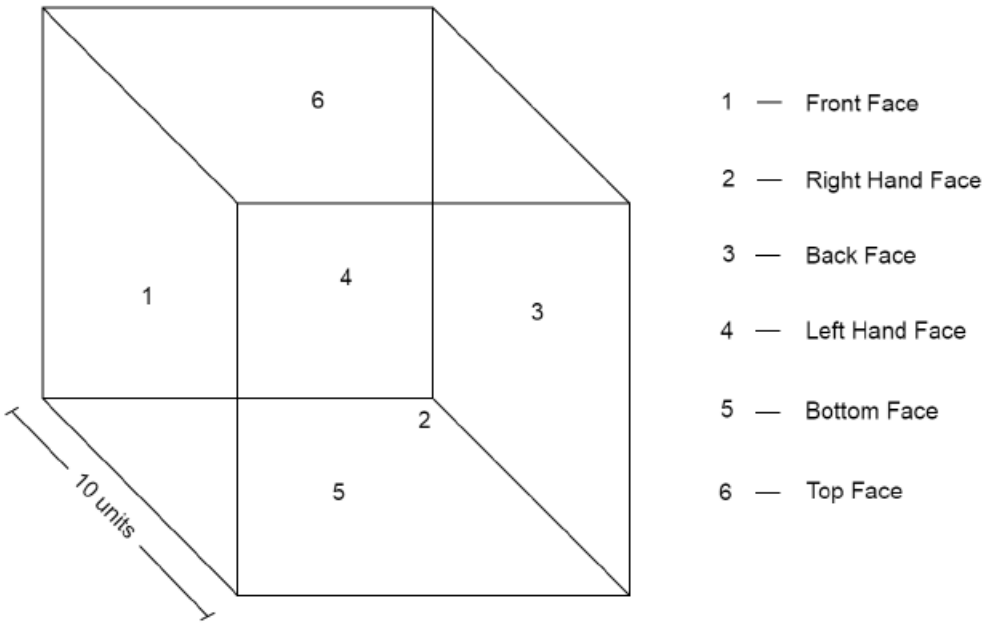

Figure 3: A 10 unit cubical room with its numbered faces

An Access Point (AP), which is nothing but a Wi-Fi router, is a device that keeps transmitting radio waves from its antenna when it is in the ON state. These APs are placed at different positions inside the room. Based on the accuracy of the tracking, the number of routers deployed and the positions at which they are deployed can be varied. The specifications of all the APs that are used in the setup are:

- Power of the transmitting antenna = 24 dBm

A Mobile User (MU) is the one who is inside the room and carries a device that is capable of receiving the radio signals transmitted by the AP. The specification of the device carried by the MU in the setup is:

Gain of the receiving antenna in the device = 6 dBi (The device with the MU is assumed to have an antenna which is similar to that of the transmitter)

At each location inside the room, the power received is different. This emphasizes the fact that location is one of the most important aspects for sensing the power variation which helps us get some useful information for indoor tracking. Therefore, for better processing of the locations, each location is assigned an ordered triple from the real space (note that the values in the ordered triple range only from 0 to 10 as the dimensions of the room restrict so). Cartesian Coordinate System is employed for assigning the ordered triples to all the interested locations inside the room. The origin of the coordinate system is taken at one of the corners of the room, preferably at the intersection of the 1st, 4th, and 5th faces of the cube. The X-axis is taken along the intersection of the first face and the fifth face, the Y-axis is taken along the intersection of the fourth face and the fifth face, and the Z-axis is taken along the intersection of the first face and the fourth face. A sample setup with APs placed on the edges of the three axes is shown below.

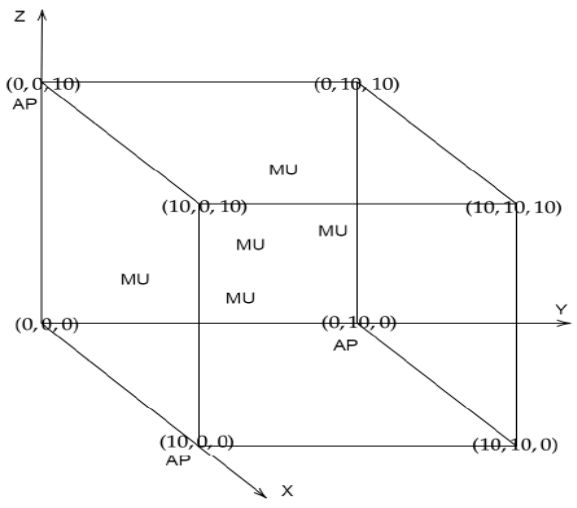

Figure 4: Setup for tracking the MU when there is a single room

3.1.2 Structure of the output data

Different MU locations are randomly selected inside the room. The selection is in such a way that the MU locations spread over the entire room. Depending upon the dimensions of the room, the number of MU locations selected can be varied; i.e., if a larger room is considered, the number of MU locations selected can be increased. A matrix is formed wherein each row corresponds to a 3-D MU location inside the room. We call this the Coordinate Matrix and denote it by C. This also forms our output data matrix. The first, second, and third elements of each row in this matrix correspond to the x, y, and z coordinates of the MU location respectively.

A general Coordinate Matrix is shown below.

$$C = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \\ \vdots & \vdots & \vdots \end{bmatrix}_{N_{MUL} \times 3}$$

where NMUL is the number of MU locations.

In the above matrix, the first row i.e., [c11 c12 c13] corresponds to the first MU location with c11 being its x-coordinate, c12 being its y-coordinate, and c13 being its z-coordinate. The second row i.e., [c21 c22 c23] corresponds to the second MU location with c21 being its x-coordinate, c22 being its y-coordinate, and c23 being its z-coordinate.

For the setup discussed in the above section, a sample Coordinate Matrix for 1000 MU locations is shown below.

$$C = \begin{bmatrix} 6.28858583 & 5.95822446 & 3.35333601 \\ 5.23703063 & 0.41524339 & 3.90588532 \\ 8.78011019 & 4.91266992 & 7.04965456 \\ \vdots & \vdots & \vdots \\ 4.12338296 & 4.05170427 & 7.51914268 \\ 6.98194044 & 2.14159756 & 3.69641588 \\ 5.16197382 & 7.77066492 & 2.57788852 \end{bmatrix}_{1000\times 3}$$

Note

3.2 EXPERIMENTAL SETUP

For the process of fingerprinting, an offline phase of the method is to be carried out by dividing the whole room into small areas with respective numeric positions. A rectangular part of our lab was divided into a 2x3 grid i.e. into 6 data points. Four slave beacons are put on the top at all the four corners of the experimental space

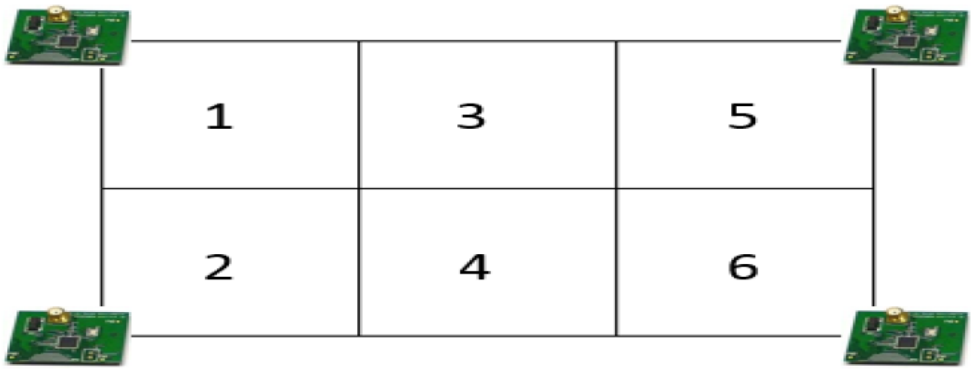

Figure - 5

3.3 Architecture of the Feedforward Neural Network

An Artificial Neural Network is a system made up of several simple and highly interconnected computing elements which process information by their dynamic state response to the external inputs. If the interconnection is directed from input to output and acyclic (meaning that there are no feedback connections or loops), then it is called a Feedforward Neural Network (FNN). The computing elements in the FNN are called nodes. These nodes are stacked to form columns which are called layers. The FNN has one input layer, one output layer, and one or more hidden layers. The number of nodes in a layer is called the size of the layer. Each node in a layer (except the output layer) is connected to every other node in the next layer and these connections are assigned some values (real numbers) which are called Weights. There are some special nodes present at almost each and every layer except the output layer in the FNN. They are special in the sense that they are connected only to the nodes that are present in the immediate next layer of it but not to that of present in the previous layer of it. They are called bias nodes. The value of the bias nodes is always 1. The bias nodes are represented as bn1, bn2, bn3 ... which are stacked under the input layer, first hidden layer, second hidden layer ... respectively. The entire data processing in the FNN takes place in the form of weight manipulations. The processing of data and the role of bias are discussed in detail in the following sections.

$$P_r = \begin{bmatrix} p_{r11} & p_{r12} & p_{r13} & \cdots & \cdots & \cdots \\ p_{r21} & p_{r22} & p_{r23} & \cdots & \cdots & \cdots \\ p_{r31} & p_{r32} & p_{33} & \cdots & \cdots & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}_{N_{MUL} \times N_{AP}}$$

Each element in the first row of the power matrix is fed to a neuron in the input layer of the FNN. Therefore, the size of the input layer in the FNN is equal to the number of columns in the power matrix room, i.e., NAP. Based on the computational circuitry and the accuracy of the tracking, the number of hidden layers and the size of each hidden layer can be varied. Say NH1, NH2, NH3 ... are the sizes of the first hidden layer, second hidden layer, third hidden layer ... respectively. After the size of each hidden layer is decided, the weights are generated in the form of matrices. Let the first weight matrix which is between the input layer and the first hidden layer be denoted as W(1), the second weight matrix which is between the first hidden layer and the second hidden layer be denoted as W(2), the third weight matrix which is between the second hidden layer and the third hidden layer be denoted as W(3) ... . The orders of the weight matrices W(1), W(2), W(3) ... are NAP x NH1, NH1 x NH2, NH2 x NH3 ... .

