An-Najah National University

Faculty of Graduate studies

Mathematical Principles and Practices for Internet of Things Data Analysis using Machine Learning Approach

By

Batoul Smeer Salameh Sulaiman

Supervisor

Dr. Mohammad Sharaf

Prof. Dr. Naji Qatanani

This Thesis is Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Computational Mathematics, Faculty of Graduate Studies, An-Najah National University, Nablus, Palestine.

2019

Mathematical Principles and Practices for Internet of Things Data Analysis using Machine Learning Approach

By

Batoul Sameer Salameh Sulaiman

This Thesis was defended successfully on 20/10/2019 and Approved by

Defense Committee Member	**Signature**
Dr. Mohammad Sharaf /Supervisor	**.....**
Prof. Dr. Naji Qatanani /Co-Supervisor	**.....**
Dr. Suhail Odeh /External Examine	**.....**
Dr. Amajad Hawash /Internal Examiner	**.....**

Dedication

الإهداء

سعيت ابتغاء الشكر فيما صنعت لي ... فقصرت مغلوباً وإني لشاكر

إلى صاحب الفضل و الكرم والقوة و الصدق والإخلاص و الذوق..

إلى الثقة العميقة..

إلى اللطف الكبير..

إلى قدوتي حضرة الإنسان الدكتور عماد النتشة

اهديك تعبك على هيئة إنجاز

"شكراً من كل قلبي"

(الإهداء مقدم مني ومن عائلتي)

Acknowledgments

Foremost, I thank almighty Allah for giving me the ability to finish this thesis. After that, I would like to thank my supervisors Dr. Mohammad Sharaf and Prof. Dr. Naji Qatanani for their support, and their belief in my ability to work on this topic. In addition to the great encouragement and motivation that I got from them to work on this thesis.

Finally, I would like to express my sincere gratitude to Dr. Emad Natsheh for his patience, motivation, immense knowledge and continuous support and I want to express my great appreciation because of the great help that I got from him which gave me the ability to finish my thesis.

الاقرار

أنا الموقع أدناه مقدم الرسالة التي تحمل العنوان:

Mathematical Principles and Practices for Internet of Things Data Analysis using Machine Learning Approach

أقر بأن ما اشتملت عليه هذه الرسالة إنما هو من نتاج جهدي الخاص باستثناء ما تمت الإشارة إليه حيثما ورد، وأن هذه الرسالة ككل، او أي جزء منها لم يقدم لنيل أي درجه أو لقب علمي أو بحثي لدى أيه مؤسسه تعليميه او بحثيه أخرى.

Declaration

The work provided in this thesis, unless otherwise referenced, is the researcher's own work, and has not been submitted elsewhere for any other degree or qualification.

Student's name: **اسم الطالبة:**

Signature: **التوقيع:**

Date: **التاريخ:**

Table of Contents

Mathematical Principles and Practices for Internet of Things Data Analysis using Machine Learning Approach

By
Batoul Smeer Salameh Sulaiman
Supervisors
Dr. Mohammad Sharaf
Prof. Dr. Najai Qatanai

Abstract

Internet of Things (IoT) environment generates the data continuously, these data need to be collected, analyzed in order to trigger an action. For many IoT applications, the locations information for the collected data is considered very important information, so a lot of localization techniques in wireless sensor networks (WSNs) are used for obtaining such information. In this work, we propose an artificial Feed-Forward Neural Network (FFNN) for the IoT sensor node localization. The proposed method was performed in many heterogeneous WSNs in which the anchor nodes distributed uniformly. Matlab software was used to implement this network which has a single hidden layer with 20 neurons. Two different training algorithms were used to evaluate this network which are the Levenberg-Marquardt algorithm and Gradient descent algorithm. The estimated locations for the sensor nodes obtained from the proposed FFNN (single layer) was compared with the results obtained from another structure for the network which is the Deep feed-forward (multilayer) neural network. Also, a comparison with a known localization algorithm 'Weighted centroid algorithm based on RSSI' was performed. Our results showed that the feedforward (single layer) neural network is a good

localization approach which gives us the most accurate estimated locations for the sensor node in WSNs.

Chapter One

Introduction

Rapid development in communication technologies have allowed the emergence of the Internet-connected devices. The term 'IoT' was introduced in 1982 by Peter T. Lewis at U. S. Federal communication s-commission (FCC). Internet of Things became a known idea in 1999 in a presentation to Proctor & Gamble introduced by Kevin Ashton. IoT is a huge network of interconnected physical objects based on a combination of the emerging technologies and the Internet. It can be considered as the future evaluation of the Internet and it provides the connectivity for everyone and everything to exchange the data over the Internet. In the recent years, IoT is increasingly spread, so the number of connected devices is increasing dramatically. IoT helps to mitigate latency in information because the information is collected automatically and it introduces a different kind of data using various types of sensors.
There exist many application domains in the IoT, like transportation, environment and telecommunication...,etc. Figure 1.1 shows different IoT application domains.

Data science provides an important contribution to make IoT applications more intelligent, and one of the most important parts of data science is Machine Learning, which include a broad range of algorithms applicable in different domains.

Machine learning plays an essential role in IoT aspects, that handle the huge amount of date generated by the connected devices. Its idea is to enable computers to learn from examples and experiences without

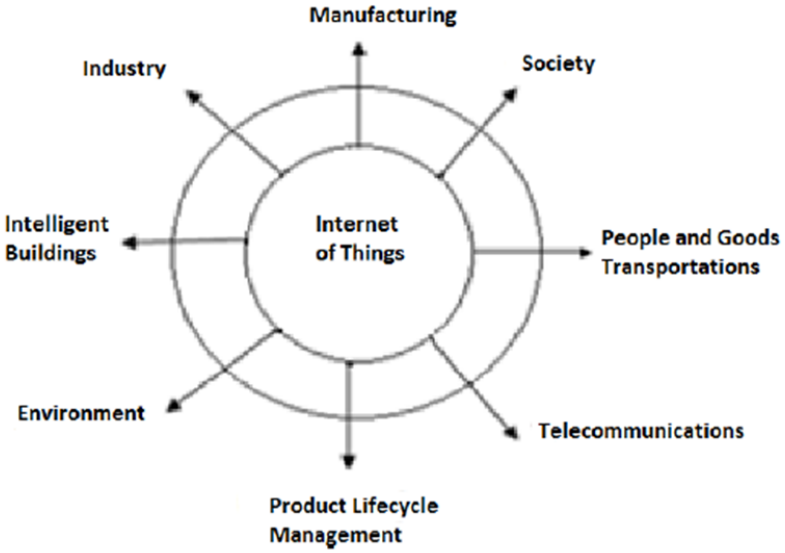

Figure 1.1: IoT application domains

being explicitly programmed.

One of the important topics use the machine learning in IoT environment is The sensor node localization process in wireless sensor networks (WSNs).

1.1 Motivation

A Systematic Literature Review (SLR) has been conducted about the application of machine learning methods in the IoT environment. We collect all the papers related to the IoT with Machine Learning. We get a lot of papers research that talks about these two topics together then we follow a three-stage research method consisting of planning, conducting and reporting.

At the first phase, planing, we investigated some questions in all the

selected articles, these research questions are:

1. What are the goals that achieved by applying the machine learning algorithms in the IoT environment?

2. Which is the machine learning algorithm used to achieve the desired goals?

3. Which are the empirical methods used in the selected studies?

The search strategy includes a manual search in google scholar for collecting pilot studies, and an automatic search with a specific search string on one of the most important scientific database, which is IEEE Xplore Digital Library. Some exclusion and inclusion criteria were applied for filtering the selected studies.

Inclusion and exclusion criteria:

These criteria made us able to select the studies to be considered in our research. A study was selected if it meets all the inclusion criteria, and it was removed from the collected studies if any of the exclusion criteria had been met.

Inclusion criteria:

- Studies proposing a machine learning algorithm using to solve a specific problem in IoT environment.

- Studies subject to peer review (e.g., paper published as a part of conference proceedings, journal papers).

- Studies published after or in 2010.

Exclusion criteria:

- Studies whose language is not English.
- Secondary studies (e.g., survey, SLR, etc.).

After that, we began the data extraction from these studies to answer the previously mentioned research questions. During the data extraction, we excluded some of the studies that it is not meet with our inclusion criteria.

At the second phase, conducting, we put the previously mentioned protocol in practice, so we got a 12 pilot studies related to the IoT and Machine learning topic from the manual search, and 1333 studies from the automatic search, then after filtering the selected studies through applying some exclusion and inclusion criteria we got 77 studies. During the data extraction, we exclude some studies that do not meet the inclusion criteria to get lastly 48 studies.

In the last phase, reporting, data extracted from the selected primary studies were collected in a spreadsheet.

Figure 1.2 shows our search strategy and the number of studies that are selected or removed during the subsequent stages.

After we do a comprehensive search and apply our protocol to collect these research papers, we noticed that one of the most frequent goals in research papers achieved by the machine learning algorithms is the

prediction goal. The most used machine learning algorithm applied in IoT environment is the Artificial Neural Network. In addition, most of the primary studies used the experiment and simulations for evaluation.

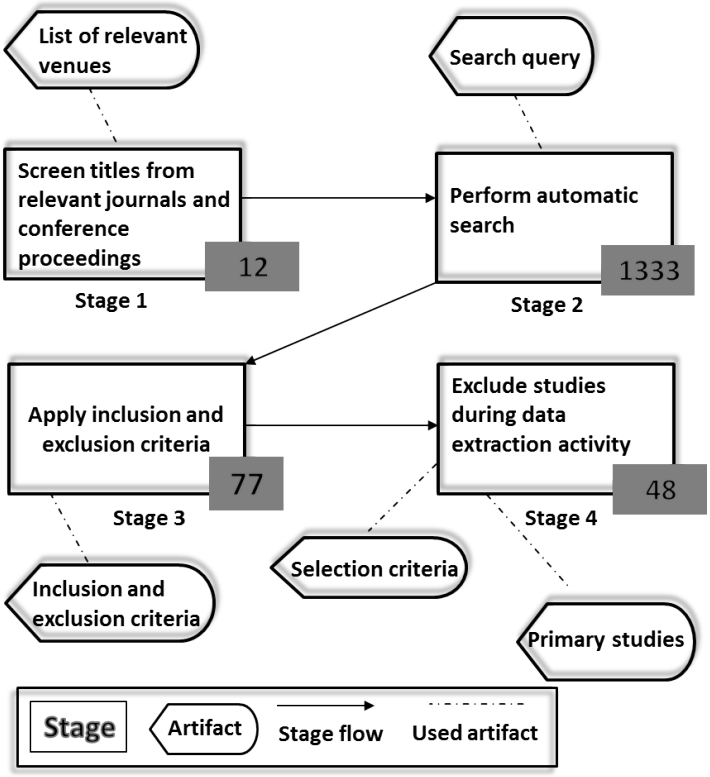

Figure 1.2: Multi-staged search and selection process

These results were taken into consideration to decide a suitable topic for this thesis. Therefore we searched for a topic about the Artificial neural network (ANN) for the prediction aim, and we observed that one of the most important topics used the artificial neural network for the prediction aim is the sensor node Localization in IoT.

1.2 Sensor node localiztion in WSNs

Wireless sensor network (WSN) is a subpart and key component of IoT. It consists of a large number of nodes that are connected wirelessly. WSN is considered a promising tool for data gathering to be used it in many domains like industrial fields, smart buildings, health care ..etc. Big challenges face WSN are energy consumption and memory storage. Due to this, the sensor node localization issue has been given a great attention, especially in data aggregation algorithms and geographic routing, in which the data are meaningless if there is no location information.

Traditional location techniques like GPS is a high-cost technique since it requires developed equipment and high energy consumption. In WSNs energy conservation is considered as the fundamental issue, so the costs and the number of nodes must be as small as possible. To solve this issue, a lot of localization algorithms that don't use GPS or employ it as an assist in some situations were developed.

In IoT applications, data collecting is done through the wireless sensor networks, so the localization issue can be referred to WSNs. Most of the localization algorithms were classified into two categories. The range-based algorithms [24, 25] that use some ranging methods like Time of Arrival (TOA) [20], Angle of Arrival (AOA) [37], and the Range-free algorithms [38, 17] that considered simple and low costs techniques in comparison with range-based ones since they require no special hardware,

so they are the widely used algorithms. Examples of these types of algorithms are DV-hop algorithm [40] and Centroid algorithm [8].

In this thesis, we are proposing to use a new approach for the sensor node localization, which is the Artificial neural network (ANNs), which is a collection of artificial neurons that are designed to recognize patterns in a set of data. In 1943 McCulloch and Pitts [31] created the first mathematical model for an artificial neuron.

ANNs are capable to predict the desired purpose through detecting the relationship between the inputs and the outputs. An optimization procedure is done for the network to get more accurate predicted location for the sensor nodes. The best results were observed by choosing the best structure for the neural network.

1.3 Problem statements

The main topics introduce by this thesis are:

- How to integrate WSNs with IoT.
- What are the used algorithms for the localization process in WSNs.
- How to employ the Neural network to Localize the sensor nodes in WSNs.
- What is the best structure for the ANN used for the localization process.

1.4 Proposed solutions

The solutions used to handle the localization process are:

- Weighted centroid localization algorithm (WCL) based on the Received Signal Strength (RSSI).
- Artificial Neural Networks (ANN)

Thesis structure

This thesis consists of 7 chapters, which is divided into a set of sections to describe this research. Basically, after the introduction, this thesis contains five chapters and the conclusion

In the second chapter, a background of the main topics will be introduced. The related Works were introduced in the third chapter. In the fourth chapter, we introduced the integration approaches between the IoT and WSNs. In addition to introduce on of the common used localization algorithms, which is the WCL algorithm, and we explain RSSI measurement method using the basic measure term path loss model. The last thing is the pseudo-code for the WCL applied in WSNs using Matlab software was discussed. In the fifth chapter, detailed sections were introduced about different types of the artificial neural networks (ANNs), the used activation functions, the training algorithms we derived, and clarification the advantages and disadvantages for each one. In addition, the pseudo-code for each algorithm is also clarified. In the sixth chapter,

simulation and results were introduced and our trials to build the best neural network structure were detailed, as well as a comparison between them. lastly, the results of our simulation were discussed.

Chapter Two

Background

2.1 Internet of Things (IoT)

Nowadays, all people use smartphones and a lot of other electronic devices. So basically, IoT is the interconnection between these smart devices on the Internet for sending and receiving data.

IoT can connect anything with anything else in the whole world like simple sensors, smartphones and wearable devices. It is very important for business communication transportation and many enterprises [28].