The generalized first, second, and third weight matrices are shown below.

$$W^{(1)} = \begin{bmatrix} w_{11}^{(1)} & w_{21}^{(1)} & w_{31}^{(1)} & \dots & \dots & \dots \\ w_{12}^{(1)} & w_{22}^{(1)} & w_{32}^{(1)} & \dots & \dots & \dots \\ w_{13}^{(1)} & w_{23}^{(1)} & w_{33}^{(1)} & \dots & \dots & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}_{N_{AP} \times N_{H1}} \qquad W^{(2)} = \begin{bmatrix} w_{11}^{(2)} & w_{21}^{(2)} & \dots & \dots & \dots \\ w_{12}^{(2)} & w_{22}^{(2)} & \dots & \dots & \dots \\ w_{13}^{(2)} & w_{23}^{(2)} & \dots & \dots & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}_{N_{H1} \times N_{H2}}$$

The main purpose of the FNN is to take the input power vector, process it, and generate the location of the MU inside the room. As the output is a location which is an ordered triple, it is obvious that the number of nodes in the output layer is equal to three. The first node is for determining the x-coordinate, the second node is for determining the y-coordinate, and the third node is for determining the z-coordinate of the MU location. The number of rows and the number of columns in the last weight matrix which is between the last hidden layer and the output layer are equal to the size of the last hidden layer and the size of the output layer respectively.

3.4 Forwarding operation

Forwarding operation is the process in the FNN by which an output is generated for a given input. Except the nodes in the input layer, all the other nodes participate in the forwarding operation.

The forwarding operation can be explained in three steps.

Step 1: Vector multiplication

Each node participating in the forward operation does a dot product between the nodes in the previous layer and a selected column from the corresponding weight matrix. The weight matrix and the column selected in the weight matrix for the dot product depend on the position of the hidden layer and the position of the node in that hidden layer. For example, the first node in the first hidden layer performs a dot product between the input nodes and the first column of the first weight matrix, the second node in the first hidden layer performs a dot product between the input nodes and the second column of the first weight matrix, the third node in the first hidden layer performs a dot product between the input nodes and the third column of the first weight matrix, the first node in the second hidden layer performs a dot product...

The whole process of predicting the location of a MU can be visualized as the surface fitting between the input power vector and the output coordinate matrix. The best fit is obtained when the standardized error (Mean Square Error) from the surface to all the points is minimum. Therefore, for the best fit, it is required that the surface is oriented in the right direction. The vector multiplication exactly does this! It orients the surface in the best possible direction for minimum error.

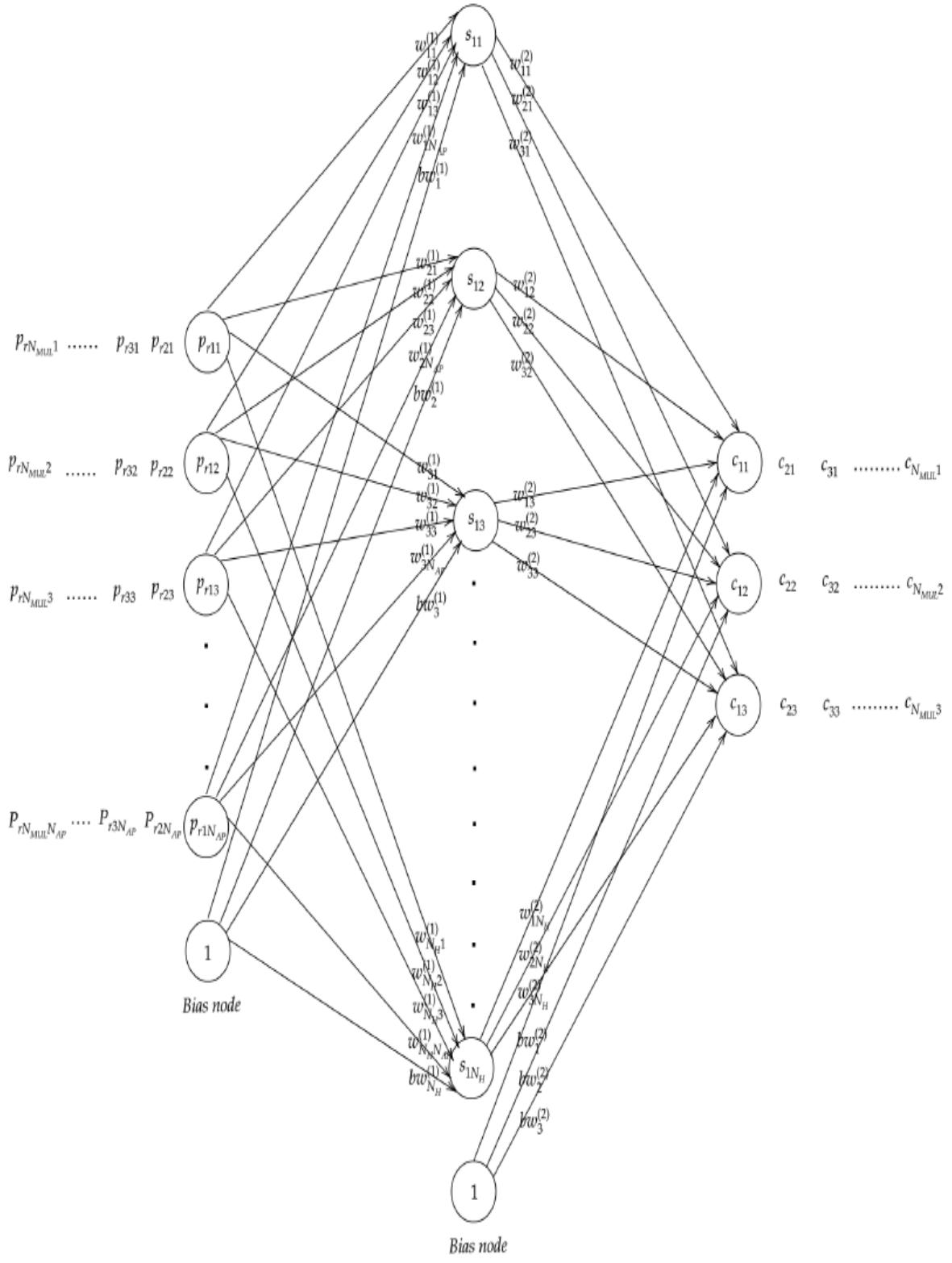

Figure – 6: A Feedforward Neural Network with one hidden layer

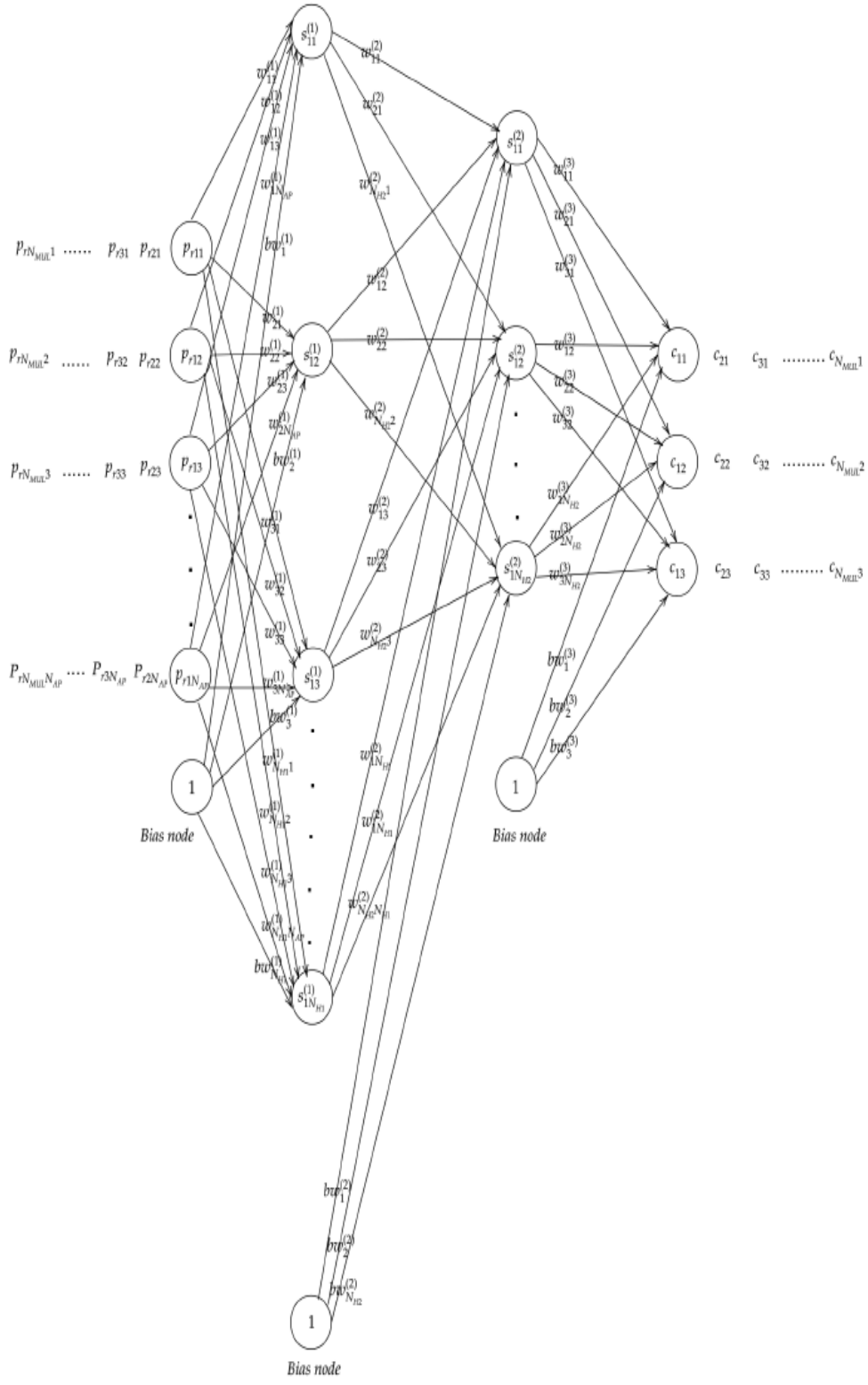

Figure – 7 : A Feedforward Neural Network with two hidden layers

Step 2: Bias addition

The resultant dot product obtained at each node is added with the bias value multiplied by the weight over the connection between the respective node and the bias node. For example, the dot product that is obtained at the third node in the fourth hidden layer is added with the bias value that is multiplied by the weight that is there over the connection between the third node in the fourth hidden layer and the fourth bias. As the values of the bias nodes are always 1, the weights that are there over their connections directly get added to the calculated dot products.

Sometimes with just the orientation, the fitting may be incomplete as the surface also needs the right position among the data points. The addition of the weighted bias node value exactly does this! It helps to locate the curve in the right position. It is important here to emphasize that positioning the surface and orienting the surface cannot be two independent tasks. They are interdependent and they keep taking place simultaneously until the best fit is obtained.

Step 3: Transformation using an activation function

At each node in the FNN, the result obtained from the above two steps is transformed to generate a new value. The function through which the transformation takes place is called an activation function. For the present implementation of IP, sigmoid is used as the activation function. The mathematical form of the sigmoid function is shown below:

$$F(x) = 1/(1+e^{-x})$$

Activation functions are really important for an FNN to learn and make sense of something really complicated. They introduce non-linear properties to the network. If an activation function is not applied, then the output signal would simply be a simple linear function. A linear function is just a polynomial of one degree. It is easy to solve but it is limited in its complexity and has less power to learn complex functional mappings from the data. We want the FNN to not just learn and compute a linear function but something more complicated than that. The main reason for using sigmoid as the activation function is because it exists only between 0 and 1. It is useful for the current model wherein the data that is processed is also normalized between 0 and 1.

$$\begin{bmatrix} p_{r11} & p_{r12} & p_{r13} & \dots & \dots & \dots \\ p_{r21} & p_{r22} & p_{r23} & \dots & \dots & \dots \\ p_{r31} & p_{r32} & p_{33} & \dots & \dots & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}_{N_{MUL} \times N_{AP}} \begin{bmatrix} w_{11}^{(1)} & w_{21}^{(1)} & w_{31}^{(1)} & \dots & \dots & \dots \\ w_{12}^{(1)} & w_{22}^{(1)} & w_{32}^{(1)} & \dots & \dots & \dots \\ w_{13}^{(1)} & w_{23}^{(1)} & w_{33}^{(1)} & \dots & \dots & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}_{N_{AP} \times N_H}$$

P_r $W^{(1)}$

$$= \begin{bmatrix} i_{11}^{(1)} & i_{12}^{(1)} & i_{13}^{(1)} & \dots & \dots & \dots \\ i_{21}^{(1)} & i_{22}^{(1)} & i_{23}^{(1)} & \dots & \dots & \dots \\ i_{31}^{(1)} & i_{32}^{(1)} & i_{33}^{(1)} & \dots & \dots & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}_{N_{MUL} \times N_H}$$

$I^{(1)}$

In the above operation, note that each element i(1) mn in the matrix I(1) denotes the dot product obtained at the nth node in the first hidden layer.

The step 2 in the forwarding operation i.e bias addition between the input layer and the hidden layer is shown below.

$$\begin{bmatrix} i_{11}^{(1)} & i_{12}^{(1)} & i_{13}^{(1)} & \dots & \dots & \dots \\ i_{21}^{(1)} & i_{22}^{(1)} & i_{23}^{(1)} & \dots & \dots & \dots \\ i_{31}^{(1)} & i_{32}^{(1)} & i_{33}^{(1)} & \dots & \dots & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}_{N_{MUL} \times N_H} + \begin{bmatrix} bw_1^{(1)} & bw_2^{(1)} & bw_3^{(1)} & \dots & \dots & \dots \\ bw_1^{(1)} & bw_2^{(1)} & bw_3^{(1)} & \dots & \dots & \dots \\ bw_1^{(1)} & bw_2^{(1)} & bw_3^{(1)} & \dots & \dots & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}_{N_{MUL} \times N_H}$$

$I^{(1)}$ $BW^{(1)}$

$$= \begin{bmatrix} i_{11}^{(1)} + bw_1^{(1)} & i_{12}^{(1)} + bw_2^{(1)} & i_{13}^{(1)} + bw_3^{(1)} & \dots & \dots & \dots \\ i_{21}^{(1)} + bw_1^{(1)} & i_{22}^{(1)} + bw_2^{(1)} & i_{23}^{(1)} + bw_3^{(1)} & \dots & \dots & \dots \\ i_{31}^{(1)} + bw_1^{(1)} & i_{32}^{(1)} + bw_2^{(1)} & i_{33}^{(1)} + bw_3^{(1)} & \dots & \dots & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}_{N_{MUL} \times N_H}$$

$I^{(1)}$

$$\begin{bmatrix} p_{r11} & p_{r12} & p_{r13} & \dots & \dots & \dots \\ p_{r21} & p_{r22} & p_{r23} & \dots & \dots & \dots \\ p_{r31} & p_{r32} & p_{33} & \dots & \dots & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}_{N_{MUL} \times N_{AP}} \begin{bmatrix} w_{11}^{(1)} & w_{21}^{(1)} & w_{31}^{(1)} & \dots & \dots & \dots \\ w_{12}^{(1)} & w_{22}^{(1)} & w_{32}^{(1)} & \dots & \dots & \dots \\ w_{13}^{(1)} & w_{23}^{(1)} & w_{33}^{(1)} & \dots & \dots & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}_{N_{AP} \times N_{H1}}$$

$$P_r \qquad\qquad W^{(1)}$$

$$= \begin{bmatrix} i_{11}^{(1)} & i_{12}^{(1)} & i_{13}^{(1)} & \dots & \dots & \dots \\ i_{21}^{(1)} & i_{22}^{(1)} & i_{23}^{(1)} & \dots & \dots & \dots \\ i_{31}^{(1)} & i_{32}^{(1)} & i_{33}^{(1)} & \dots & \dots & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}_{N_{MUL} \times N_{H1}}$$

$$I^{(1)}$$

$$\begin{bmatrix} i_{11}^{(1)} & i_{12}^{(1)} & i_{13}^{(1)} & \dots & \dots & \dots \\ i_{21}^{(1)} & i_{22}^{(1)} & i_{23}^{(1)} & \dots & \dots & \dots \\ i_{31}^{(1)} & i_{32}^{(1)} & i_{33}^{(1)} & \dots & \dots & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}_{N_{MUL} \times N_{H1}} + \begin{bmatrix} bw_1^{(1)} & bw_2^{(1)} & bw_3^{(1)} & \dots & \dots & \dots \\ bw_1^{(1)} & bw_2^{(1)} & bw_3^{(1)} & \dots & \dots & \dots \\ bw_1^{(1)} & bw_2^{(1)} & bw_3^{(1)} & \dots & \dots & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}_{N_{MUL} \times N_{H1}}$$

$$I^{(1)} \qquad\qquad BW_1^{(1)}$$

$$= \begin{bmatrix} i_{11}^{(1)} + bw_1^{(1)} & i_{12}^{(1)} + bw_2^{(1)} & i_{13}^{(1)} + bw_3^{(1)} & \dots & \dots & \dots \\ i_{21}^{(1)} + bw_1^{(1)} & i_{22}^{(1)} + bw_2^{(1)} & i_{23}^{(1)} + bw_3^{(1)} & \dots & \dots & \dots \\ i_{31}^{(1)} + bw_1^{(1)} & i_{32}^{(1)} + bw_2^{(1)} & i_{33}^{(1)} + bw_3^{(1)} & \dots & \dots & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}_{N_{MUL} \times N_{H1}}$$

$$I^{(1)}$$

$$S^{(1)} = \begin{bmatrix} f(i_{11}^{(1)} + bw_1^{(1)}) & f(i_{12}^{(1)} + bw_2^{(1)}) & f(i_{13}^{(1)} + bw_3^{(1)}) & \dots & \dots & \dots \\ f(i_{21}^{(1)} + bw_1^{(1)}) & f(i_{22}^{(1)} + bw_2^{(1)}) & f(i_{23}^{(1)} + bw_3^{(1)}) & \dots & \dots & \dots \\ f(i_{31}^{(1)} + bw_1^{(1)}) & f(i_{32}^{(1)} + bw_2^{(1)}) & f(i_{33}^{(1)} + bw_3^{(1)}) & \dots & \dots & \dots \\ \vdots & & \vdots & & \vdots & \vdots & \vdots & \vdots \end{bmatrix}_{N_{MUL} \times N_{H1}}$$

$$= \begin{bmatrix} s_{11}^{(1)} & s_{12}^{(1)} & s_{13}^{(1)} & \dots & \dots & \dots \\ s_{21}^{(1)} & s_{22}^{(1)} & s_{23}^{(1)} & \dots & \dots & \dots \\ s_{31}^{(1)} & s_{32}^{(1)} & s_{33}^{(1)} & \dots & \dots & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}_{N_{MUL} \times N_{H1}}$$

$$\begin{bmatrix} s_{11}^{(1)} & s_{12}^{(1)} & s_{13}^{(1)} & \dots & \dots & \dots \\ s_{21}^{(1)} & s_{22}^{(1)} & s_{23}^{(1)} & \dots & \dots & \dots \\ s_{31}^{(1)} & s_{32}^{(1)} & s_{33}^{(1)} & \dots & \dots & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}_{N_{MUL} \times N_{H1}} \begin{bmatrix} w_{11}^{(2)} & w_{21}^{(2)} & w_{31}^{(2)} & \dots & \dots & \dots \\ w_{12}^{(2)} & w_{22}^{(2)} & w_{32}^{(2)} & \dots & \dots & \dots \\ w_{13}^{(2)} & w_{23}^{(2)} & w(2)_{33} & \dots & \dots & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}_{N_{H1} \times N_{H2}}$$

$S^{(1)}$ $W^{(2)}$

$$= \begin{bmatrix} i_{11}^{(2)} & i_{12}^{(2)} & i_{13}^{(2)} & \dots & \dots & \dots \\ i_{21}^{(2)} & i_{22}^{(2)} & i_{23}^{(2)} & \dots & \dots & \dots \\ i_{31}^{(2)} & i_{32}^{(2)} & i_{33}^{(2)} & \dots & \dots & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}_{N_{MUL} \times N_{H2}}$$