The Internet of Things (IoT) is also denoted as the Internet of Everything (IoE), or CyberPhysical Systems (CPS). Nowadays, over 9 billion 'Things' (physical objects) are connected to the Internet. In the near future, this number is expected to rise to approximately 20 billion, in which emphasize the importance of IoT.

2.2 wireless sensor networks (WSNs)

Recently, the development of wireless sensor networks have become an important research area. These networks consisting of a huge number of sensor nodes that are distributed in the target area, and have the ability to communicate wirelessly. The basic function for the wireless sensor network is to monitor the target areas for a long periods of time.

WSNs consist of many different types of sensors, like visual, magnetic, acoustic and radar. As a result, different types of sensors result in various kinds of applications as identified by Akyıldız and W.Su (2001) [1]. A lot

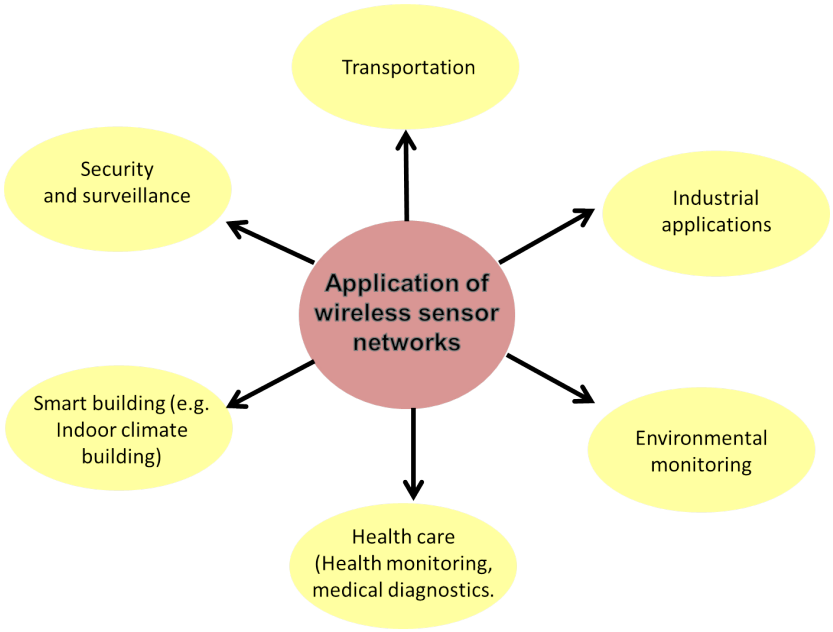

Figure 2.1: Wireless Sensor Network applications

of ambient conditions can be monitored by these sensors, such as humidity, temperature, movement and pressure. Hence, WSNs applications can be categorised into health, military, environment, home and commercial areas. In addition, It is also possible to extend this classification with more categories as summarised in figure 2.1

The sensor nodes (WSN) have many characteristics which are describes as follow:

(1) a small physical size.

(2) low power consumption.

(3) limited processing power.

(4) short-range communications.

(5) a small amount of memory storage.

Wireless sensor nodes (WSn) are used in many aspects of life, and

there are many reasons for this use which are presented as follows [7]:

- WSn are relatively cheap to use and very cost-effective.
- WSn have the ability to resist the environmental changing factors so they can be deployed anywhere.
- Due to the small sizes of WSNs, they consider robust and easy to apply.
- WSNs can have a huge number of sensor nodes, so they can cover a large indoors and outdoors areas.
- Due to quickly and efficiently deployment of WSNs in the areas of interest, a quick data collection can be achieved.

The sensor nodes can form different types of wireless networks which facilitate the life for humans. For example, patients in a hospital can be equipped with a vital sign sensor nodes which are able to measure the heart rate and blood oxygenation for the patients [29].
As shown in figure 2.2, through the WSNs organized by these nodes in easy way, the doctors can monitor the status of patients through smartphones or computers.

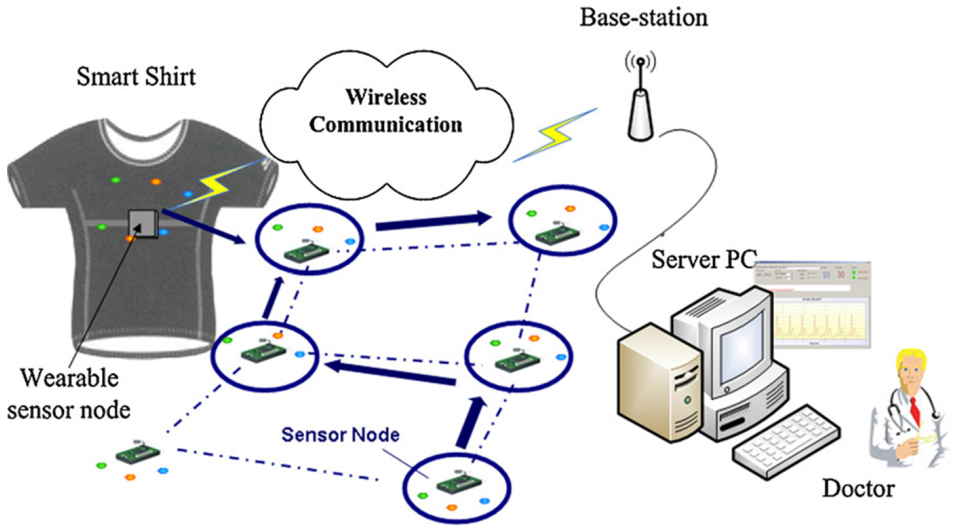

Figure 2.2: WSN for Hospital Monitoring

2.3 Artificial Neural network (ANN)

A biological neural network is an important part of the human brain. It is a complex system that has the ability to process huge amounts of data simultaneously. Biological network outperforms all modern high-end computers, also it has the ability to recognise and process different visual inputs in a fast way.

The similarity between the artificial neural networks with a biological neural network is that they get knowledge through learning that can be stored within connection strengths known as synaptic weights.

An ANN consists of a number of simple, small and interconnected processors, which are called neurons, and these neurons are arranged in consecutive layers. Chapter 4 introduces a detailed explanation of ANN.

Chapter Three

Related Works

Over the last years, a lot of researches have been carried out related to the localization process in the WSNs. Commonly, it is known that GPS [18] is not an efficient way for the sensor node localization in WSNs due to the high cost and energy consumption for the GPS dependent devices [5].

Therefore, for a robust, flexible, practical and inexpensive localization in WSNs, many researchers investigate innovative ideas for the localization process. Many survey on the localization schemes are available [22, 19, 51].

The localization schemes for WSNs are commonly divided into range-based and range-free. For example for range free schemes, the works presented in [58] and [16] focus on Centroid Localization Algorithm. In [12] Convex position estimation (CPE) algorithm was introduced, its basic principle is to define an estimative rectangle (ER) which bounds the overlapping region and regards the centre of this rectangle as the estimative location of the unlocalized node. Niculescu proposed the DV-hop algorithm [36], which is improved by many researchers, like, Hongyang Chen [9] in which it is considered more complicated than Centroid and CPE.

For the range based localiztion algorithms, the locations of sensor nodes can be estimated by geographical calculations such as triangulation [2, 13] and trilateration [44, 34].

A new approach is proposed to use the Artificial neural network for

the localization process by many researchers like, Kumar *et al*, [26] proposed to use implement a feedforward neural network with three hidden layers 12-12-2 for the node localization in the WSNs. In [14] the fingerprint method of indoor localization using feedforward neural network is presented. In addition, Battiti *et al* [4], proposed a method based on neural networks for reducing the errors in determining the location of mobile node. An artificial synaptic network (ASN) which is a novel multilayer neural network was proposed in [48]. The distance between the nodes was estimated through the TOA method. In addition, a comparison with the radial basis function neural network model (RBF) is considered. the results found that the ASN is better in term of the number of iterations and Root Mean Square Error (RMSE).

In the field of WSNs, the area and the topology of the network is an important factor affects the localization process. Because of that, we need to explore an ANN model to be able to predict the sensor nodes locations in different WSNs areas with trying to take into consideration different criteria. From these criteria, we can point out, finding a simple neural network topology to achieve an accurate, low-cost localization process with supporting different areas for a specific WSNs topology. In this work, an artificial FFNN model for different square-based topology areas is proposed based on the WCL algorithm, such that we obtain a neural network model able to predict the sensor node locations in various squared areas with more accurate results.

Chapter Four

Localization algorithms for WSNs based on Internet of Things.

4.1 Wireless Sensor Networks and Internet of Things

For communication between the nodes in WSNs, it is very necessary to have a centralized system. The necessity of this system leads to develop the notion of the internet of things (IoT). As a result, in parallel to WSNs, the idea of IoT is developed.

IoT is considered at a higher level than WSNs, which means that, WSNs are often technologies used within IoT systems. A large number of sensors can be used for gathering the data individually and then send these data through a router to the internet in IoT systems.

IoT provides interaction between people and environment as shown in Fig 4.1

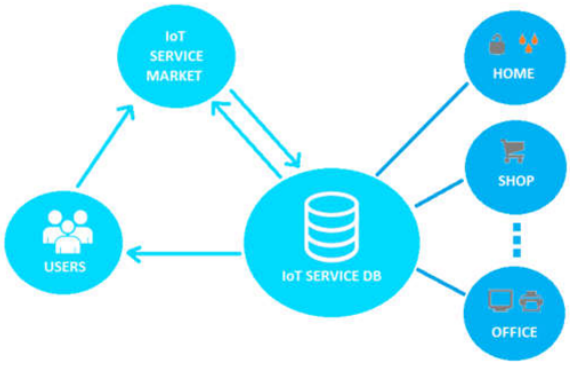

Figure 4.1: Internet of Things

In IoT systems, all the used sensor nodes send their information directly to the internet. contrariwise, in WSNs, there is no direct connection to the internet. Instead, the different sensors connected to several kinds of (sinks) central node or router.

4.2 Integration Approaches

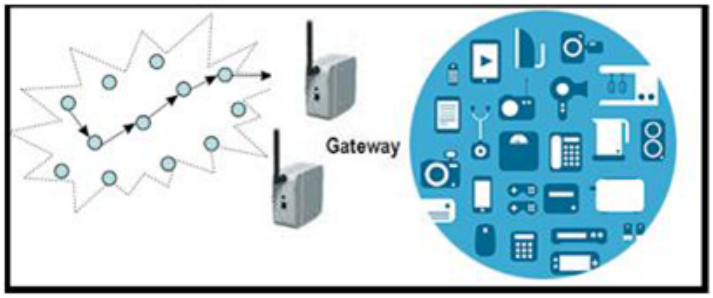

Figure 4.2: Integration of IoT with WSNs

A very important point is to know the integration approaches which can be applied to integrate WSNs in IoT systems 4.2. Three main approaches are discussed here for connecting WSNs to the Internet, taking into account the WSN integration degree into the Internet structure [10]. In the first approach, the connecting between the independent WSN and the Internet is through a single gateway as shown in fig 4.3

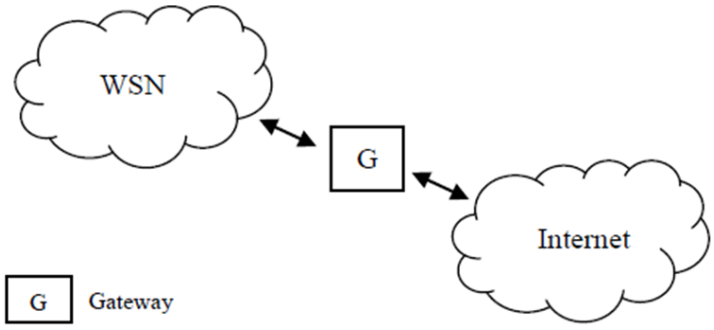

Figure 4.3: Independent network

It is obvious that this approach presents a single point of failure because of the gateway uniqueness. Therefore if a gateway dysfunction exists then the connection between the WSNs and the internet would break down.

The second approach showed an increasing integration degree, this approach forms a hybrid network which is composed of independent networks structure where some of dual sensor nodes have the ability to access the internet as shown in fig 4.4

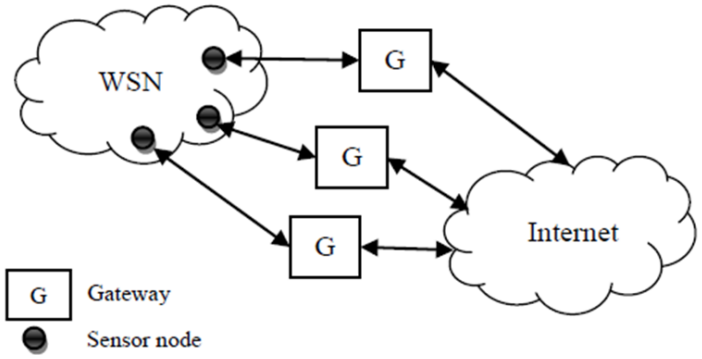

Figure 4.4: Hybrid network

In the third approach illustrated by fig 4.5, multiple sensor nodes can connect to the Internet in one hop.

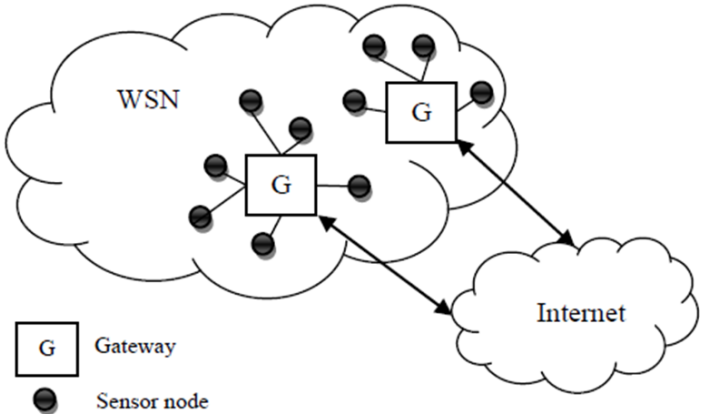

Figure 4.5: Access point network

Now, with several gateways and access points as in the second and third approaches, they will not present such weakness, so to ensure the network's robustness, the last two approaches would be preferred.

4.3 Localization process in Wireless Sensor Networks

Knowing the location of the sensor node in the network is a very critical issue for many applications, that is because the users usually need to know not only what happens but where interesting events take effect too.

For example, in the hospital, the knowledge about where the patient is will help the doctors to arrive at the right place as fast as possible in the urgent cases [29]. In a disaster relief operation the WSN is using to locate survivors in the collapsed building. It is very critical for the sensors to report the information location [53].