$I^{(2)}$

$$\begin{bmatrix} i_{11}^{(2)} & i_{12}^{(2)} & i_{13}^{(2)} & \dots & \dots & \dots \\ i_{21}^{(2)} & i_{22}^{(2)} & i_{23}^{(2)} & \dots & \dots & \dots \\ i_{31}^{(2)} & i_{32}^{(2)} & i_{33}^{(2)} & \dots & \dots & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}_{N_{MUL} \times N_{H2}} + \begin{bmatrix} bw_1^{(2)} & bw_2^{(2)} & bw_3^{(2)} & \dots & \dots & \dots \\ bw_1^{(2)} & bw_2^{(2)} & bw_3^{(2)} & \dots & \dots & \dots \\ bw_1^{(2)} & bw_2^{(2)} & bw_3^{(2)} & \dots & \dots & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}_{N_{MUL} \times N_{H2}}$$

$$I^{(2)} \qquad\qquad BW_1^{(2)}$$

$$= \begin{bmatrix} i_{11}^{(2)} + bw_1^{(2)} & i_{12}^{(2)} + bw_2^{(2)} & i_{13}^{(2)} + bw_3^{(2)} & \dots & \dots & \dots \\ i_{21}^{(2)} + bw_1^{(2)} & i_{22}^{(2)} + bw_2^{(2)} & i_{23}^{(2)} + bw_3^{(2)} & \dots & \dots & \dots \\ i_{31}^{(2)} + bw_1^{(2)} & i_{32}^{(2)} + bw_2^{(2)} & i_{33}^{(2)} + bw_3^{(2)} & \dots & \dots & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}_{N_{MUL} \times N_{H2}}$$

$$I^{(2)}$$

$$S^{(2)} = \begin{bmatrix} f(i_{11}^{(2)} + bw_1^{(2)}) & f(i_{12}^{(2)} + bw_2^{(2)}) & f(i_{13}^{(2)} + bw_3^{(2)}) & \dots & \dots & \dots \\ f(i_{21}^{(2)} + bw_1^{(2)}) & f(i_{22}^{(2)} + bw_2^{(2)}) & f(i_{23}^{(2)} + bw_3^{(2)}) & \dots & \dots & \dots \\ f(i_{31}^{(2)} + bw_1^{(2)}) & f(i_{32+bw_2^{(2)}}^{(2)}) & f(i_{33}^{(2)} + bw_3^{(2)}) & \dots & \dots & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}_{N_{MUL} \times N_{H2}}$$

$$= \begin{bmatrix} s_{11}^{(2)} & s_{12}^{(2)} & s_{13}^{(2)} & \dots & \dots & \dots \\ s_{21}^{(2)} & s_{22}^{(2)} & s_{23}^{(2)} & \dots & \dots & \dots \\ s_{31}^{(2)} & s_{32}^{(2)} & s_{33}^{(2)} & \dots & \dots & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}_{N_{MUL} \times N_{H2}}$$

$$\begin{bmatrix} s_{11}^{(2)} & s_{12}^{(2)} & s_{13}^{(2)} & \dots & \dots & \dots \\ s_{21}^{(2)} & s_{22}^{(2)} & s_{23}^{(2)} & \dots & \dots & \dots \\ s_{31}^{(2)} & s_{32}^{(2)} & s_{33}^{(2)} & \dots & \dots & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}_{N_{MUL} \times N_{H2}} \begin{bmatrix} w_{11}^{(3)} & w_{21}^{(3)} & w_{31}^{(3)} \\ w_{12}^{(3)} & w_{22}^{(3)} & w_{32}^{(3)} \\ w_{13}^{(3)} & w_{23}^{(3)} & w(2)_{33} \\ \vdots & \vdots & \vdots \end{bmatrix}_{N_{H2} \times 3} = \begin{bmatrix} i_{11}^{(3)} & i_{12}^{(3)} & i_{13}^{(3)} \\ i_{21}^{(3)} & i_{22}^{(3)} & i_{23}^{(3)} \\ i_{31}^{(3)} & i_{32}^{(3)} & i_{33}^{(3)} \\ \vdots & \vdots & \vdots \end{bmatrix}_{N_{MUL} \times 3}$$

$S^{(2)}$ $W^{(3)}$ $I^{(3)}$

$$\begin{bmatrix} i_{11}^{(3)} & i_{12}^{(3)} & i_{13}^{(3)} \\ i_{21}^{(3)} & i_{22}^{(3)} & i_{23}^{(3)} \\ i_{31}^{(3)} & i_{32}^{(3)} & i_{33}^{(3)} \\ \vdots & \vdots & \vdots \end{bmatrix}_{N_{MUL} \times 3} + \begin{bmatrix} bw_1^{(3)} & bw_2^{(3)} & bw_3^{(3)} \\ bw_1^{(3)} & bw_2^{(3)} & bw_3^{(3)} \\ bw_1^{(3)} & bw_2^{(3)} & bw_3^{(3)} \\ \vdots & \vdots & \vdots \end{bmatrix}_{N_{MUL} \times 3}$$

$I^{(3)}$ $BW^{(3)}$

$$= \begin{bmatrix} i_{11}^{(3)} + bw_1^{(3)} & i_{12}^{(3)} + bw_2^{(3)} & i_{13}^{(3)} + bw_3^{(3)} & \dots & \dots & \dots \\ i_{21}^{(3)} + bw_1^{(3)} & i_{22}^{(3)} + bw_2^{(3)} & i_{23}^{(3)} + bw_3^{(3)} & \dots & \dots & \dots \\ i_{31}^{(3)} + bw_1^{(3)} & i_{32}^{(3)} + bw_2^{(3)} & i_{33}^{(3)} + bw_3^{(3)} & \dots & \dots & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}_{N_{MUL} \times N_3}$$

$I^{(3)}$

$$\hat{C} = \begin{bmatrix} f(i_{11}^{(3)}) & f(i_{12}^{(3)}) & f(i_{13}^{(3)}) \\ f(i_{21}^{(3)}) & f(i_{22}^{(3)}) & f(i_{23}^{(3)}) \\ f(i_{31}^{(3)}) & f(i_{32}^{(3)}) & f(i_{33}^{(3)}) \\ \vdots & \vdots & \vdots \end{bmatrix}_{N_{MUL} \times 3} = \begin{bmatrix} \hat{c}_{11} & \hat{c}_{12} & \hat{c}_{13} \\ \hat{c}_{21} & \hat{c}_{22} & \hat{c}_{23} \\ \hat{c}_{31} & \hat{c}_{32} & \hat{c}_{33} \\ \vdots & \vdots & \vdots \end{bmatrix}_{N_{MUL} \times 3}$$

The two important points that can be inferred from this section are:

- The processing happens with input power vectors sequentially, i.e., after the first power vector is processed, the second power vector is processed, after the second power vector is processed, the third power vector is processed, and so on.
- The dimensions of the weight matrices do not depend on the number of MU locations inside the room.

3.5 Optimization algorithm

From the forwarding operation discussed above, it is to be inferred that the error cost function depends on both the weight matrices and the weighted bias matrices.

The main objective of the AFNN is to minimize the cost function with respect to these matrices. When the cost function is plotted against the weight matrices and the weighted bias matrices, different 3-D surfaces are obtained. The minimization of the error cost function is done by descending to a point on each of these 3-D surfaces at which the gradients (slopes) with respect to weight matrices and weighted bias matrices become approximately zero. The descent of the point is not a one-step process but rather happens through several iterations. In each iteration, we find the gradients (slopes) with respect to weight matrices and weighted bias matrices and reduce the old weights by the scaled versions of these differentials to get the new weights. With these new weights, the forwarding operation takes place again to generate an error cost function which in turn gives new gradients. This process is repeated until the gradients tend to zero. The updation of the weights and bias weights takes place through the following equations: For a single hidden layer AFNN,

$$W_{i+1}^{(1)} = W_i^{(1)} - step * \frac{dJ}{dW_i^{(1)}}$$

$$W_{i+1}^{(2)} = W_i^{(2)} - step * \frac{dJ}{dW_i^{(2)}}$$

$$BW_{i+1}^{(2)} = BW_i^{(2)} - step * \frac{dJ}{dBW_i^{(2)}}$$

$$BW_{i+1}^{(1)} = BW_i^{(1)} - step * \frac{dJ}{dBW_i^{(1)}}$$

where $W_{i+1}^{(1)}$, $W_{i+1}^{(2)}$ are the new weight matrices after the updation

$BW_{i+1}^{(2)}$, $BW_{i+1}^{(1)}$ are the new bias weight matrices after the updation

$dJ/dW_i^{(1)}$ and $dJ/dW_i^{(2)}$ are the weight differentials evaluated during the iteraion

$dJ/dBW_i^{(1)}$ and $dJ/dBW_i^{(2)}$ are the bas weight differentials evaluated during the iteration. The step value is appropriately selected such that the cost function descends at a decent rate.

For a two hidden layer AFNN,

$$W_{i+1}^{(1)} = W_i^{(1)} - step * \frac{dJ}{dW_i^{(1)}}$$

$$W_{i+1}^{(2)} = W_i^{(2)} - step * \frac{dJ}{dW_i^{(2)}}$$

$$W_{i+1}^{(3)} = W_i^{(3)} - step * \frac{dJ}{dW_i^{(3)}}$$

$$BW_{i+1}^{(3)} = BW_i^{(3)} - step * \frac{dJ}{dBW_i^{(3)}}$$

$$BW_{i+1}^{(2)} = BW_i^{(2)} - step * \frac{dJ}{dBW_i^{(2)}}$$

$$BW_{i+1}^{(1)} = BW_i^{(1)} - step * \frac{dJ}{dBW_i^{(1)}}$$

3.6 Backpropagation

The most important thing in the entire implementation of the FNN is the evaluation of the gradients. For a single hidden layer FNN, the evaluation of gradients is explained through the following equations:

$$\begin{aligned}\frac{dJ}{dW^{(2)}} &= -k(C-\hat{C})\frac{d\hat{C}}{dW^{(2)}} = -k(C-\hat{C})\frac{df(I^{(2)}+BW^{(2)})}{dW^{(2)}} \\ &= -k(C-\hat{C})\frac{df(I^{(2)}+BW^{(2)})}{dI^{(2)}}\frac{d(I^{(2)})}{dW^{(2)}} = -k(C-\hat{C})\frac{df(I^{(2)}+BW^{(2)})}{dI^{(2)}}\frac{d(S^{(1)}W^{(2)})}{dW^{(2)}} \\ &= (S^{(1)})^T - k(C-\hat{C})\frac{df(I^{(2)}+BW^{(2)})}{dI^{(2)}} = (S^{(1)})^T - k(C-\hat{C})f'(I^{(2)}+BW^{(2)})\end{aligned}$$