There are alot of localization techniques used to provide location information for each of the deployed sensor node in a wireless sensor network. Global Positioning System (GPS), is one of the famous ways to identify the location of the nodes but GPS does not work efficiently as the line of sight between satellite and receiver is not always available due to the high buildings and the dense tree areas. In addition, the GPS has some limitations such as large cost and power consumption.

Internet of Things (IoT) combines many technologies [15], such as Internet, Wi-Fi, Bluetooth, 3G, etc to provide location based service that enables different ways to get the location information of various objects. Recently IoT is considered a popular and upcoming topic in wireless sensor networks and the knowledge of sensors location is a very

important issue, so it is extremely useful to propose a proficient procedure for sensor localization.

Therefore many localization estimating methods to estimate the location of nodes in WSNs are proposed, these methods classified into two categories according to the mechanism used for the location estimating [46], such as:

1. Range-based position algorithms

Range-based approach determine the absolute distance estimate or angle estimate between nodes based on range information such as Received Signal Strength Indicator (RSSI), Time of Arrival (TOA), Time Difference of Arrival (TDOA), and Angle of Arrival (AOA), then the desired position of the nodes is estimated with the help of triangulation or trilateration techniques. This scheme has some drawbacks such as, an additional range devices are needed, which results to increase the cost and to consume more energy.

2. Range-free position algorithms

Range-free scheme is a cost-effective alternative method since there is no need for additional range hardware, so it is more popular among all other range-free localization algorithms due to its simplicity. This scheme applies distance approximation algorithms to determine the node's location. In range-free localization schemes, the sensor nodes knowing their positions are called Anchor nodes, while the others are called Normal or Unknown nodes. In this scheme, unlike the range-based scheme, the con-

nectivity information is used, and it is indicate how two nodes are closing together, these information can be the hop count between the two nodes. The Unknown nodes gather the connectivity information and the position of anchors, after that they calculate their own positions through a specific localization algorithm.

The two main range free localization algorithms are the Centroid and Distance Vector-hop(DV-Hop).

In DV-Hop algorithm, the position of the Unknown nodes is calculated depending on hop count. So to localize a node, multi-hops calculations are performed. The multi-hopping between the nodes consumes large energy, so the main focus will be on energy efficient algorithm which is Centroid algorithm where the distance between the nodes and the centroid is calculated as it will be described in the follows sections.

4.4 Centroid Localization algorithm

Centroid algorithm [52] is a range-free localization algorithm, it is put forward by professor Bulusu at the University of California. Its principle is that when the unknown node exists within the range of anchor nodes, the un-localized nodes locate it self as the centroid of all the received beacon's positions. Centroid algorithm is simple and easy to implement. Many cases were demonstrated as in Figure 4.6

In Figure 4.6 (a), when the unknown node is within the scope of the anchor node communication. Figure 4.6 (b) shows that when the

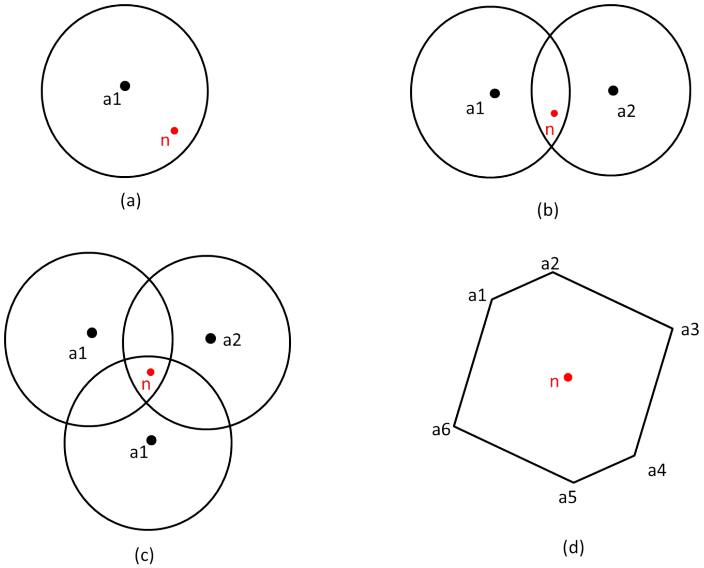

Figure 4.6: The principle diagram of the centroid localization algorithm

unknown node exists in the second anchor node scope. In the same way, when unknown nodes exist in the third anchor node communication range as shown in Figure 4.6 (c). As shown in Figure 4.6 (d), the unknown node coordinates in the case of multiple anchor nodes in vertices of a polygon are the centre of this polygon, so in the case of six anchor nodes with coordinates of (x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4), (x_5, y_5), (x_6, y_6), depending on the centroid localization algorithm the estimated location of the unknown node coordinates (X, Y) is:

$$\begin{cases} x = \dfrac{x_1+x_2+x_3+x_4+x_5+x_6}{6}, \\ y = \dfrac{y_1+y_2+y_3+y_4+y_5+y_6}{6} \end{cases} \tag{4.1}$$

from these mathematical expressions, it is noticed that the six anchors nodes location information has exactly the same effect in the process of the unknown node localization. Therefore, this estimation method in the process of actual position is not reasonable, since the nearer anchor node has a greater effect to the estimated location of the unknown node so its location will be closer to the nearer anchor node. As a result an improvement is introduced to this algorithm in the next section

4.5 Weighted Centroid Localization algorithm

Due to the low accuracy in location estimation of the centroid localization method, the development of a weighted centroid localization (WCL) algorithm based on RSSI [52] was stimulated. The basic idea of the weighted centroid algorithm is to take the anchor nodes that located within the communication range into consideration and give more influence to those anchors which are nearer to the unknown node and use the data received from them for the localization process. WCL introduces the weights of the beacons depending on their received signal strength indicator (RSSI) towards the unknown node. RSSI can reflect the distance between the two nodes.

Assume that we have **n** anchor nodes in wireless sensor network, their coordinates (x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4), ...,(x_n, y_n), respectively. Also let $w_1, w_2, ..., w_n$ are the weights of the RSSI of the anchor nodes By the weighted centroid algorithm the estimated coordinates computed

through the follwing formula:

$$\begin{cases} X_{est} = \frac{w_1x_1+w_2x_2+...+w_nx_n}{w_1+w_2+...+w_n}, \\ Y_{est} = \frac{w_1y_1+w_2y_2+...+w_ny_n}{w_1+w_2+...+w_n}, \\ w_i = \frac{RSSI_i}{RSSI_1+RSSI_2+...+RSSI_n}. \end{cases} \tag{4.2}$$

Where, X_{est} and Y_{est} are the unknown sensor node coordinates, w_i is the anchor node weight.

So we can conclude that from equation 4.2 that the signal transmission distance is the important factor in the localization process, such that the grater distance the smaller received signal strength, so the value of RSSI reflect the distance between the nodes.

RSSI Parameter Calculation in WSN

RSSI (Received Signal Strength Indicator) is a more common name for the signal value. It is the strength that one device is receiving from another device and it is a measurement of the power that present in the received radio signal measured in decibels.

RSSI measurement calculates the signal loss in the dissemination process with the theory of signal propagation model, there are some measure terms which have an important role in RSSI measurement as follows:

1. Path Loss Model

Wireless signal propagation path loss influence the positioning preci-

sion of the localization algorithm in a great way, it is a measure of how much signal power loses by the device over a given distance. Using the theoretical models of the wireless signal transmission, we have the ability to locate by the signal strength. The most common models to estimate the distance based on the attenuation are the free space propagation model and the log-distance path loss model

In the free space, the signal strength attenuates logarithmically with respect to the distance between the transmitters and receivers. The assumption in the free space model is the ideal communication environment, that has an only one unobstructed straight path. H.T Friis [50] proposed to use equation 4.3 to calculate the received signal ability strength in the free space when the distance to the transmitter is d :

$$P_r = \frac{P_t G_t G_r \lambda^2}{(4\pi d)^2} \tag{4.3}$$

where, P_t is Received signal power in Watts between the transmitter receiver, G_t is Gain of the Transmitter, G_r is the Gain of the Receiver, $\lambda = \frac{c}{f}$ is Wavelength of transmission in meters, where $\mathbf{c} = 3 * 10^8$ is the velocity of the electromagnetic and $\mathbf{f}$ is the frequency of transmission in Hertz.

In the real environments, the obstacles interference, diffraction and multipath and other factors, cause difference between the radio propaga-

tion path loss in comparison with the theoretical value, so the shadowing distribution model would be more realistic environment.

Log-Normal shadowing model [57] is a general extension to the free space model. If the environment contains some objects like trees and buildings, there is some part of the transmitted signal gets affected by reflection, absorption, scattering and diffraction. This effect is called shadowing.

Log Normal Shadowing path loss model [49] is formally expressed as:

$$PL(d) = PL(d_0) + 10nlog\frac{d}{d_0} + X_\sigma \tag{4.4}$$

Where, $PL(d)$ is the path loss after distance d, n is the path loss exponent which measures the rate at which the RSSI decrease with distance, the value of n depends on the specific propagation environment, X_σ is a zero mean Gaussian distributed random variable whose mean value is 0 and it reflects the change of the received signal power in certain distance. This variable is used only when there is a shadowing effect. If there is no shadowing effect, then this variable is zero, d_0 is reference distance and usually equals 1 meter, PL(d_0) is a known reference power value at a reference distance d_0 from the transmitter.

2. Received Signal Power at Reference distance.

Assume that A is the received signal power in the distance d_0 between

the transmitter and the receiver, the formula 4.5 can be generated.

$$A = P_t - PL(d_0) \tag{4.5}$$

P_t: is power of transmitter

PL(d_0): is a known reference power value from the transmitter to reference at a distance d_0. The reference path loss $PL(d_0)$, it is the power value at reference point and it can be obtained using Friis equation 4.6 [6, 50] or by field measurements at d_0.

$$Pr_0 = \frac{P_t G_t G_r \lambda^2}{(4\pi d_0)^2} \tag{4.6}$$

where, P_t is Received signal power in Watts between the transmitter receiver, G_t is Gain of the Transmitter, G_r is the Gain of the Receiver, $\lambda = \frac{c}{f}$ is Wavelength of transmission in meters, where $\mathbf{c} = 3 * 10^8$ is the velocity of the electromagnetic and **f** is the frequency of transmission in Hertz.

3. Received Signal strength RSSI.

RSSI can be calculated from the following equation [33]:

$$RSSI = P_t - PL(d) \tag{4.7}$$

then the RSSI will be

$$RSSI = A - 10nlog\frac{d}{d_0} + X_\sigma \tag{4.8}$$

Through these measure terms the value of RSSI between the anchor node and the unknown node was obtained

Weighted Centroid Localization Algorithm Steps:

suppose that there are **n** anchor nodes in a wireless sensor network, and (X,Y) is the unknown sensor node coordinates, then the location estimation steps based on the weighted centroid localization algorithm are defined as follows:

1. The anchor nodes send their information, including their own position coordinates information to the surrounding nodes.

2. Unknown node receives the broadcast information from the anchor nodes and calculate the average received signal strength from anchors.

3. While the unknown nodes receive anchor node information, formulas for the anchor node coordinates = x_1, x_2, x_3, ...,x_n and the received signal strength RSSI = $RSSI_1$, $RSSI_2$, $RSSI_n$ are established.

4. Using formula 4.2 we can calculate the coordinates (X_{est}, Y_{est}) of the node under test.

5. For evaluating of the precision of result, a definition of localization

error E for n anchor nodes, CR Communication Range, (X_{est},Y_{est}) the estimated position by WCL and (X,Y) actual positions, since the error is impacted by communication radius, the error is given by [55, 23] :

$$E = \frac{\sum_i^n \frac{1}{CR}\sqrt{(Xest - X)^2 + (Yest - Y)^2}}{n}$$

pseudo-code for WCL

Require: Anchors Locations (x_i,y_i) with respective RSSI values.

1: Divide the area into k number of grid.

2: Let in each grid have the anchor node (anch).

3: Distribute the Unknown nodes (unk) randomly in the desired area.

4: **IF**

The distance between unk & anch $\leq$ CR

5: **THEN**

Compute the value of RSSI for each anchor within the communication range through using Log Normal Shadowing path loss model.

$PL(d) = PL(d_0) + 10nlog\frac{d}{d_0} + X_\sigma$

6: For each anchor, Compute the weight depending on RSSI.

$$wi = \frac{RSSI_i}{RSSI_1 + RSSI_2 + ... + RSSI_n}.$$

7: Calculate the estimated X_{est},Y_{est} coordinate of the node under test.

$$X_{est} = \frac{w_1 * x_1 + w_2 * x_2 + ... + w_n * x_n}{w_1 + w_2 + ... + w_n},$$
$$Y_{est} = \frac{w_1 * y_1 + w_2 * y_2 + ... + w_n * y_n}{w_1 + w_2 + ... + w_n}.$$

8: Calculate the estimated error position between the actual position (X,Y) and the estimated positions (X_{est}, Y_{est}.)

$$E = \frac{\sum_i^n \frac{1}{CR}\sqrt{(Xest - X)^2 + (Yest - Y)^2}}{n}$$

9: **END IF**

Chapter Five

Artificial Neural Network

5.1 Introduction

Neural networks are considered from the most robust and widely used machine learning algorithms and they designed to recognize patterns and predictions. These patterns are numerical contained in vectors, which all the real-world data have to be translated for it, like images, sound, text ... etc. These networks interpret the data through a kind of machine perception, labelling or clustering the raw input data.

An artificial neural network consists of a number of simple, small and interconnected processors called neurons. These neurons are connected through weighted links, each of which has a numerical weight associated with it, in which signals passing through them from one neuron to another, then a number of input signals are received by each one of the neurons through its connections and it is produce just an output signal.

The artificial neural network is made up of many layers, and the artificial neurons in the network are arranged along these layers. The weights of the links are modified to make the input/output behaviour of the network is suitable with the target environment.

Choosing the structure of the artificial neural network is considered the most important issue. The number of neurons and hidden layers is changed. In addition to choose the activation function and training algorithm to be used

Basically, the neural network consists of three different layers:

1. Input Layer (All the inputs are fed in the model through this layer).
2. Hidden Layers (These layers are used for processing the received inputs from the input layers, there is a possibility to be more than one hidden layer.).
3. Output Layer (The data after processing is called the 'outputs' and they are available in the output layer).

There exist different types of artificial neural networks, like, feedforward neural network (FFNN), convolutional neural network, recurrent neural network... etc. These types of the neural network can be supervised or unsupervised, in our case we have the input and the desired output so we rely on the supervised types. Therefore in our study, we used the feedforward neural network and deep feedforward neural network (DNN).

In this chapter we will introduce the concepts for the feedforward neural network and its types, and then we explain the effect of each type of the activation functions and the training algorithms on the performance of the network, and explain the stages of back propagation learning algorithm.

5.2 Feedforward Neural Network (FFNN)

Feedforward neural network is a multilayer perceptron with one or more hidden layers. The neural network type depends on the arrangement and number of the composing neurons. The network architecture can be

represented as a directed graph, where each node represent a neuron and the edges represent connectives between theses neurons.

The neuron is the basic building element of an ANN. The first thing is that a neuron receives some signals from its input links, and then it computes a new activation level and passes it as an output signal through the output links.

The forms of the inputs can be raw data or outputs of other neurons. Also, the outputs can be either a final solution to the problem or input to other neurons. Figure 5.1 shows a typical neuron.

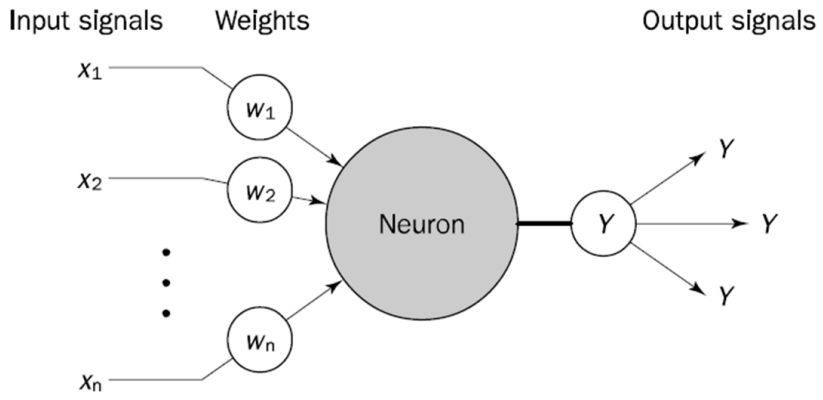

Figure 5.1: Diagram of a neuron

There are three types of neurons contained in feedforward Neural Network:

1. Input neurons: these types of neurons take input vector and they don't do any type of computation, they just pass the input vector to the next neurons.

2. Output neurons: those neurons receive signals from the previous neurons and then transform it using the net weighted formula with ap-

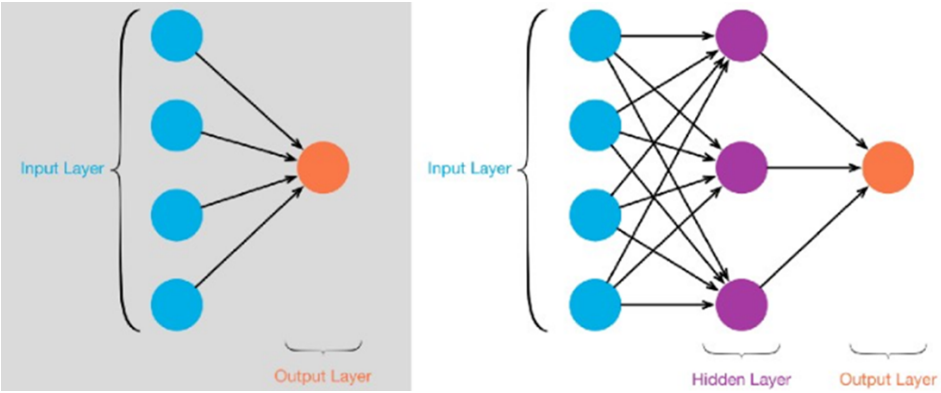

Figure 5.2: Left: single layer perceptron; Right: Multi-layer perceptron.

propriate activation function. These values represent the output values of the whole neural network.

3. Hidden neurons: are considered as the basis of the neural network, they receive the signals from the input neurons or previously hidden neurons and handled them through the net weighted formula and the appropriate activation function then pass the result signals to the next hidden or output neurons. In Feed Forward Neural Network, neurons are distributed into different layers.

Input and output neurons are distributed in separated layers which are the input layer and output layer, respectivly. Hidden neurons form the hidden layers which can be one or more than a hidden layer. Every neuron in feedforward neural network (except the input neurons) is connected through synapses with all neurons of the previous layer as illustrated in figure 5.2

There are a lot of different learning algorithms that multilayer neural networks used to learn but the most common used method is back-

propagation which was derived by many of researchers in the early 60's, and the researchers who help in proposing this algorithm are Arthur E. Bryson and Yu-Chi Ho in 1969. Paul Werbos was first to propose that back propagation could be used for neural networks after analyzing his 1974 PhD Thesis [41], section 5.4 has a detailed explanation about back-propagation.

5.2.1 Feedforward (single layer) Neural Networks

This type is consider the first and simplest type of FFNN. In this network the information moves only from the input layer directly through the single hidden layers to the output layer. This network consist of just one hidden layer as shown in figure 5.3.

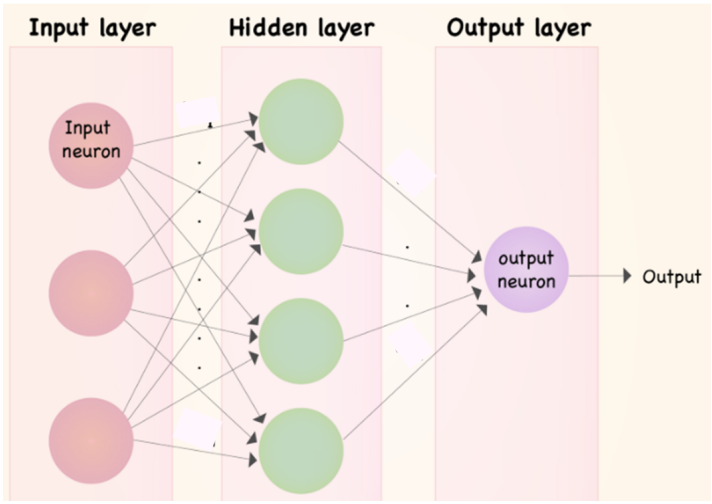

Figure 5.3: Feed Forward (single layer) neural network

FFNN in general is a supervised learning system consisting of a large number of neurons, each neuron makes simple decisions and then introduces these decisions to another neuron which are organized in intercon-

nected layers.

In FFNN with just one hidden layer, any continuous function of the inputs can be represented. In figure 5.4 a feedforward (single layer) neural network is shown, and the result is represent a continuous function.

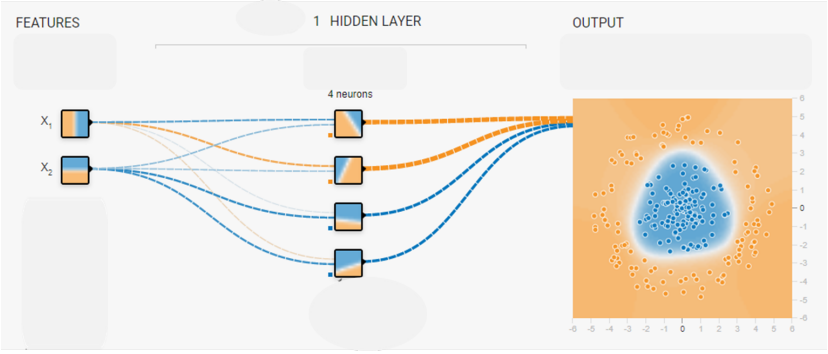

Figure 5.4: FFNN represent a continuous function of the input signals

The number of neurons in the hidden layer can be changed to choose the best structure of the network in addition to choose the suitable activation function and training algorithm.

5.2.2 Deep Feedforward (multilayers) Neural Networks

Deep neural networks differ from the feedforward (single layer) neural networks by their depth i.e, the number of layers the data must pass through in a multistep process of pattern recognition.

Deep Neural Networks (DNNs) used to solve complex problems that need a lot of effort to get good results, and it is composed of the input and output layer and two or more hidden layers.

In feedforward (single layer) neural network any continuous function of the inputs can be represented, while in Deep feedforward (multilayers)

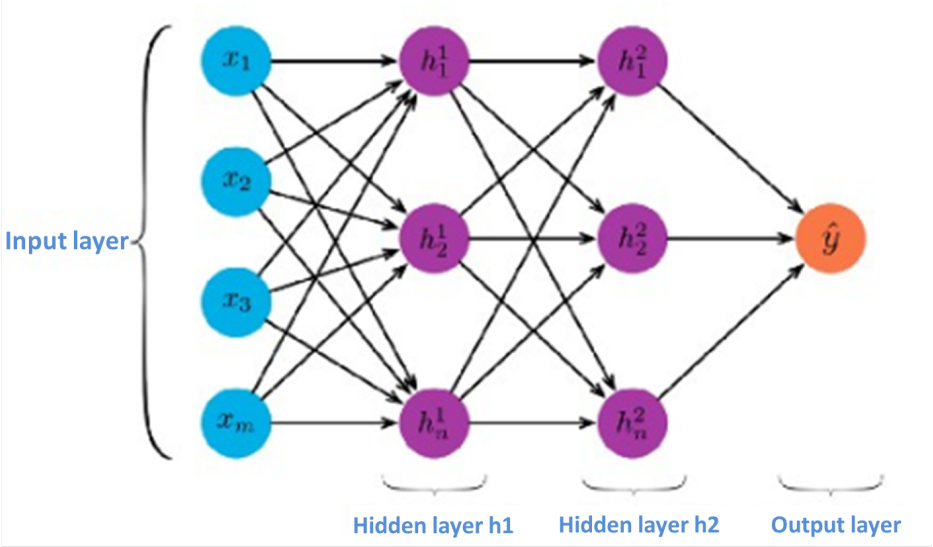

Figure 5.5: Deep neural network with two hidden layer

neural network, even discontinuous functions can be represented.

A more complicated function can be represented through the deep neural network because of the increased number of the hidden layer. Figure 5.6 represents deep neural network with three hidden layers that have the ability to classify the input features through the prediction of a discontinuous function.

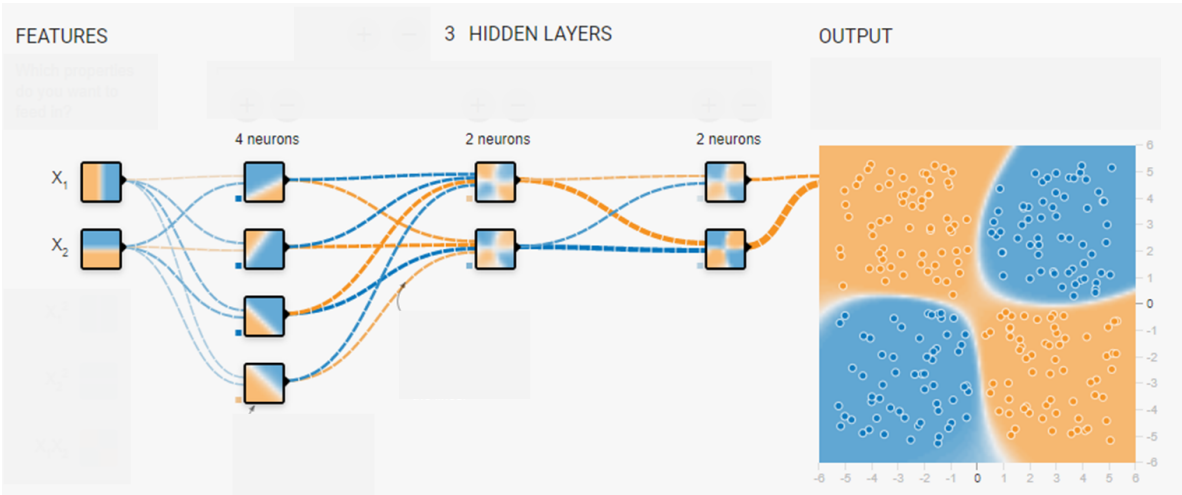

Figure 5.6: Deep neural network represent a discontinuous function of the input signals

In DNN when an additional layer is added, the computational burden will increases.

5.3 Activation Functions

Activation functions are the processing that taks place after the input is passed into the neuron, they are mathematical equations used to determine the output of the ANN. This function is attached for each neuron in the network and they also help in normalizing the output of each neuron in the range between -1 and 1 or between 0 and 1.

There are many practical activation functions. Four common and useful activiation functions are: 1. the step, 2. sign, 3. linear, 4. sigmoid, and 5. hyperbolic tangent function (tanh).

The activation function f defines the output Y from the neuron in terms of its net input X. There exists common activation functions [11] [42], such that, step function, linear function and the sigmoid function.

- **STEP ACTIVATION FUNCTION**

  Step function in equation 5.1 is the hard-limit transfer function shown in fig 5.7, this function limits the output of the neuron to either 0, if the net input X is less than 0; or 1, if X is greater than

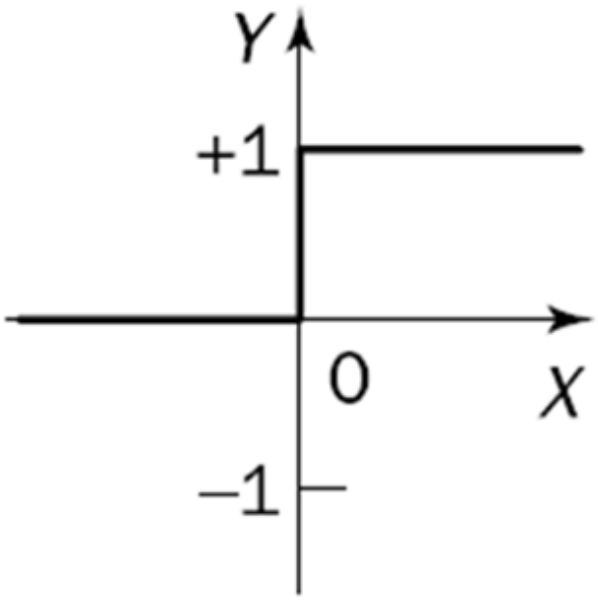

Figure 5.7: Step activation functions of a neuron

or equal to 0.

$$Y = \begin{cases} 1 & if \quad X \geq 0, \\ 0 & if \quad X < 0 \end{cases} \tag{5.1}$$

The problem with the step function is, there is no possibility for a multi-value output, that means, for example, that there is no possibility for classifying the inputs into one of several categories. The sign activation function is shown in equation 5.2, also called hard limit function.

$$Y = \begin{cases} 1 & if \quad X \geq 0, \\ -1 & if \quad X < 0 \end{cases} \tag{5.2}$$

- **LINEAR ACTIVATION FUNCTION**

  In linear activation function given in equation 5.4, the inputs were taken and then multiplied by the weights of each neuron as in equation 5.3 and then using the linear activation function an output is created proportional to the net weighted input. That's mean, the linear activation function is better than the step activation function because it allows multiple outputs, unlike the step function that is allowing just yes and no. fig 5.8 shows the linear activation function

$$X = \sum x_i w_i \tag{5.3}$$

  Where, x_i is the input values and w_i is the corresponding weights

$$Y = X \tag{5.4}$$

  Thus, the output signal of a neuron with linear activation function is equal to the neuron weighted input. The activation derivative for the linear function is given by:

$$Y' = 1 \tag{5.5}$$

  and the second derivative is:

$$Y''(X) = 0 \tag{5.6}$$

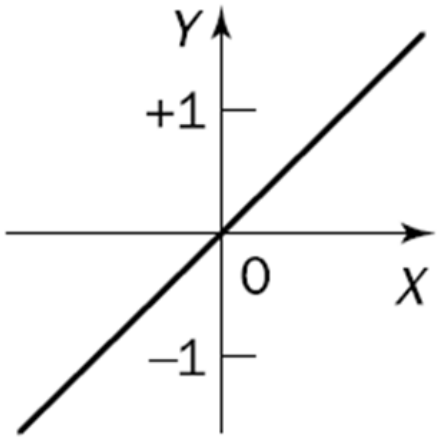

Figure 5.8: Linear activation function

A major problem of the linear activation function is that there is no possibility to use backpropagation "gradient descent" to train the network since the derivative of the linear function is a constant so it has no relation to the input X. Consequently, there is no possibility to go back and realise the weights in the input neurons that can provide a better prediction.