$$\frac{dJ}{dBW^{(2)}} = -k(C-\hat{C})\frac{d\hat{C}}{dBW^{(2)}} = -k(C-\hat{C})\frac{df(I^{(2)}+BW^{(2)})}{dBW^{(2)}} = -k(C-\hat{C})f'(I^{(2)}+BW^{(2)})$$

$$\begin{aligned}\frac{dJ}{dW^{(1)}} &= -k(C-\hat{C})\frac{d\hat{C}}{dW^{(1)}} = -k(C-\hat{C})\frac{df(I^{(2)}+BW^{(2)})}{dW^{(1)}} \\ &= -k(C-\hat{C})\frac{df(I^{(2)}+BW^{(2)})}{dI^{(2)}}\frac{d(I^{(2)})}{dW^{(1)}} \\ &= -k(C-\hat{C})\frac{df(I^{(2)}+BW^{(2)})}{dI^{(2)}}\frac{d(S^{(1)}.W^{(2)})}{dW^{(1)}} \\ &= -k(C-\hat{C})fracdf(I^{(2)}+BW^{(2)})dI^{(2)}(W^{(2)})^T\frac{d(S^{(1)})}{dW^{(1)}} \\ &= -k(C-\hat{C})fracdf(I^{(2)}+BW^{(2)})dI^{(2)}(W^{(2)})^T\frac{df(I^{(1)}+BW^{(1)})}{dW^{(1)}} \\ &= -k(C-\hat{C})\frac{df(I^{(2)}+BW^{(2)})}{dI^{(2)}}(W^{(2)})^T\frac{df(I^{(1)}+BW^{(1)})}{dI^{(1)}}\frac{dI^{(1)}}{dW^{(1)}} \\ &= -k(C-\hat{C})\frac{df(I^{(2)})}{dI^{(2)}}(W^{(2)})^T\frac{df(I^{(1)}+BW^{(1)})}{dI^{(1)}}\frac{d(P_r.W^{(1)})}{dW^{(1)}} \\ &= (P_r)^T. - k(C-\hat{C})\frac{df(I^{(2)}+BW^{(2)})}{dI^{(2)}}(W^{(2)})^T\frac{df(I^{(1)}+BW^{(1)})}{dI^{(1)}} \\ &= (P_r)^T. - k(C-\hat{C})f'(I^{(2)}+BW^{(2)})(W^{(2)})^T.f'(I^{(1)}+BW^{(1)})\end{aligned}$$

$$\frac{dJ}{dBW^{(1)}} = -k(C-\hat{C})\frac{d\hat{C}}{dBW^{(1)}} = -k(C-\hat{C})\frac{df(I^{(2)}+BW^{(2)})}{dBW^{(1)}}$$
$$= -k(C-\hat{C})\frac{df(I^{(2)}+BW^{(2)})}{dI^{(2)}}\frac{d(I^{(2)})}{dBW^{(1)}} = -k(C-\hat{C})\frac{df(I^{(2)}+BW^{(2)})}{dI^{(2)}}\frac{d(S^{(1)}.W^{(2)})}{dBW^{(1)}}$$
$$= -k(C-\hat{C})fracdf(I^{(2)}+BW^{(2)})dI^{(2)}(W^{(2)})^T\frac{d(S^{(1)})}{dBW^{(1)}}$$
$$= -k(C-\hat{C})fracdf(I^{(2)}+BW^{(2)})dI^{(2)}(W^{(2)})^T\frac{df(I^{(1)}+BW^{(1)})}{dBW^{(1)}}$$
$$= -k(C-\hat{C})f'(I^{(2)}+BW^{(2)})(W^{(2)})^T f'(I^{(1)}+BW^{(1)})$$

This error term is involved even in the evaluation of the gradient which can be thought of as error propagation from the last layer to the first layer. As the error seems propagated from back to the first layer, the method of evaluation is termed as the backpropagation algorithm. The evaluation of gradients for an AFNN with two hidden layers is explained through the following equations:

$$\frac{dJ}{dW^{(3)}} = -k(C-\hat{C})\frac{d\hat{C}}{dW^{(3)}} = -k(C-\hat{C})\frac{df(I^{(3)}+BW^{(3)})}{dW^{(3)}}$$
$$= -k(C-\hat{C})\frac{df(I^{(3)}+BW^{(3)})}{dI^{(3)}}\frac{d(I^{(3)})}{dW^{(3)}}$$
$$= -k(C-\hat{C})\frac{df(I^{(3)}+BW^{(3)})}{dI^{(3)}}\frac{d(S^{(2)}.W^{(3)})}{dW^{(3)}} = (S^{(2)})^T - k(C-\hat{C})\frac{df(I^{(3)}+BW^{(3)})}{dI^{(3)}}$$
$$= (S^{(2)})^T.-k(C-\hat{C})f'(I^{(3)}+BW^{(3)})$$

$$\frac{dJ}{dBW^{(3)}} = -k(C-\hat{C})\frac{d\hat{C}}{dBW^{(3)}} = -k(C-\hat{C})\frac{df(I^{(3)}+BW^{(3)})}{dBW^{(3)}}$$
$$= -k(C-\hat{C})f'(I^{(3)}+BW^{(3)})$$

$$
\begin{aligned}
\frac{dJ}{dW^{(2)}} &= -k(C-\hat{C})\frac{d\hat{C}}{dW^{(2)}} = -k(C-\hat{C})\frac{df(I^{(3)}+BW^{(3)})}{dW^{(2)}} \\
&= -k(C-\hat{C})\frac{df(I^{(3)}+BW^{(3)})}{dI^{(3)}}\frac{d(I^{(3)})}{dW^{(2)}} \\
&= -k(C-\hat{C})\frac{df(I^{(3)}+BW^{(3)})}{dI^{(3)}}\frac{d(S^{(2)}.W^{(3)})}{dW^{(2)}} \\
&= -k(C-\hat{C})\frac{df(I^{(3)}+BW^{(3)})}{dI^{(3)}}(W^{(3)})^T\frac{d(S^{(2)})}{dW^{(2)}} \\
&= -k(C-\hat{C})\frac{df(I^{(3)}+BW^{(3)})}{dI^{(3)}}(W^{(3)})^T\frac{df(I^{(2)}+BW^{(2)})}{dW^{(2)}} \\
&= -k(C-\hat{C})\frac{df(I^{(3)}+BW^{(3)})}{dI^{(3)}}(W^{(3)})^T\frac{df(I^{(2)}+BW^{(2)})}{dI^{(2)}}\frac{dI^{(2)}}{dW^{(2)}} \\
&= -k(C-\hat{C})\frac{df(I^{(3)}+BW^{(3)})}{dI^{(3)}}(W^{(3)})^T\frac{df(I^{(2)}+BW^{(2)})}{dI^{(2)}}\frac{d(S^{(1)}.W^{(2)})}{dW^{(2)}} \\
&= (S^{(1)})^T.k - (C-\hat{C})\frac{df(I^{(3)}+BW^{(3)})}{dI^{(3)}}(W^{(3)})^T\frac{df(I^{(2)}+BW^{(2)})}{dI^{(2)}} \\
&= (S^{(1)})^T.k - (C-\hat{C})f(I^{(3)}+BW^{(3)})(W^{(3)})^T f'(I^{(2)}+BW^{(2)})
\end{aligned}
$$

$$
\begin{aligned}
\frac{dJ}{dBW^{(2)}} &= -k(C-\hat{C})\frac{d\hat{C}}{dBW^{(2)}} = -k(C-\hat{C})\frac{df(I^{(3)}+BW^{(3)})}{dBW^{(2)}} \\
&= -k(C-\hat{C})\frac{df(I^{(3)}+BW^{(3)})}{dI^{(3)}}\frac{d(I^{(3)})}{dBW^{(2)}} \\
&= -k(C-\hat{C})\frac{df(I^{(3)}+BW^{(3)})}{dI^{(3)}}\frac{d(S^{(2)}.W^{(3)})}{dBW^{(2)}} \\
&= -k(C-\hat{C})\frac{df(I^{(3)}+BW^{(3)})}{dI^{(3)}}(W^{(3)})^T\frac{d(S^{(2)})}{dBW^{(2)}} \\
&= -k(C-\hat{C})\frac{df(I^{(3)}+BW^{(3)})}{dI^{(3)}}(W^{(3)})^T\frac{df(I^{(2)}+BW^{(2)})}{dBW^{(2)}} \\
&= -k(C-\hat{C})f(I^{(3)}+BW^{(3)})(W^{(3)})^T f'(I^{(2)}+BW^{(2)})
\end{aligned}
$$