- **SIGMOID FUNCTION**

  The sigmoid function [21] given by equation 5.7 transforms the input that could have any value between $\pm\infty$, into a reasonable value between 0 and 1. This function can be used in the back-propagation networks.

$$Y = \frac{1}{1 + e^{-X}} \tag{5.7}$$

Sigmoid is the most used activation function when we construct

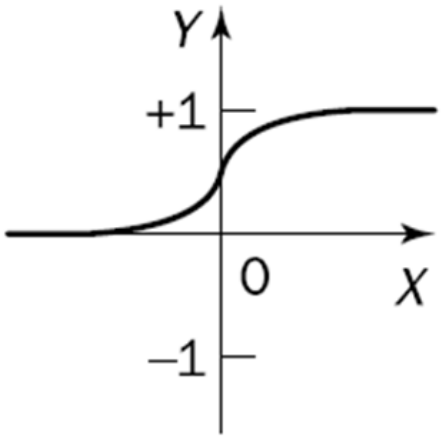

Figure 5.9: Sigmoid activation function

neural networks. This function is monotonic, it gives good balance between a linear and non-linear behaviour.

The hyperbolic tangent function defined by the equation 5.8 [3] and showed in fig 5.10 is an important example of a sigmoid function (s - shaped) but it's output range is from (-1 to 1). Thus strongly negative inputs in *tanh* function are mapped to negative outputs. Also, only zero-valued inputs are mapped to near-zero outputs which make the network less likely to get "stuck" during training in contrast with the sigmoid function which has the fact that if a strongly-negative input is provided to the logistic sigmoid, it output values will be very close to zero, which can cause a neural network to get "stuck" during training.

$$Y = tanh(X) \qquad or \qquad Y = \frac{2}{1 + e^{-2X}} - 1 \tag{5.8}$$

This looks very similar to sigmoid function

$$tanh(X) = 2simoid(2X) - 1$$

Its activation derivative is given by

$$Y' = 1 - tanh^2(X)$$

The second derivative of this activation function is:

$$Y'' = -2tanh(X)(1 - tanh^2(X))$$

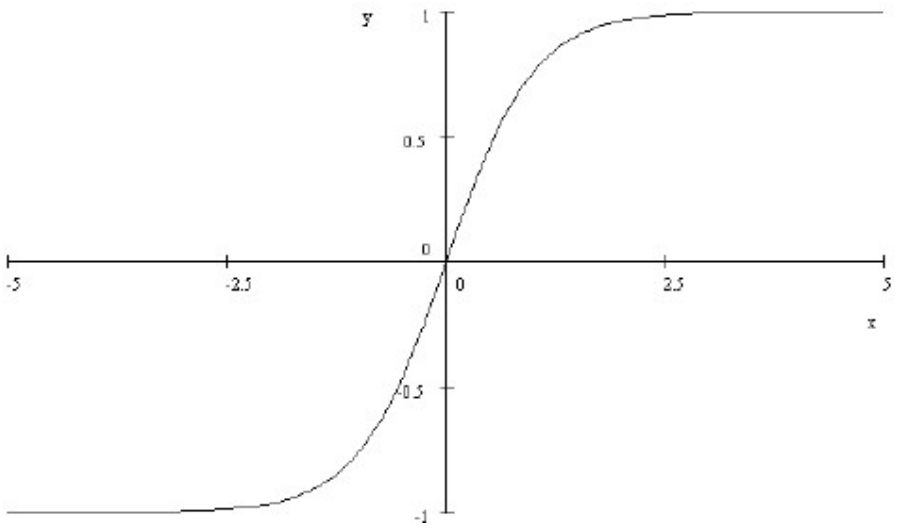

Figure 5.10: Hyperbolic tangent Activation Function

5.4 Back-propagation algorithm

The main principle for the back propagation approach is modelling a given function by modifying the weights of input signals for producing an expected output signal. The network is trained through a supervised learning method, where the error between the actual network's output and a known expected output is presented to the network and used for modifying its internal state.

Technically, the back propagation algorithm is a method used for training the synaptic weights in a multilayer feedforward neural network. As such, this algorithm requires a network structure with one or more layer where each layer is fully connected to the next layer.

Back-propagation algorithm searches for the weight values which minimize the total error of the network over a set of training data.

In back propagation algorithm the training input values are introduced to the input layer, after that the network propagates the input values from layer to layer until the activations of output are generated through the output layer it then backward propagates the output activations. Then the error is calculated, which means the difference between the desired and the actual output, and this error propagated backwards through the network from the output layer to the hidden layers until we reach to the input layer and the weight for each synaptic is modified as the error is propagated.

Back-propagation process is determined by, the network's architecture (the connections between neurons), the activation function used by the neurons in the network and the learning law that is used to determine and adjust the weights.

The following basic steps provide the foundation to implement the back propagation algorithm and to apply it to our own predictive modeling problem.

1. Initialize Network.

Consider a neural network with a three layers as shown in Figure 5.11, and assume that i, j and k are the indices refer to neurons in the input, hidden and output layers, respectively. And assume that x_1, x_2, . . ., x_n are the input signals which are propagated from left to right through the neural network, and the error signals, e_1, e_2, . . ., e_n, from right to left. The weight between the neurons in the input layer and the neurons in the hidden layer is denoted by w_{ij}, and the weight between the neurons in the hidden layer and the neurons in the output layer is denoted by w_{jk}.

The first thing we should start with is the creation of a network to be ready for training, so the data preparation is the first important thing we should start with, so for having an efficient Neural network there are some preprocessing steps on the inputs and the desired outputs in the neural network. The network's inputs have to be transformed into better form to be used. This can be done through the Normalization process

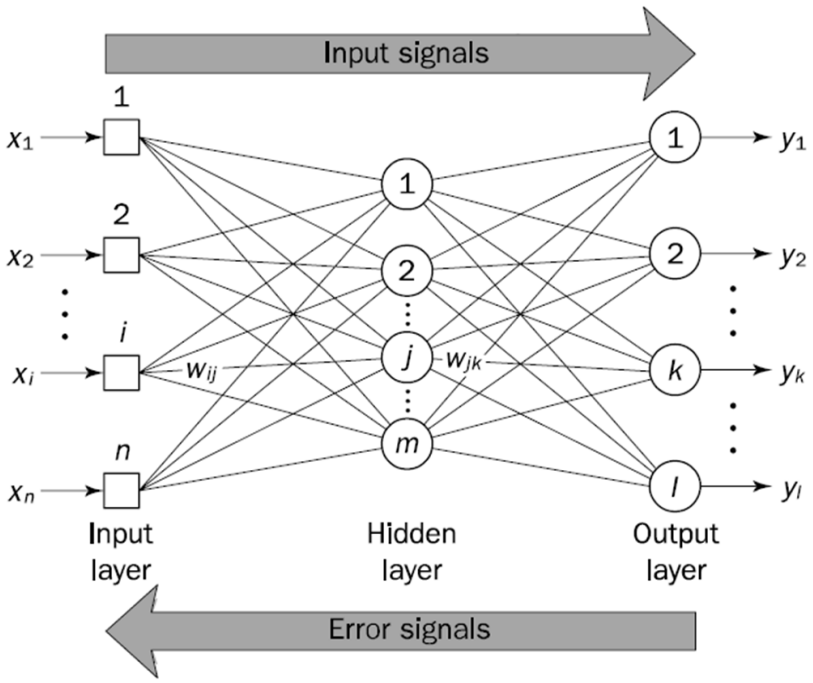

Figure 5.11: Three layer back-propagation neural network

which is a process for rescaling the data from the original range into the range between -1 and 1.

If we have unscaled input values, it will result in a slow or unstable learning process. Since the neural network is tasked to learn how to combine the inputs values through a series of linear combinations and nonlinear activations, the parameters associated with each of these inputs will also exist on different scales which will lead to the slowness or instability in the learning process.

There exist different normalization techniques used for increasing the reliability of the training in neural networks since through normalizing all of the inputs to a standard scale, this will allow the network to learn the optimal parameters for each input node more quickly.

A common used normalization method is **Min-Max Normalization** [43] this method rescales the input features or outputs from a specific range of values to another new range. Usually, the features are rescaled to be in the range from 0 to 1 or from -1 to 1. This can be done using formula 5.9.

$$I_N = (I - I_{min})[\frac{N_{max} - N_{min}}{I_{max} - I_{min}}] + N_{min} \tag{5.9}$$

where,

I: the non-normalized input value for the training process.

I_N: the normalized input value.

I_{min}: the minimum value of the input vector.

I_{max}: the maximum value for the input vector.

After the training process is done and the output predicting results are observed, the normalized values must denormalized values as shown in equation 5.10

$$O = (O_N - O_{min})[\frac{O_{max} - O_{min}}{N_{max} - N_{min}}] + O_{min} \tag{5.10}$$

where,

O: the non-normalized output value for the training process.

O_N: the normalized input value.

O_{min}: the minimum value of the output vector.

O_{max}: the maximum value for the output vector.

In the neural network there is one weight for each input connection. In general, good initializing weights for the networks is a small random number. In our case, will we use random numbers which are uniformly distributed in the range of $(-\frac{2.4}{F_i}, +\frac{2.4}{F_i})$, where F_i represent the number of inputs to neuron i in the network for the initial weights w_1,w_2,..., w_n and threshold t [35].

2. Forward Propagate

the output of a neural network can be computed by propagating the input signal through each layer until reaching the output layer to get the outputs values.

This is called a forward-propagation which is a technique need for generating the predictions during training that usually needs to be modified and it also will need after the network is trained to make predictions based on new data.

In similar way to the perceptron, at first, computes the weighted sum of the inputs as before:

$$X = \sum_{i}^{n} W_i x_i - t$$

such that n is the number of inputs and t is the threshold applied to the neuron.

we will used the hyperbolic tangent activation function given by the following:

$$Y_{tanh} = tanh(X) \tag{5.11}$$

or

$$Y_{tanh} = \frac{2}{1 + e^{-2X}} - 1$$

It is possible to compute the derivative of this function and it is guaranteed that the output of the neuron is bounded between -1 and 1.

$$Y'_{tanh} = 1 - tanh^2(x) \tag{5.12}$$

or

$$Y'_{tanh} = \frac{4e^{-2x}}{(e^{-2x} + 1)^2}$$

In a multilayer neural network, the output neurons have different inputs from the inputs for the neurons in the hidden layer. The output y_j of the neurons in the hidden layer is considered as the inputs for the output layer instead of initial input x_i.

At first, compute the actual outputs for the neurons in the hidden layer:

$$y_j(p) = tanh[\sum_{i=1}^{n} x_i(p) \times W_{ij}(p) - t_j];$$

where,

n: is the number of the inputs for neuron j in the hidden layer, and tanh is the hyperbolic tangent activation function.

Then, Calculate the actual outputs for the output neurons,

$$y_k(p) = tanh[\sum_{j=1}^{m} x_{jk}(p) \times W_{jk}(p) - t_k];$$

where m denoted the number of inputs for the output neuron k.

Define the error signal at the output neuron k at iteration p with the formula 5.13

$$e_k(p) = y_{d,k} - y_k(p) \tag{5.13}$$

$y_{d,k}(p)$: the desired output of the neuron k at iteration p.

3. Back Propagate Error

Error is the difference between the desired and the actual outputs for the network. These errors are propagated backwards through the network from the output layer to the hidden layer and updating weights as they go.

The error gradient for neuron k in the output layer is:

$$\delta_k(p) = \frac{\partial y_k(p)}{\partial X_k(p)} \times e_k(p), \tag{5.14}$$

such that $y_k(p)$ represent the output of the neuron k at itration p, and $X_k(p)$ denote the weighted sum of the input to neuron k at the same iteration.

The hyperbolic tangent activation function shown in equation 5.15:

$$\delta_k(p) = \frac{\partial(\frac{2}{1+e^{-2X}} - 1)}{\partial X_k(p)} \times e_k(p),$$

So,

$$\delta_k(p) = (1 - y_k^2(p)) \times e_k(p) \tag{5.15}$$

such that,

$$\delta_k(p) = \frac{2}{1 + e^{-2X}} - 1$$

In the hidden layer, a little more complicated than output layer, The error signal for the hidden neurons is calculated as weighted error for each output neuron, as illustrated in the following equation 5.16

$$\delta_j(p) = (1 - y_j^2(p)) \times \sum_{k=1}^{l} \delta_k(p) W_{jk} \tag{5.16}$$

where l represents the number of neurons in output layer.

$$Y_j(p) = \frac{2}{1 + e^{-2X_j(p)}} - 1 \ ;$$

$$X_j(p) = \sum_{i=1}^{n} x_i(p) \times W_{ij}(p) - t_j;$$

such that n represent the number of the neurons in the input layer.
A more detailed information is introduced in section 5.5.1 about Gradient descent algorithm.

4. Train Network

This part is described in two steps:

a. Update Weights.

b. Train Network.

4(a). Updating the Weights.

Once the errors are calculated for all the neuron in the network through the back propagation method as mentioned before, these errors can be used for updating the weights.

For updating the weights for neuron k at the output layer which is provided with the desired output, the formula 5.24 is used,

$$W_{jk}(p+1) = W_{jk}(p) + \Delta W_{jk}(p) \tag{5.17}$$

$\Delta W_{jk}(p)$: the weight correction which can defined as follow

$$\Delta W_{jk}(p) = \alpha y_j(p)\delta_k(p) \tag{5.18}$$

such that α is the learning rate which is a hyperparameter that controls how much we are modifying the weights in the network to correct the error and $\delta_k(p)$ is the error gradient for the neuron k in the output layer.

The same steps for the updating weights for the neuron j in the hidden layer,

$$W_{ij}(p+1) = W_{ij}(p) + \Delta W_{ij}(p) \tag{5.19}$$

$\Delta W_{ij}(p)$: the weight correction which can defined as follow

$$\Delta W_{ij}(p) = \alpha x_i(p)\delta_j(p) \tag{5.20}$$

where $\delta_k(p)$ is the error gradient for the neuron j in the hidden layer.

4(b). Train Network.

The network is trained using many training algorithms, such that, gradient descent algorithm, Newton's method, Quasi-Newton method, Levenberg-Marquardt algorithm The training involves many iterations to explore a relationship between the data.

The training function implements the training for an initialized neural network with a given training dataset, learning rate (α), fixed number of epochs and an expected number of output values.

The sum squared error between the desired output and the actual network output is accumulated each epoch. This is very helpful for us to know about how much the network is learning and improving each epoch. Once a network is trained, it is used to make predictions.

5. Predict

Making predictions with a trained neural network is the goal for all process. As described previously about how to forward-propagate inputs to get the outputs, make a prediction to be the same. We just need to provide the network with new input data and the network dicrectly through the forward-propagate will make a prediction and give good

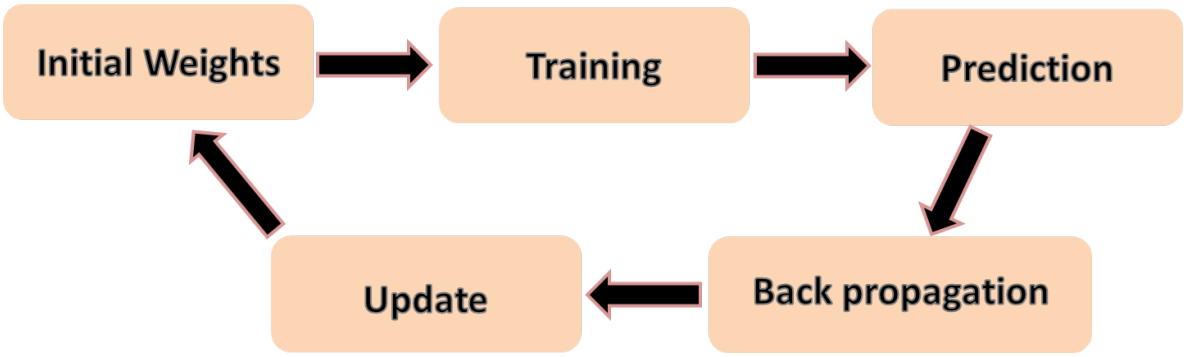

Figure 5.12: Train neural network

outputs.

5.5 Training algorithms

The learning problem in a neural network is formed mainly to search for the weight w at which the loss function takes its minimum value.
In general, the loss function is considered a non-linear function. Therefore, it is very hard to find a closed training algorithm for the minimum, so we should search for neural network parameters in consecutive steps. At each step, the loss decreases by modifying the parameters in the network.

Therefore, for training the neural network, the user has to provide initial random values for the weights. Then, a sequence of parameters will be generated, and the loss will change between each step, and this change is called the loss decrement.

The training process stops when a specific stopping criterion is satisfied [27]. This criterion is a specific value for the error or a specific number of epochs which are should be predefined.

Two of the most common training algorithms for the neural network are Gradient descent, Levenberg-Marquardt algorithm.

5.5.1 Gradient descent

Gradient descent is an iterative optimization algorithm used to find the minimum for the loss function. As mentioned previously, Backpropagation algorithm is used for updating the weights for the neural network when it is not able to make the correct predictions with the old weights. Gradient descent is one of the used training algorithms to update the network weights.

For starting the minimization process, initialize the weights for making the first output prediction,
The pattern error is the sum of squared errors of the output neurons:

$$E(x, w) = \frac{1}{2} \sum_{i=1}^{I} \sum_{k=1}^{K} e_k^2 \tag{5.21}$$

where, i is the index of patterns, from 1 to I, and I is the number of patterns, k is the index of outputs, from 1 to K, and K is the total number of outputs, x is the input vector, w is the weight vector, e is the training error at output k when applying pattern p and it is defined as

$$e_{p,k} = y_{(desired)p,k} - y_{(actual)p,k} \tag{5.22}$$

Where, $y_{(desired)p,k}$ is the desired output and $y_{(actual)p,k}$ is the actual output.

The gradient descent is a first-order algorithm, it uses the first-order derivative of the total error function to find the minimum error. Gradient g is defined as the first order derivative of total error function 5.23:

$$g = \frac{\partial E(x,w)}{\partial w} = \left(\frac{\partial E}{\partial w_1} \quad \frac{\partial E}{\partial w_2} \cdots \frac{\partial E}{\partial w_N}\right)^T \tag{5.23}$$

So, after measure the output error, if we got large value, we need to minimize it, so the question is How to minimize the error?

In neural network the loss function should be minimized. The used equations in section 5.4 are mentioned again,
The weight update rule as mentioned in equation 5.24 is,

$$W_{jk}(p+1) = W_{jk}(p) + \Delta W_{jk}(p) \tag{5.24}$$

$$\Delta W_{jk}(p) = -\alpha \frac{\partial E}{\partial W_{kj}} \tag{5.25}$$

Input of neuron k is:

$$X = \sum_{i}^{n} W_{jk} y_j \tag{5.26}$$

using the chain role,

$$\frac{\partial E}{\partial W_{kj}} = \frac{\partial E}{\partial X_{kj}} \times \frac{\partial X}{\partial W_{kj}} \tag{5.27}$$

So,

$$\Delta W_{jk}(p) = -\alpha \times \frac{\partial E}{\partial X_{kj}} \times \frac{\partial X}{\partial W_{kj}} \tag{5.28}$$

Let δ_k the error signal of neuron k:

$$\delta_k = -\frac{\partial E}{\partial X_k} \tag{5.29}$$

and

$$\frac{\partial X}{\partial W_{kj}} = y_j \tag{5.30}$$

So,

$$\Delta W_{kj} = \alpha \delta_k y_j \tag{5.31}$$

In order to compute the weight change ΔW_{kj} and the error signal δ_k of neuron k, there are two cases, depending on whether neuron is in the output layer or in the hidden layer.

In the case of that k is an output neuron then by using the chain rule we obtain that:

$$\delta_k = -\frac{\partial E}{\partial X_k} = \frac{\partial E}{\partial y_k} \times \frac{\partial y_k}{\partial X_k} \tag{5.32}$$

$$\frac{\partial E}{\partial y_k} = \frac{\partial E}{\partial e_k} \times \frac{\partial e_k}{\partial y_k} \tag{5.33}$$

$$= e_j(-1)\phi' \tag{5.34}$$

such that ϕ is the used activiation function.

becuase $e_k = y_{d,k} - y_k$ and $y_k = \phi(X_k)$

So, the weight w_{jk} from the output neuron k to the hidden neuron j is updated as:

$$\Delta W_{jk} = \alpha e \phi'(X_k) y_j \tag{5.35}$$

However, In the case of k=j is hidden neuron, also the chain rule is used the same as the output neuron.

$$\delta_j = -\frac{\partial E}{\partial X_j} = \frac{\partial E}{\partial y_j} \times \frac{\partial y_j}{\partial X_j} \tag{5.36}$$

$$\frac{\partial E}{\partial y_j} = \sum \frac{\partial E}{\partial x_k} \times \frac{\partial X_k}{\partial y_k}$$

(5.37)

Then,

$$\delta_j = -\sum_{k=1}^{K} \delta_k w_{jk} * \phi'(X_j) \tag{5.38}$$

So, the weight w_{ij} from the hidden neuron j to the input neuron i is

updated as:

$$\Delta W_{ij} = \alpha x_i \phi'(X_j) \times \sum_{k=1}^{K} \delta_k w_{jk} \tag{5.39}$$

Summary : The delta rule,

$$\delta_j = \begin{cases} \phi'(X_j)e, & \text{If j in output layer} \\ \phi'(X_j) \sum_{k=1}^{K} \delta_k w_{jk}, & \text{If j in hidden layer} \end{cases} \tag{5.40}$$

pseudo-code for Gradient descent

1. **Give a random Initial values for the weights and thresholds in the interval** $(\frac{-2.4}{F_i}, \frac{+2.4}{F_i})$**.**
2. **Calculate the Mean squared errors (MSE) over all the inputs.**
3. **Compute** $\Delta W = -\alpha \frac{\partial E}{\partial W}$**.**
4. **Update the network weights w using** $W = W + \Delta W$
5. **Recompute the Mean Squared Errors MSE using the updated weights.**

IF

MSE > stopping criteria

THEN

Go back to step 2

ELSE

Print the lowest value of the weights

END IF

5.5.2 Levenberg-Marquardt algorithm (LM)

Levenberg Marquardt [30] is an a standard procedure used for solving nonlinear least squares problems.

Levenberg Marquardt method is an iterative technique used to reduce the sum of squares errors between the function and the measured data points using succession steps for updating the values of the weights. It works with the gradient vector and the Jacobian matrix [32].

LM locates the minimum of the loss function which is the sum squared error, it finds only a local minimum, which is not necessarily the global minimum. It is considered the most efficient training algorithm for artificial neural network with median size.

The gradient component $g_1, g_2, \cdots, g_N$ is assumed to be functions of weights as equation 5.41

$$\begin{cases} g_1 = F_1(w_1, w_2 \cdots w_N) \\ g_2 = F_2(w_1, w_2 \cdots w_N) \\ : \\ g_N = F_N(w_1, w_2 \cdots w_N) \end{cases} \tag{5.41}$$

By taking First-order approximation Taylor series, we can write each

g_i(i=1,2,...,N) in equation 5.41 as:

$$\begin{cases} g_1 \approx g_1(w_0) + \frac{\partial g_1}{\partial w_1}(w_1 - w_0) + \frac{\partial g_1}{\partial w_2}(w_2 - w_0) + ... + \frac{\partial g_1}{\partial w_N}(w_N - w_0) \\ g_2 \approx g_2(w_0) + \frac{\partial g_2}{\partial w_1}(w_1 - w_0) + \frac{\partial g_2}{\partial w_2}(w_2 - w_0) + ... + \frac{\partial g_2}{\partial w_N}(w_N - w_0) \\ : \\ g_N \approx g(w_0) + \frac{\partial g_N}{\partial w_1}(w_1 - w_0) + \frac{\partial g_1}{\partial w_2}(w_2 - w_0) + ... + \frac{\partial g_N}{\partial w_N}(w_N - w_0) \end{cases} \tag{5.42}$$

By the definition of the gradient vector g in equation 5.23, we can define:

$$\frac{\partial g_i}{\partial w_j} = \frac{\partial(\frac{\partial E}{\partial w_i})}{\partial w_j} = \frac{\partial^2 E}{\partial w_i \partial w_j} \tag{5.43}$$

Now, insert equation 5.43 to equation 5.42

$$\begin{cases} g_1 \approx g_{1,0} + \frac{\partial^2 E}{\partial w_1^2}(w_1 - w_0) + \frac{\partial^2 E}{\partial w_1 \partial w_2}(w_2 - w_0) + ... + \frac{\partial^2 E}{\partial w_1 \partial w_N}(w_N - w_0) \\ g_2 \approx g_{2,0} + \frac{\partial^2 E}{\partial w_2 \partial w_1}(w_1 - w_0) + \frac{\partial^2 E}{\partial w_2^2}(w_2 - w_0) + ... + \frac{\partial^2 E}{\partial w_2 \partial w_N}(w_N - w_0) \\ : \\ g_N \approx g_{N,0} + \frac{\partial^2 E}{\partial w_N \partial w_1}(w_1 - w_0) + \frac{\partial^2 E}{\partial w_N \partial w_2}(w_2 - w_0) + ... + \frac{\partial^2 E}{\partial w_N^2}(w_N - w_0) \end{cases} \tag{5.44}$$

Therefore, the second order derivatives of the total error function must be computed for each component of the gradient vector. In order to minimize the total error function E, the gradient vector component should

be equal to zero.

$$\begin{cases} 0 \approx g_{1,0} + \frac{\partial^2 E}{\partial w_1^2}\Delta w_1 + \frac{\partial^2 E}{\partial w_1 \partial w_2}\Delta w_2 + ... + \frac{\partial^2 E}{\partial w_1 \partial w_N}\Delta w_N \\ 0 \approx g_{2,0} + \frac{\partial^2 E}{\partial w_2 \partial w_1}\Delta w_N + \frac{\partial^2 E}{\partial w_2^2}\Delta w_2 + ... + \frac{\partial^2 E}{\partial w_2 \partial w_N}\Delta w_N \\ : \\ 0 \approx g_{N,0} + \frac{\partial^2 E}{\partial w_N \partial w_1}\Delta w_1 + \frac{\partial^2 E}{\partial w_N \partial w_2}\Delta w_2 + ... + \frac{\partial^2 E}{\partial w_N^2}\Delta w_N \end{cases} \tag{5.45}$$

Through combining equation 5.23 with 5.45 :

$$\begin{cases} -\frac{\partial E}{\partial w_1} = -g_{1,0} \approx +\frac{\partial^2 E}{\partial w_1^2}\Delta w_1 + \frac{\partial^2 E}{\partial w_1 \partial w_2}\Delta w_2 + ... + \frac{\partial^2 E}{\partial w_1 \partial w_N}\Delta w_N \\ -\frac{\partial E}{\partial w_1} = -g_{2,0} \approx +\frac{\partial^2 E}{\partial w_2 \partial w_1}\Delta w_N + \frac{\partial^2 E}{\partial w_2^2}\Delta w_2 + ... + \frac{\partial^2 E}{\partial w_2 \partial w_N}\Delta w_N \\ : \\ -\frac{\partial E}{\partial w_1} = g_{N,0} \approx +\frac{\partial^2 E}{\partial w_N \partial w_1}\Delta w_1 + \frac{\partial^2 E}{\partial w_N \partial w_2}\Delta w_2 + ... + \frac{\partial^2 E}{\partial w_N^2}\Delta w_N \end{cases} \tag{5.46}$$