$$\frac{dJ}{dW^{(1)}} = -k(C-\hat{C})\frac{d\hat{C}}{dW^{(1)}} = -k(C-\hat{C})\frac{df(I^{(3)}+BW^{(3)})}{dW^{(1)}}$$
$$= -k(C-\hat{C})\frac{df(I^{(3)}+BW^{(3)})}{dI^{(3)}}\frac{d(I^{(3)})}{dW^{(1)}}$$
$$= -k(C-\hat{C})\frac{df(I^{(3)}+BW^{(3)})}{dI^{(3)}}\frac{d(S^{(2)}.W^{(3)})}{dW^{(1)}}$$
$$= -k(C-\hat{C})\frac{df(I^{(3)}+BW^{(3)})}{dI^{(3)}}(W^{(3)})^T\frac{d(S^{(2)})}{dW^{(1)}}$$
$$= -k(C-\hat{C})\frac{df(I^{(3)}+BW^{(3)})}{dI^{(3)}}(W^{(3)})^T\frac{df(I^{(2)}+BW^{(2)})}{dW^{(1)}}$$
$$= -k(C-\hat{C})\frac{df(I^{(3)}+BW^{(3)})}{dI^{(3)}}(W^{(3)})^T\frac{df(I^{(2)}+BW^{(2)})}{dI^{(2)}}\frac{dI^{(2)}}{dW^{(1)}}$$
$$= -k(C-\hat{C})\frac{df(I^{(3)}+BW^{(3)})}{dI^{(3)}}(W^{(3)})^T\frac{df(I^{(2)}+BW^{(2)})}{dI^{(2)}}\frac{d(S^{(1)}.W^{(2)})}{dW^{(1)}}$$
$$= -k(C-\hat{C})\frac{df(I^{(3)}+BW^{(3)})}{dI^{(3)}}(W^{(3)})^T\frac{df(I^{(2)}+BW^{(2)})}{dI^{(2)}}(W^{(2)})^T\frac{dS^{(1)}}{dW^{(1)}}$$
$$= -k(C-\hat{C})\frac{df(I^{(3)}+BW^{(3)})}{dI^{(3)}}(W^{(3)})^T\frac{df(I^{(2)}+BW^{(2)})}{dI^{(2)}}(W^{(2)})^T\frac{df(I^{(1)}+BW^{(1)})}{dW^{(1)}}$$
$$= -k(C-\hat{C})\frac{df(I^{(3)}+BW^{(3)})}{dI^{(3)}}(W^{(3)})^T\frac{df(I^{(2)}+BW^{(2)})}{dI^{(2)}}(W^{(2)})^T\frac{df(I^{(1)}+BW^{(1)})}{dI^{(1)}}\frac{dI^{(1)}}{dW^{(1)}}$$
$$= -k(C-\hat{C})\frac{df(I^{(3)}+BW^{(3)})}{dI^{(3)}}(W^{(3)})^T\frac{df(I^{(2)}+BW^{(2)})}{dI^{(2)}}(W^{(2)})^T$$
$$\frac{df(I^{(1)}+BW^{(1)})}{dI^{(1)}}\frac{d(P_r.W^{(1)})}{dW^{(1)}}$$
$$= (P_r)^T. - k(C-\hat{C})\frac{df(I^{(3)}+BW^{(3)})}{dI^{(3)}}(W^{(3)})^T\frac{df(I^{(2)}+BW^{(2)})}{dI^{(2)}}(W^{(2)})^T\frac{df(I^{(1)}+BW^{(1)})}{dI^{(1)}}$$
$$= (P_r)^T. - k(C-\hat{C})f'(I^{(3)}+BW^{(3)})(W^{(3)})^T f'(I^{(2)}+BW^{(2)})(W^{(2)})^T f'(I^{(1)}+BW^{(1)})$$

The three most important and fundamental things to remember while evaluating the gradients are:

- The differentiation of a scalar with respect to a matrix is a matrix with the same order as that of the first matrix.
- The summation is not necessary during the evaluation of the gradient as ultimately the matrix multiplication takes care of summing the appropriate terms.
- If A = f(C) and B are two matrices, then the product of A and B differentiated with respect to C always results in the transpose of B.

CHAPTER 4
METHODOLOGY

Mobile User location for a given input power vector

4.1 Training the Feedforward Neural Network

Normalization is the process of modifying the range of a dataset (generally to 0 and 1 or to -1 and +1) by appropriately scaling the elements in the dataset. There are different techniques through which normalization can be achieved. For the current implementation of Indoor Positioning System (IPS), at the input side, the min-max normalization technique is used, whereas at the output side, simple scaling technique is used.

$$output = \frac{input - min(thewholedataset)}{max(thewholedataset) - min(thewholedataset)}$$

Similarly, the scaling operation is performed with the following equation.

$$output = \frac{input}{maximumlengthintheparticulardirection}$$

The outputs of both the above techniques range from 0 to 1. It is important to note that these normalization techniques are performed over the columns of the data sets. So at the input side, the normalization is performed over the input powers received at different locations from a single router, and at the output side, the normalization is performed over the output coordinates at different locations in a single direction.

Normalization is required when there is a huge difference in the ranges of different features. For example, in the case where there are three adjacent rooms, the x, y, and z coordinates differ in their maximum values. The difference might be very significant if the rooms are big and unsymmetrical. So we cast all the three in one unified range, i.e., 0 to 1.

It becomes even more important when the activation function is Sigmoid. At very high values and at very low values, the function saturates, i.e., its gradient becomes close to zero. The learning of the network and updating of the weights during backpropagation solely depend on the gradients of output with respect to the weights. A very small gradient results in an insignificant update in weights, making the whole learning process slow.

The step size for the training process is chosen appropriately by trial and error based on the descending rate of the cost function. Generally, the step size is chosen such that at each iteration there is a reduction of the gradient scaled by 0.01.

The standard input sets mentioned in the above flowchart are matrices with a fixed number of rows and varying number of columns. The column number of the matrices depends on the number of routers placed in the room. The main purpose of these standard input sets is to aid in the normalization of the input vector. Since the first row entries of the standard input sets are removed after obtaining the output, the sizes of the standard input remain constant, thereby avoiding memory-related issues.

4.2 Testing the Feedforward Neural Network

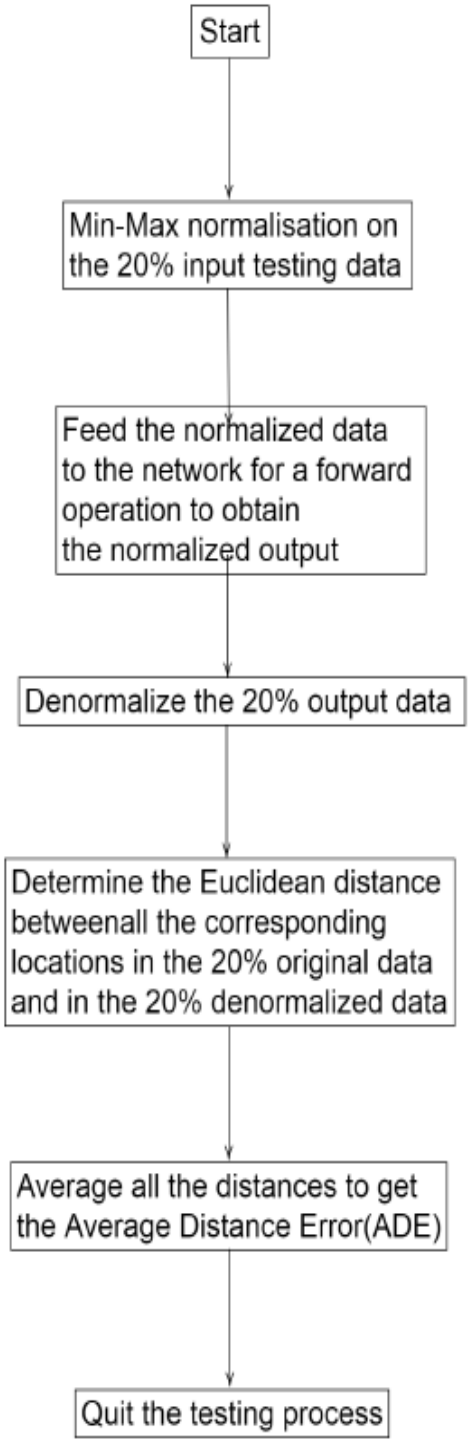

Figure – 8: Testing procedure for the Feedforward Neural Network

4.3 Determination of the output for a single input power vector

The prediction of the MU location for a single input power vector is explained through the following flow chart.

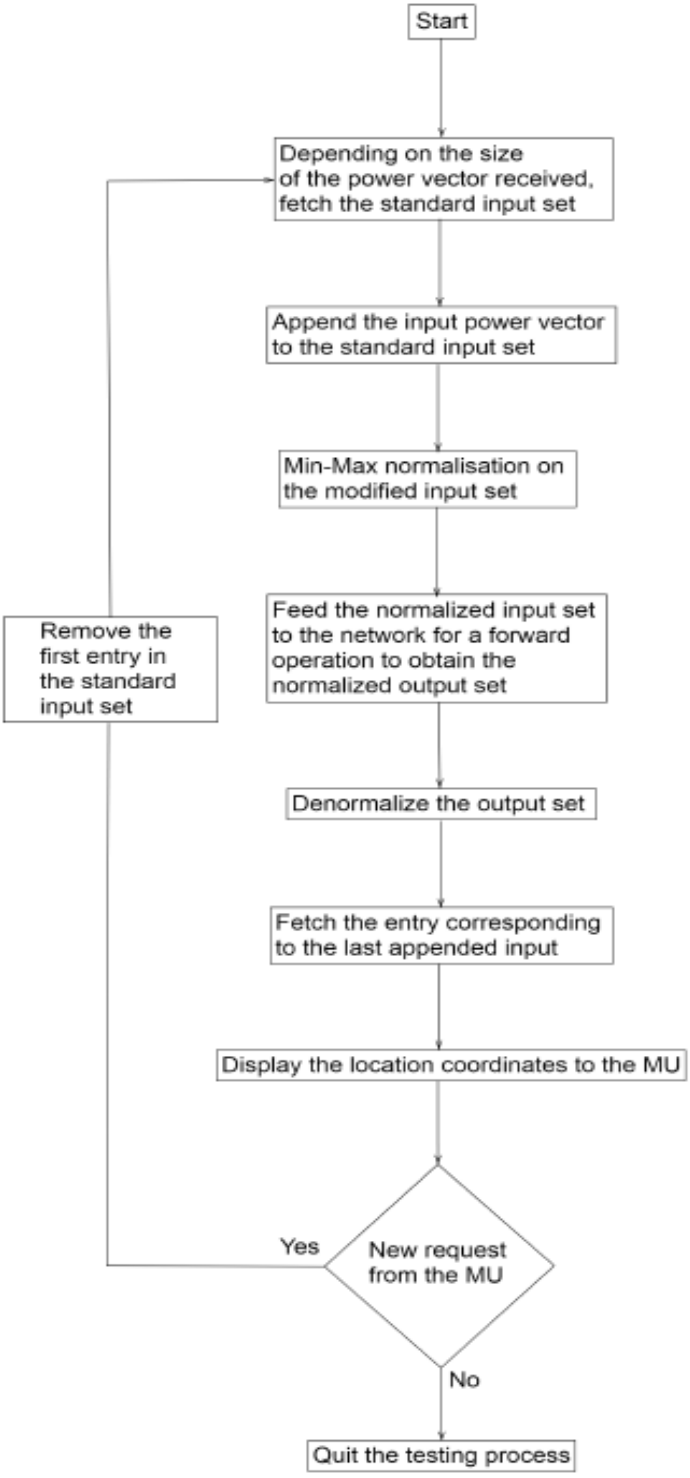

Figure – 9: Procedure for fetching the output to a single input power vector

CHAPTER 5

CODING AND OUTPUT

Data_for_Analysis.py

```
import serial
import csv

serial = serial.Serial("COM1", 115200)
with open("data_unfiltered.txt", "w") as f:
    i=1
    while i<=5000:
        for v in range(1,22):
            data = serial.read()
            #print (data)
            if v==21:
                f.write("\n")
            elif v==17:
                pass
            else:
                f.write(str(data))
        i+=1

with open("data_unfiltered.txt", "r") as f:
    with open("data_filtered.txt", "w") as f1:
        for s in range(5000):
            data1=f.readline()
            f1.write(data1[11:15]+","+data1[16:19]+"\n")

with open('data_filtered.txt', 'r') as f1:
    stripped_data = (line.strip() for line in f1)
    #lines = (line for line in stripped_data if line)
    grouped = [line.split(',') for line in stripped_data]
    with open('data_filtered.csv', 'w') as f2:
        writer = csv.writer(f2)
        writer.writerow(['Slave', 'RSSI'])
        writer.writerows(grouped)

with open('data_filtered.csv', 'r') as f2:
```

```
reader = csv.reader(f2)
v=1
x=1
y=1
z=1
fb1=open('data_filtered_b1.csv', 'w')
fb2=open('data_filtered_b2.csv', 'w')
fb3=open('data_filtered_b3.csv', 'w')
fb4=open('data_filtered_b4.csv', 'w')
for row in reader:
    for field in row:
        if field=="69DF":
            writer = csv.writer(fb4)
            if v==1:
                writer.writerow(["Slave","RSSI"])
                v+=1
            writer.writerow(row)
        elif field=="41F3":
            writer = csv.writer(fb2)
            if x==1:
                writer.writerow(["Slave","RSSI"])
                x+=1
            writer.writerow(row)
        elif field=="6CDD":
            writer = csv.writer(fb1)
            if y==1:
                writer.writerow(["Slave","RSSI"])
                y+=1
            writer.writerow(row)
        elif field=="459F":
            writer = csv.writer(fb3)
            if z==1:
                writer.writerow(["Slave","RSSI"])
                z+=1
            writer.writerow(row)
fb1.close()
fb2.close()
fb3.close()
fb4.close()
```

```
v=[]
w=[]
with open('data_filtered_b1.csv', 'r') as fb1:
    reader=csv.reader(fb1)
    for row in reader:
        if row[1]!="RSSI":
            v.append(row[1])
z=len(v)%5
y=len(v)-z
a=int(y/5)
i=0
for r in range(a):
    s=0
    for x in range(5):
        s+=int(v[i])
        i+=1
    avg=(s/5)
    w.append(str(avg))
print (w)
with open('data_filtered_b1_avg.csv', 'w') as fb1a:
    writer=csv.writer(fb1a)
    writer.writerow(["Slave","RSSI"])
    for t in w:
        writer.writerow(["6CDD",t])

q=[]
e=[]
with open('data_filtered_b2.csv', 'r') as fb2:
    reader=csv.reader(fb2)
    for row in reader:
        if row[1]!="RSSI":
            q.append(row[1])
z=len(q)%5
y=len(q)-z
a=int(y/5)
i=0
for r in range(a):
    s=0
    for x in range(5):
        s+=int(q[i])
```

```
        i+=1
    avg=(s/5)
    e.append(str(avg))
print (e)
with open('data_filtered_b2_avg.csv', 'w') as fb2a:
    writer=csv.writer(fb2a)
    writer.writerow(["Slave","RSSI"])
    for t in e:
        writer.writerow(["41F3",t])

o=[]
p=[]
with open('data_filtered_b3.csv', 'r') as fb3:
    reader=csv.reader(fb3)
    for row in reader:
        if row[1]!="RSSI":
            o.append(row[1])
z=len(o)%5
y=len(o)-z
a=int(y/5)
i=0
for r in range(a):
    s=0
    for x in range(5):
        s+=int(o[i])
        i+=1
    avg=(s/5)
    p.append(str(avg))
print (p)
with open('data_filtered_b3_avg.csv', 'w') as fb3a:
    writer=csv.writer(fb3a)
    writer.writerow(["Slave","RSSI"])
    for t in p:
        writer.writerow(["459F",t])

g=[]
h=[]
with open('data_filtered_b4.csv', 'r') as fb4:
    reader=csv.reader(fb4)
    for row in reader:
```

```
        if row[1]!="RSSI":
            g.append(row[1])
z=len(g)%5
y=len(g)-z
a=int(y/5)
i=0
for r in range(a):
    s=0
    for x in range(5):
        s+=int(g[i])
        i+=1
    avg=(s/5)
    h.append(str(avg))
print (h)
with open('data_filtered_b4_avg.csv', 'w') as fb4a:
    writer=csv.writer(fb4a)
    writer.writerow(["Slave","RSSI"])
    for t in h:
        writer.writerow(["69DF",t])
```

The graph illustrates the variation of mean ADE (Average Distance Error) for different positions of APs:

P1 - 1 AP at 1 corner (origin) of the room
P2 - 1 AP at the center of the room
P3 - 2 APs on 1 edge (X-axis) of the room
P4 - 2 APs at 2 diagonally opposite corners of the room
P5 - 2 APs at 2 opposite top corners of the room
P6 - 2 APs on the center line along the Y-axis
P7 - 3 APs at 3 corners (i.e., on X, on Y, and on Z)
P8 - 4 APs at 4 alternate corners of the room
P9 - 4 APs at 4 bottom corners of the room
P10 - 6 APs at 6 faces of the room
P11 - 6 APs along 3 axes and inside the room
P12 - 8 APs at 8 corners of the room
P13 - 8 APs in 8 quadrants of the room
P14 - 15 APs, i.e., 1 AP at the center, 6 APs at the 6 faces, and 8 APs at the 8 corners of the room
P15 - 16 APs, i.e., 8 APs on the top edges and 8 APs on the bottom edges
This graph provides insights into the performance of the indoor positioning system for different configurations of AP placements.

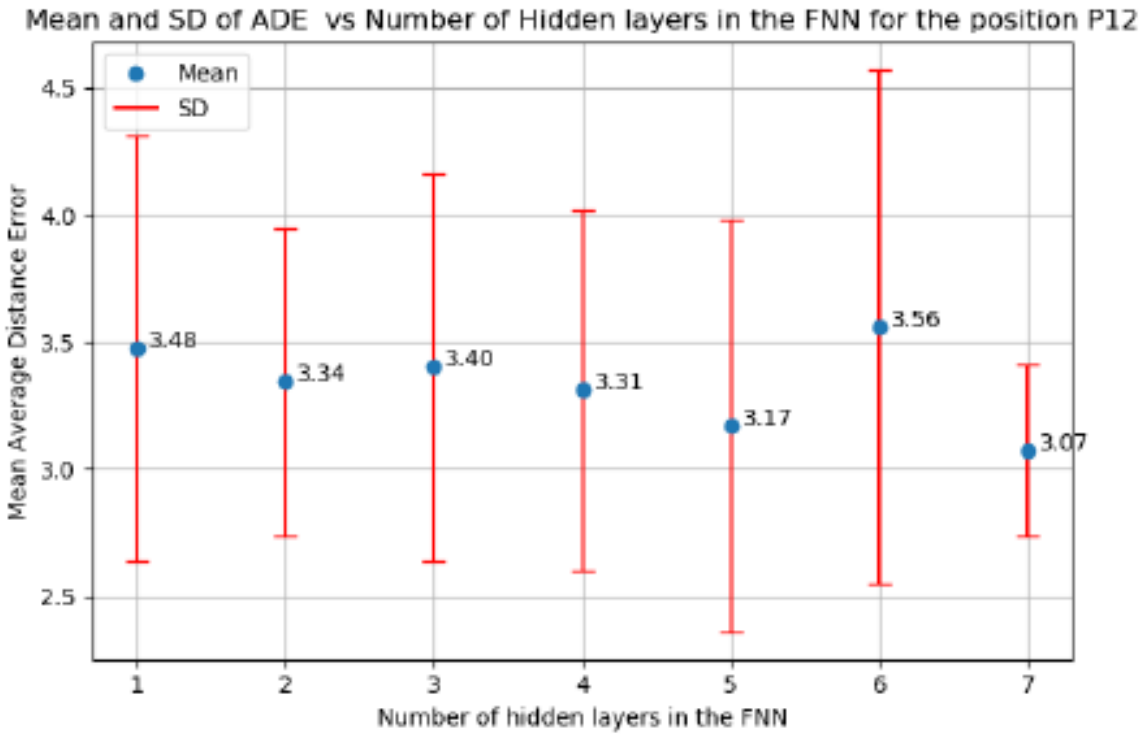

Figure - 10

The graph showing the variation of mean ADE for different number of hidden layers in the FNN

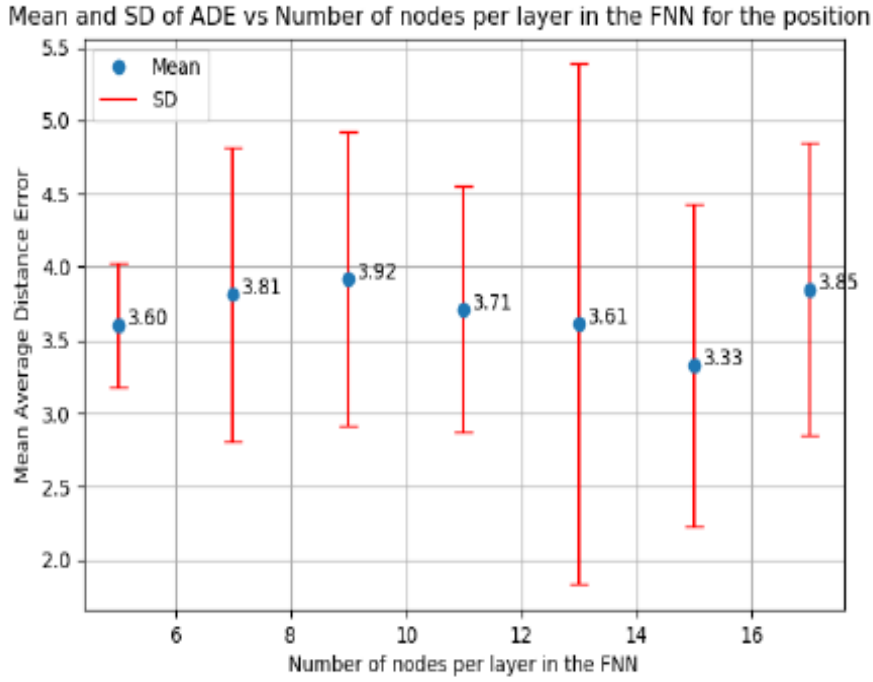

Figure - 11

The graph showing the variation of mean ADE for different number of nodes per a layer in the FNN