So, all Δw_i can be computed since we have N equations for N parameters. Through the solution the weights can be updated iteratively. we can write equation 5.47 in matrix form

$$\begin{bmatrix} -g_1 \\ -g_2 \\ : \\ -g_N \end{bmatrix} = \begin{bmatrix} -\frac{\partial E}{\partial w_1} \\ -\frac{\partial E}{\partial w_2} \\ : \\ -\frac{\partial E}{\partial w_N} \end{bmatrix} = \begin{bmatrix} \frac{\partial^2 E}{\partial w_1^2} & \frac{\partial^2 E}{\partial w_1 \partial w_2} & \cdots & \frac{\partial^2 E}{\partial w_1 \partial w_N} \\ \frac{\partial^2 E}{\partial w_2 \partial w_1} & \frac{\partial^2 E}{\partial w_2^2} & \cdots & \frac{\partial^2 E}{\partial w_2 \partial w_N} \\ : & : & : & : \\ \frac{\partial^2 E}{\partial w_N \partial w_1} & \frac{\partial^2 E}{\partial w_N \partial w_2} & \cdots & \frac{\partial^2 E}{\partial w_N^2} \end{bmatrix} + \begin{bmatrix} \Delta w_1 \\ \Delta w_2 \\ : \\ \Delta w_N \end{bmatrix} \tag{5.47}$$

Such that the square matrix is called Hessian matrix H, which is a ma-

trix of the second derivatives for the loss function with respect of all combinations of the weights.

$$H = \begin{bmatrix} \frac{\partial^2 E}{\partial w_1^2} & \frac{\partial^2 E}{\partial w_1 \partial w_2} & \cdots & \frac{\partial^2 E}{\partial w_1 \partial w_N} \\ \frac{\partial^2 E}{\partial w_2 \partial w_1} & \frac{\partial^2 E}{\partial w_2^2} & \cdots & \frac{\partial^2 E}{\partial w_2 \partial w_N} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial^2 E}{\partial w_N \partial w_1} & \frac{\partial^2 E}{\partial w_N \partial w_2} & \cdots & \frac{\partial^2 E}{\partial w_N^2} \end{bmatrix} \tag{5.48}$$

By equations 5.23 and 5.48, equation 5.47 can be written as,

$$-g = H\Delta w \tag{5.49}$$

So,

$$\Delta w = -H^{-1}g \tag{5.50}$$

Since Hessian matrix is second order derivatives, it can estimate the curvature of the loss function and give a good evaluation on the change of the gradient vector. From this we can obtain the updated role for the Newton's method as given in equation 5.51

$$\Delta w_{p+1} = w_p - H_p^{-1} g_p \tag{5.51}$$

Nowdays, most of the architectures used billions of parameters, and compute a billion of second derivatives is very complicated process. In order to simplify the calculating process, Jacobian matrix [54] J was intro-

duced, which is defined as :

$$J = \begin{vmatrix} \frac{\partial e_{1,1}}{\partial W_1} & \frac{\partial e_{1,1}}{\partial W_2} & \cdots & \frac{\partial e_{1,1}}{\partial W_N} \\ \frac{\partial e_{1,2}}{\partial W_1} & \frac{\partial e_{1,2}}{\partial W_2} & \cdots & \frac{\partial e_{1,2}}{\partial W_N} \\ \vdots & \vdots & \ldots & \vdots \\ \frac{\partial e_{1,M}}{\partial W_1} & \frac{\partial e_{1,M}}{\partial W_2} & \cdots & \frac{\partial e_{1,M}}{\partial W_N} \\ \vdots & \vdots & \ldots & \vdots \\ \frac{\partial e_{I,1}}{\partial W_1} & \frac{\partial e_{I,1}}{\partial W_2} & \cdots & \frac{\partial e_{I,1}}{\partial W_N} \\ \frac{\partial e_{I,2}}{\partial W_1} & \frac{\partial e_{I,2}}{\partial W_2} & \cdots & \frac{\partial e_{I,2}}{\partial W_N} \\ \vdots & \vdots & \ldots & \vdots \\ \frac{\partial e_{I,M}}{\partial W_1} & \frac{\partial e_{I,M}}{\partial W_2} & \cdots & \frac{\partial e_{I,M}}{\partial W_N} \end{vmatrix}$$

By equation 5.21 and 5.23, The element of the gradient vector can be computed as:

$$g_i = \frac{\partial E}{\partial w_i} = \frac{\partial(\frac{1}{2}\sum_{l=1}^{L}\sum_{m=1}^{M} e_{l,m}^2)}{\partial w_i} = \sum_{l=1}^{L}\sum_{m=1}^{M}(\frac{\partial e_{i,m}}{\partial w_i} e_{i,m}) \tag{5.52}$$

So the relation between the jacobian matrix and gradient vector is:

$$g = J^T e \tag{5.53}$$

where e is the error vector defend as:

$$e = \begin{bmatrix} e_{1,1} \\ e_{1,2} \\ ... \\ e_{1,M} \\ ... \\ e_{l,1} \\ e_{l,2} \\ ... \\ e_{l,M} \end{bmatrix} \tag{5.54}$$

The elements of the Hessian matrix can be calculated as:

$$h_{i,j} = \frac{\partial E}{\partial w_i} = \frac{\partial^2(\frac{1}{2}\sum_{l=1}^{L}\sum_{m=1}^{M} e_{l,m}^2)}{\partial w_i \partial w_j} = \sum_{l=1}^{L}\sum_{m=1}^{M} \frac{\partial e_{l,m}}{\partial w_i}\frac{e_{l,m}}{\partial w_j} + S_{i,j} \tag{5.55}$$

where

$$S_{i,j} = \sum_{l=1}^{L}\sum_{m=1}^{M} \frac{\partial^2 e_{l,m}}{\partial w_i \partial w_j} e_{l,m} \tag{5.56}$$

Since $e_{l,m}$ is very small approximately zero, the summation term can be ignored [56]. Then, The relation between the Jacobian matrix and Hessian matrix is:

$$H \approx J^T J \tag{5.57}$$

So by combining the equations 5.51, 5.53 and 5.57 the updated rule will

be:

$$w_{p+1} = w_p - (J^T J)^{-1} J^T e \tag{5.58}$$

One limitation could be in this formula, is that the approximated Hessian matrix may not be invertible, so, in order to ensure that the approximated Hessian matrix is invertible, Levenberg–Marquardt algorithm use a modified Hessian matrix:

$$H \approx J^T J + \mu I \tag{5.59}$$

where μ: is always positive, called combination coefficient, I: is the identity matrix.

The update rule for the Levenberg–Marquardt algorithm [39] can be defined as:

$$w_{p+1} = w_p + (J^T J + \mu I)^{-1} J^T e \tag{5.60}$$

If the combination coefficient μ result in increased the MSE, it is multiplied by some factor β. When a step reduces the MSE, μ divided by β.

Since Levenberg-Marquardt uses the Jacobian matrix, it needs big memory requirements. On the other hand, it is the fastest algorithm.
If the neural networks have a huge amount of parameters then the suitable choice for the training is the gradient descent training algorithm to save memory. If the neural network has just a few thousands of parameters, the Levenberg-Marquardt algorithm might be the best choice for

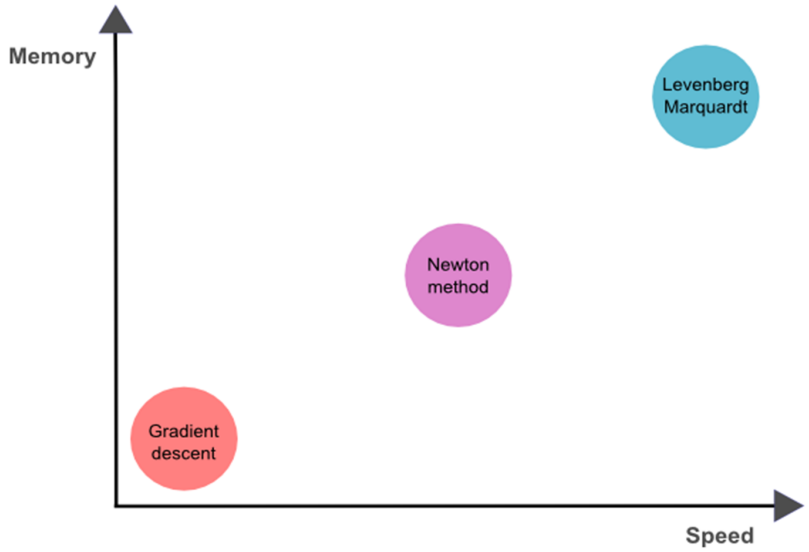

Figure 5.13: Training algorithms

training. In the rest of cases, Newton method will work better. figure 5.13 depicts a good comparison between the training algorithms in terms of the computational speed and the memory requirements.

pseudo-code for LM

1. **Give a random Initial values for the weights and thresholds in the interval** $(\frac{-2.4}{F_i}, \frac{+2.4}{F_i})$ **and take an appropriate value of the parameter** μ**.**

2. **calculate the sum of the squared errors (SSE) for all the inputs.**

3. **Solve** $\Delta W = (J^T J + \mu I)^{-}1 \times J^T e$ **to obtain the weight correction** Δw**.**

4. **update the network weights w using** $W = W + \Delta W$

5. **Recompute the sum of squared errors E Using the updated weights.**

IF

SSE decreased

THEN

μ= 10 μ

Go back to step 2

ELSE

μ=$\frac{\mu}{10}$

go back to step 4

END IF

Chapter Six

Simulation and Results

6.1 Weighted centroid algorithm based on RSSI

In our experiment evaluation of the performance of weighted centroid localization algorithm (WCL) was implemented using the MATLAB R2018a simulator. In this algorithm, we have set up the following conditions for the simulation environment:

1. The anchor nodes are regularly distributed in a squared area of 100 x 100 m^2 and the number of these anchors is set to be 100.

2. The unknown sensor nodes are deployed randomly in the experiment area.

3. The communication range of sensor node (CR) is set to 20 m.

4. the estimated coordinates computed through WCL formula,

$$\begin{cases} X_{est} = \frac{w_1 * x_1 + w_2 * x_2 + ... + w_n * x_n}{w_1 + w_2 + ... + w_n}, \\ Y_{est} = \frac{w_1 * y_1 + w_2 * y_2 + ... + w_n * y_n}{w_1 + w_2 + ... + w_n}, \\ wi = \frac{RSSI_i}{RSSI_1 + RSSI_2 + ... + RSSI_n}. \end{cases} \tag{6.1}$$

5. Log normal Shadowing distribution model used to compute the RSSI values.

$$PL(d) = PL(d_0) + 10nlog\frac{d}{d0} + X_\sigma \tag{6.2}$$

Where, $PL(d)$ is the path loss after distance d, n is the path loss exponent, X_σ is a zero mean Gaussian distributed random variable, d_0 is reference distance, PL(d_0) is a known reference power value at a reference distance d_0 from the transmitter.

The network was generated using the parametres given in table 6.1

Table 6.1: Parameters for WCL algorithm

Parameters	**Values**
Number of anchor nodes	100
Number of unknown nodes	50
Communication range	20 m
Deployment area	100 × 100
Pt	10 watt
n	2
d_0	1

In figure 6.1, the black dots represent the positions of the anchor nodes, stars represent the actual locations of the unknown nodes (unlocalized nodes), and circles represent the estimated locations of the unknown node by the weighted centroid localization algorithm. The displacement of the unknown nodes from their actual positions to the estimated position represented by the red lines.

In figure 6.2, the estimated error represent for 50 unknown nodes such that the errors are defied as the difference between the real and the estimated locations, node 7 represents the maximum error as it is displayed maximum from its real location to the estimated location. Node 12 has

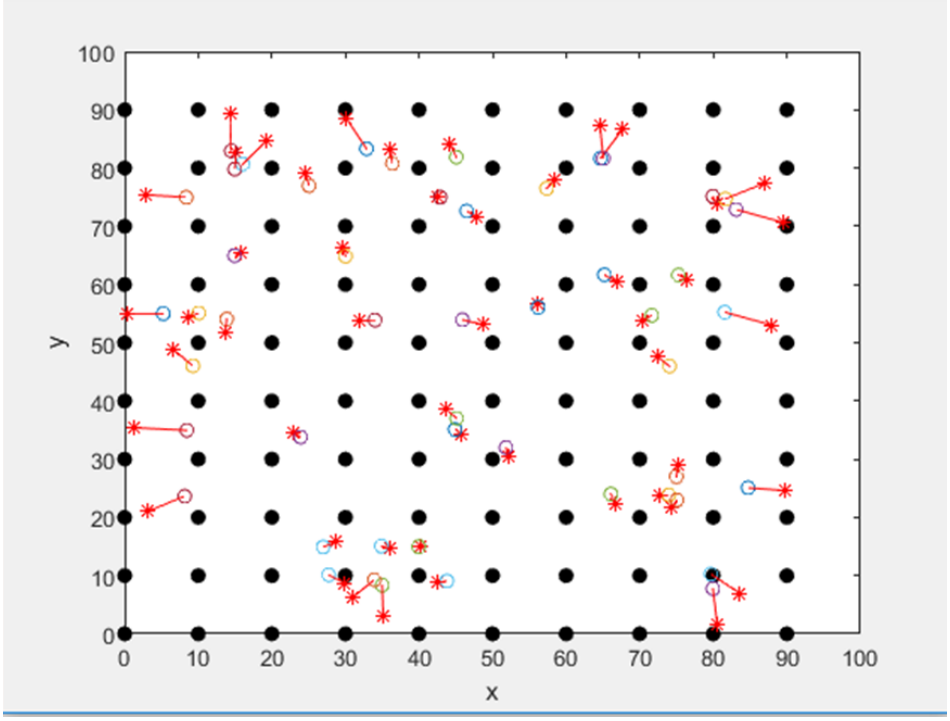

Figure 6.1: Location estimation using WCL algoritm for 50 nodes

the minimum error, the avarege error for all 50 unknown nodes is equal to 0.1554 m.