OUTPUTS:

Location-2

```
[[-0.27879998 -0.1675953  -0.55141319 -0.09946124]
 [ 0.17524216 -0.88826629  0.68758681 -0.13349286]
 [-0.24168931  0.51509126  0.68010211 -0.02440393]
 [-0.35448862 -0.36288186 -0.53309404 -0.19994965]
 [ 1.08020708  0.86798582  0.05926881  0.49819605]
 [-0.38047305  0.0356676  -0.34245068 -0.04088927]]
[  58.28019313    7.94205234  -48.41119511   76.74399221 -133.37250948
   38.8174669 ]
FOR 6 locations
The number of samples predicted correctly are 874
The number of samples predicted incorrectly are 128
```

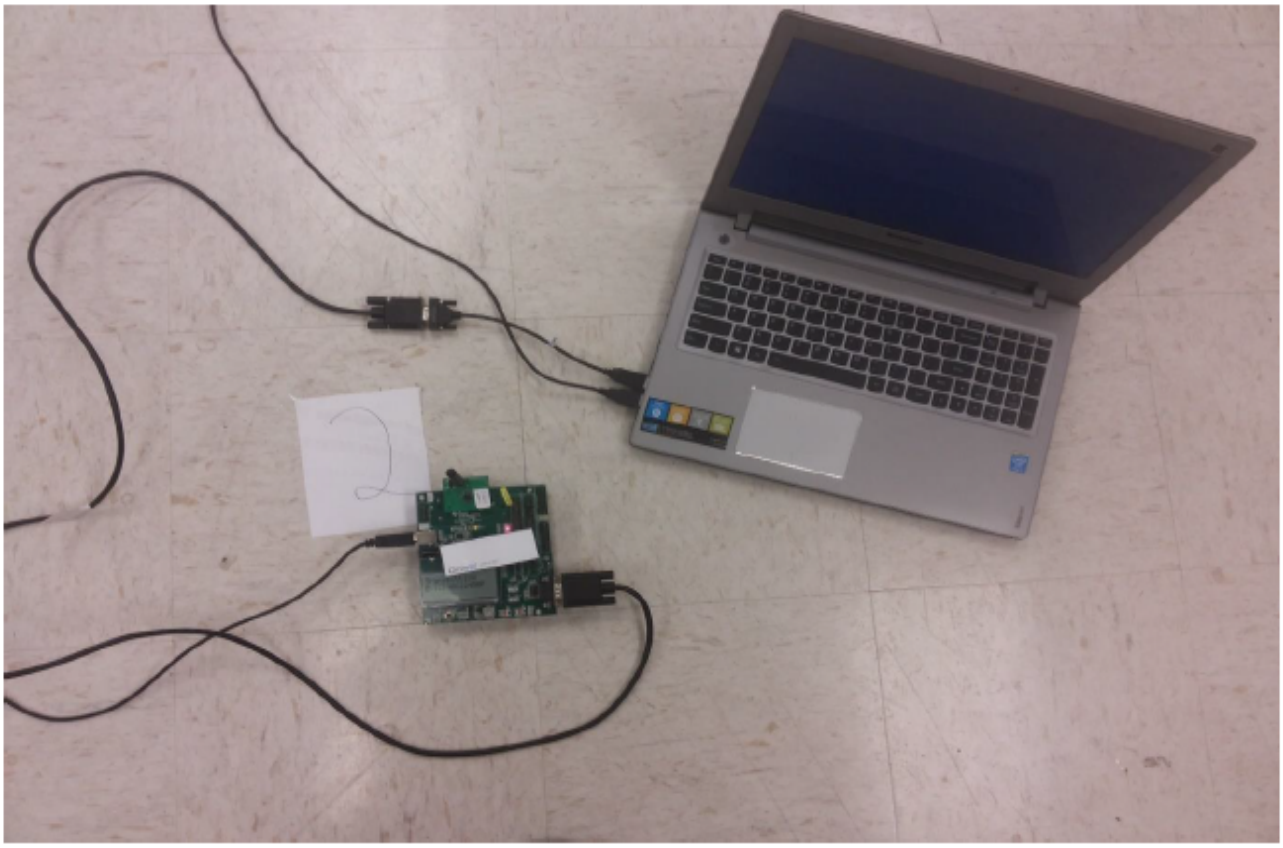

Figure – 12: Object at Location 2

Location - 3

```
Starting to regress
[[-0.27879933 -0.16759507 -0.55141231 -0.09946109]
 [ 0.17524164 -0.88826834  0.68758574 -0.13349345]
 [-0.24168986  0.5150907   0.68010097 -0.02440434]
 [-0.35448766 -0.36288139 -0.53309273 -0.1999494 ]
 [ 1.080208    0.86798616  0.05926807  0.49819673]
 [-0.38047231  0.03566791 -0.34244984 -0.04088908]]
[  58.28010596    7.94228267  -48.41104366   76.74384966 -133.37256574
   38.81737111]
Finished Regression

Enter to start0
The object is at location 2
```

When the object was placed at 3 location, the output was

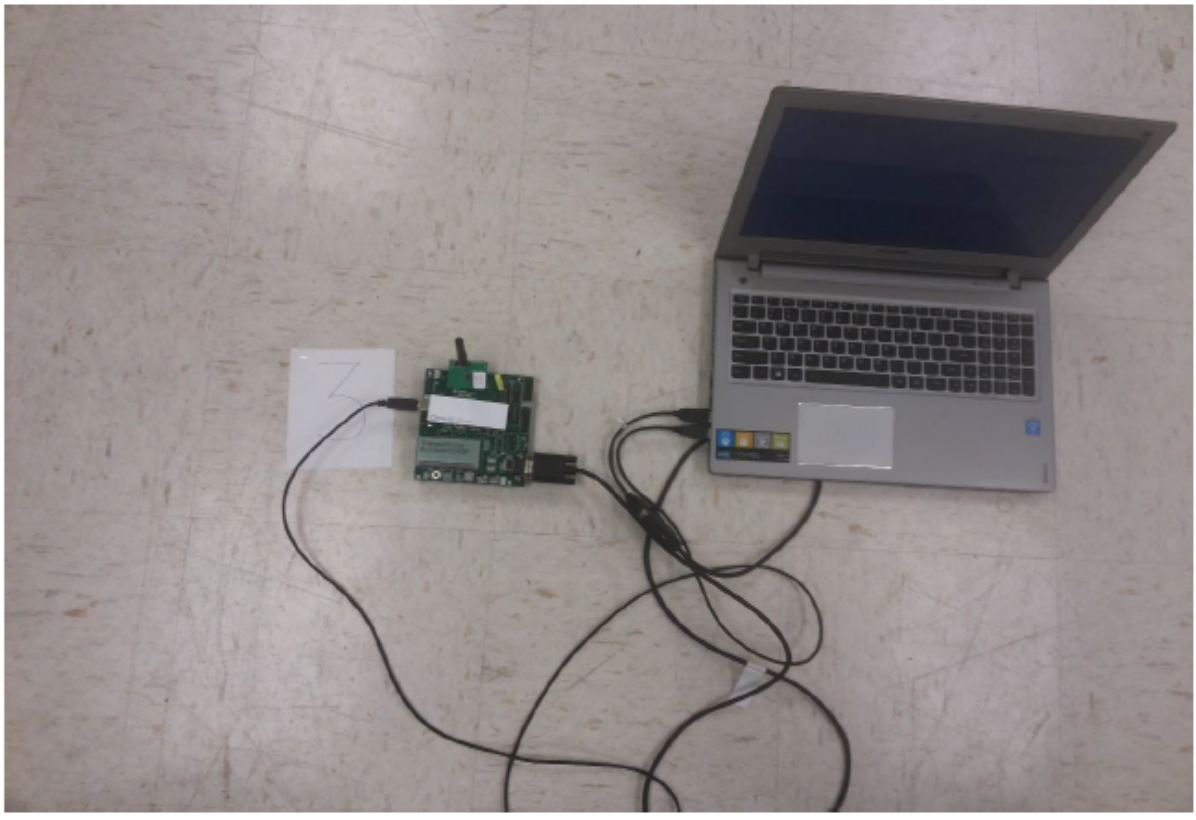

Figure – 13: Object at Location 2

CHAPTER 6

CONCLUSION

In both the single room and three adjacent room scenarios, it's observed that the key factors influencing the minimization of ADE (Average Distance Error) are the positioning of the APs, the number of neurons per layer in the FNN, and the number of hidden layers in the FNN. Surprisingly, the number of MU locations did not impact the ADE significantly in either scenario. Additionally, in the three adjacent room scenarios, the standard deviation (SD) of the shadowing noise emerged as another key factor affecting ADE.

By applying all the optimal configurations together, significant reductions in ADE are guaranteed, leading to improved prediction results. However, for future work, there is an intention to expand the implementation of Indoor Positioning (IP) to larger and real indoor environments. This expansion would involve accounting for additional factors such as fading, interference, and other environmental effects besides shadowing. Moreover, various types of neural networks, including Radial Basis Function Neural Network, Kohonen Self-Organizing Neural Network, Recurrent Neural Network (RNN), Convolutional Neural Network, and Modular Neural Network, could be explored for implementing IP with different optimization algorithms beyond backpropagation.

CHAPTER 7

REFERENCES

[Kim et al. (2012)] Kim, W., M. Park, and S.-J. Lee, Effects of shadow fading in indoor RSSI ranging. In 2012 14th International Conference on Advanced Communication Technology (ICACT). 2012. ISSN 1738-9445.

[Rojas (1996)] Rojas, R., Neural Networks - A Systematic Introduction. Springer-Verlag, Berlin, New-York, 1996.

[Zou et al. (2017)] Zou, H., Z. Chen, H. Jiang, L. Xie, and C. Spanos, Accurate indoor localization and tracking using mobile phone inertial sensors, WiFi and iBeacon. In 2017 IEEE International Symposium on Inertial Sensors and Systems (INERTIAL). 2017.

[Zou et al. (2015)] Zou, H., X. Lu, H. Jiang, and L. Xie (2015). A fast and precise indoor localization algorithm based on an online sequential extreme learning machine. 15, 1804–24.