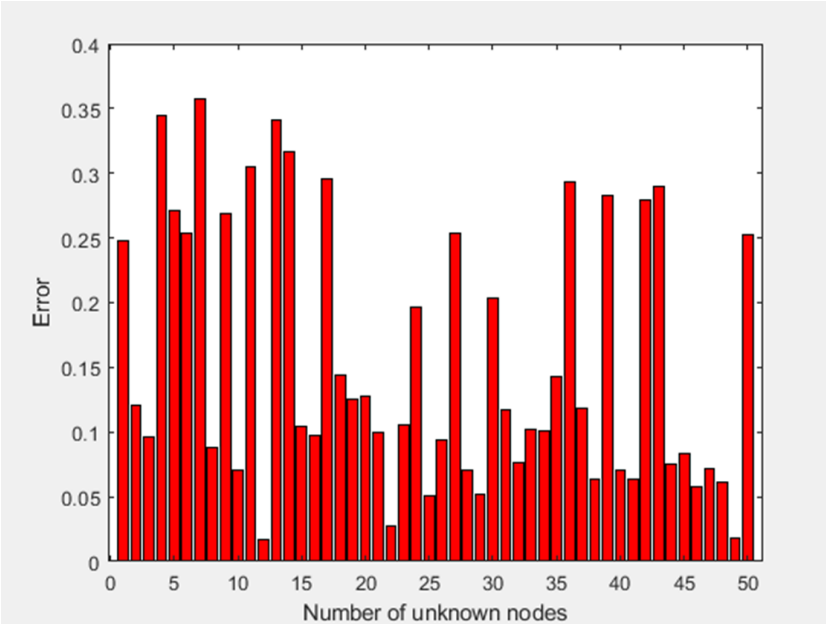

Figure 6.2: Error estimation for 50 nodes using WCL

Localization Error when number of Anchor node is varied:

In this simulation, we analyse the effect of number of anchor nodes on the localization error.

We take different numbers of the anchor nodes described in table 6.2

Table 6.2: Varied number of anchors vs the value of the Localization error obtaind by WCL

Number of Anchors	Localization error
100	0.1449 m
144	0.1268 m
196	0.1088 m

Table 6.2 shows the effect of the increased number of anchors when we use communication range 20m, and 100 unknown nodes, we notice that the error decreases as the number of anchors increases. Anchor nodes used to help in the unknown node localization process, the localization improves as the number of the anchors increase, but on the other hand the cost will increase, so we should try to locate the unknown nodes with minimum number of anchor nodes. Figures 6.3, 6.4, 6.5 shows the error of 100 unknown nodes distributed with 100, 144, 196 anchor nodes, respectively.

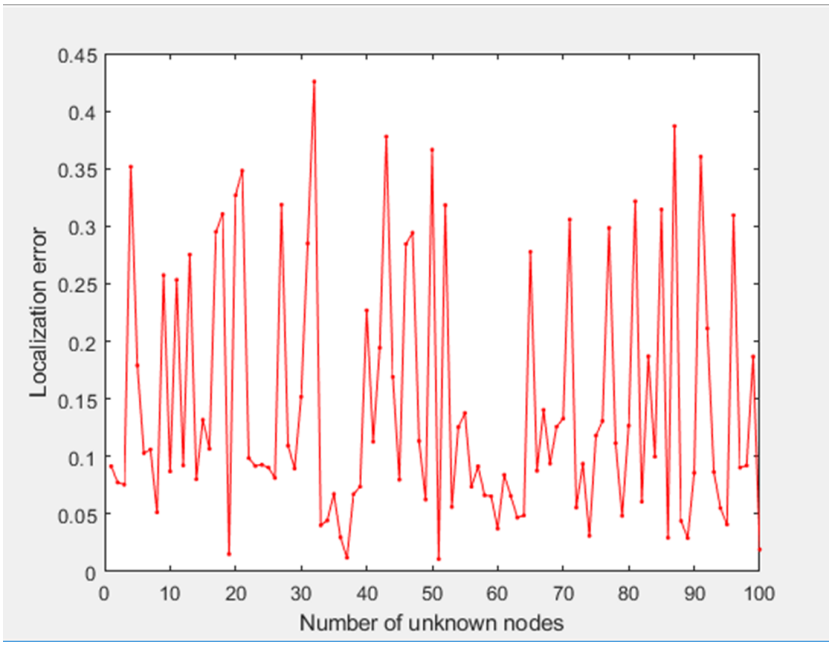

Figure 6.3: Error estimation for 100 unknown nodes with 100 anchors

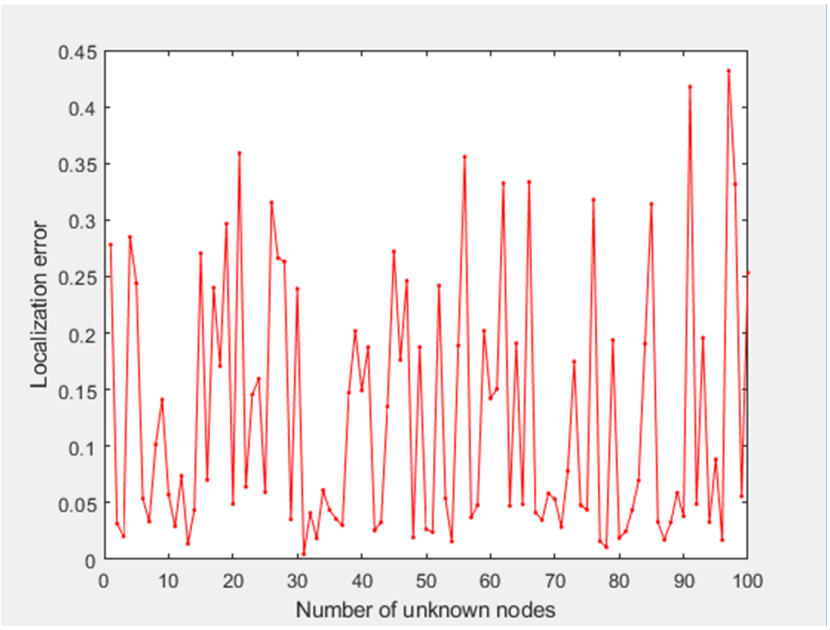

Figure 6.4: Error estimation for 100 unknown nodes with 144 anchors

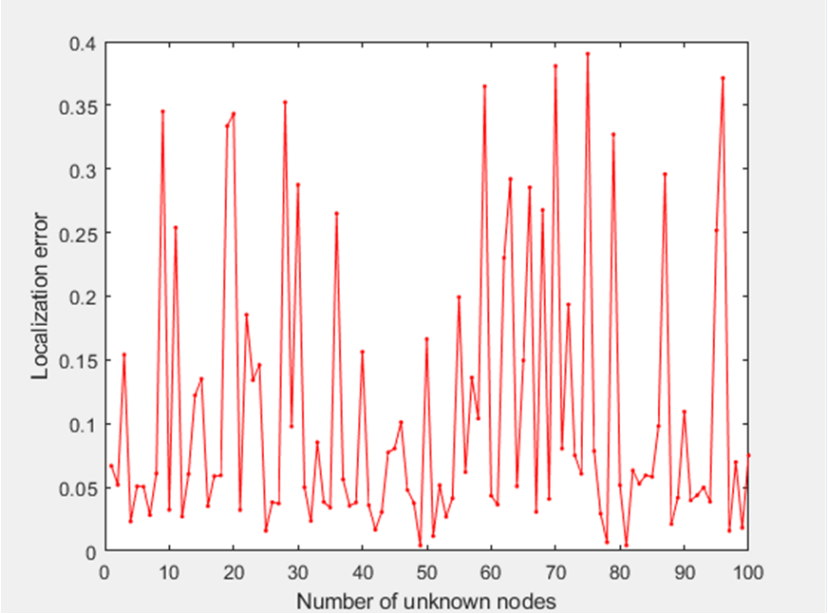

Figure 6.5: Error estimation for 100 unknown nodes with 196 anchors

6.2 Feedforward Neural Network

After we study the performance of the weighted centroid algorithm, we will introduce the use of the artificial neural network, especially the feed-forward neural network in sensor node localization and we will compare it's results with WCL algorithm and DNN.

6.2.1 Data Collection

Data collection for the FFNN is a critical issue. Therefore our dataset consists of, the input data which is the random locations for the unknown nodes, and the desired output is the estimated locations obtained from the WCL algorithm. The RSSI measurements used in the WCL which was explained in chapter 4 is considered the most important factor used to apply the WCL algorithm for helping us to collect these required data.

Table 6.3 shows a part of the training data used to train our FFNN model.

Table 6.3: Samples of the training data.

X	**Y**	**Anchors**	**Width**	**Length**	$\boldsymbol{X_{est}}$	$\boldsymbol{Y_{est}}$
63.2421	45.9368	100	100	100	63.8928	46.1567
17.1165	44.4277	100	100	100	16.0529	43.7939
89.8132	44.511	100	100	100	84.7544	45.0956
16.7709	52.9684	225	150	150	16.0913	53.9389
124.7502	35.2376	225	150	150	125.0299	34.9931
47.2888	96.1498	225	150	150	46.0366	96.1385
68.5729	127.8781	400	200	200	67.3496	126.5028
35.1601	34.7811	400	200	200	34.9888	35.0222
28.3134	129.9582	400	200	200	27.8921	130.0209
96.1611	62.3364	625	250	250	96.1209	64.027
190.5943	4.1689	625	250	250	192.1931	9.3643
154.826	91.0939	625	250	250	155.0135	93.025
181.6974	122.3443	900	300	300	184.0838	124.0403
178.9024	131.9046	900	300	300	178.0624	131.8682
171.3831	150.7318	900	300	300	171.9142	151.989

Using Matlab to handle the data in table 6.3 and simulating the FFNN. The First step is normalizing the data, using the method of Min-Max Normalization [43] using formula 5.9 for the input data and 5.10 formula for the outputs.

6.2.2 Artificial Neural Network Structures

1. Feed Forward (single layer) Neural Networks

   Using MATLAB R2018a, we build FFNN with different structures to obtain which one gives the best results for the node localization,

the inputs and the desired outputs were fed to the network. A series of trails were performed, The first trail is a network with 10 hidden neuron as shown in figure 6.6.

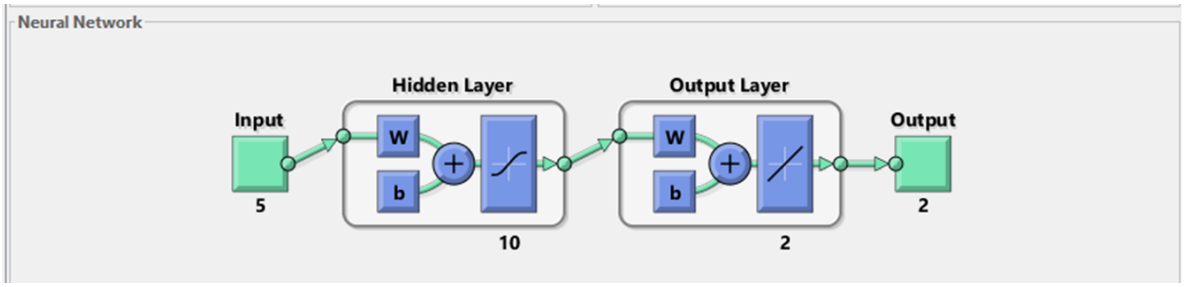

Figure 6.6: The proposed FFNN 10 hidden neurons.

The total number of input data patterns were separated into three samples, training set, 75% of the data were used to train the network. Validation set, 10% of the data were used to validate how well the network generalised, and the rest 15% of the data was used as testing set which provide a test of the network for data that the network has never seen.

The next trails for the network structure is a network with 15, 20, 25 and 30 hidden neurons.

2. Deep feedforward (multilayers) Neural Networks

Another trail to estimate the sensor node locations in WSNs is to use the deep feedforward (multilayer) neural network, the tried structures is a networks with two hidden layers, as detailed in table 6.4.

Table 6.4: Summary of different deep networks structure.

Neurons in hidden layers	Training algorithm	MSE error	R
10, 10	LM	2.0539	0.99979
20, 20	LM	1.7592	0.99984

As we notice that the network with 20 neurons is better than the network with 10 neurons. Figure 6.7. shows the DNN structure.

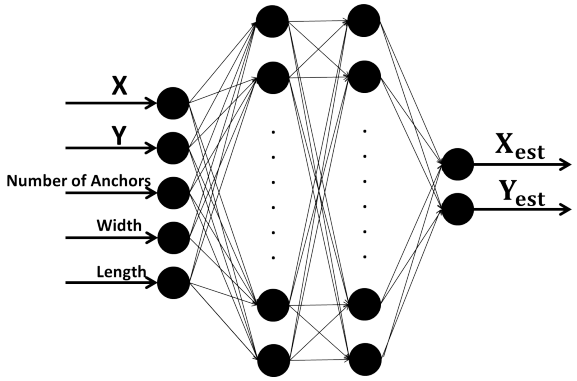

Figure 6.7: Structure of the proposed DNN

6.2.3 Evaluation of the proposed neural network models

To select the best neural network, MSE and the correlation cofficient is the criterion to perform the selection.

MSE values for the training, validation and testing samples in the first structure with 10 hidden neuron were illustrated in figure 6.8
As we noticed that the values of MSE for the three sets is considered large.

In addition, the correlation coefficient R was taken into consideration

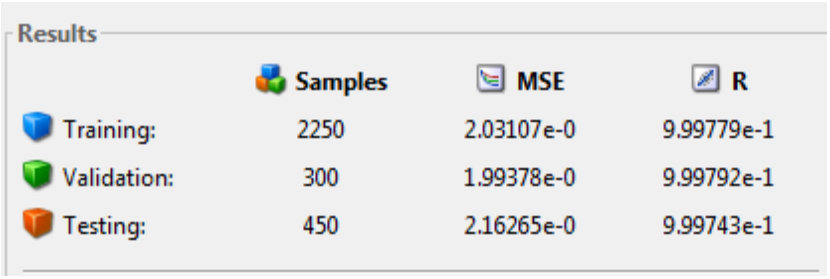

Figure 6.8: MSE for the FFNN data with 10 hidden neuron

to decide if the ANN is acceptable or not. The correlation coefficient R used to give an idea about the correlation between the actual output (predicted output) data and the desired output and to explain how good a fit between the predicted value and the desired value.

The formula used to compute the correlation coefficient [45] is given by equation 6.3,

$$R = \frac{\sum_{i=1}^{n}(t-\bar{t})\cdot(a-\bar{a})}{\sqrt{\sum_{i=1}^{n}(t-\bar{t})^2}\sqrt{\sum_{i=1}^{n}(a-\bar{a})^2}} \tag{6.3}$$

Where t is the target value, a is the actual output of the network and n is the number of the sample data, $\bar{t}$ is the mean of the target data, $\bar{a}$ is the mean of the actual data.

If R is zero then the relation between the predicted value and the target value is irrelevant and when R is equal to one, there is a perfect fit.

Figure 6.9 illustrate Correlation Coefficient for network performance.

And as we noticed that the value of R is near 1 so it is good.

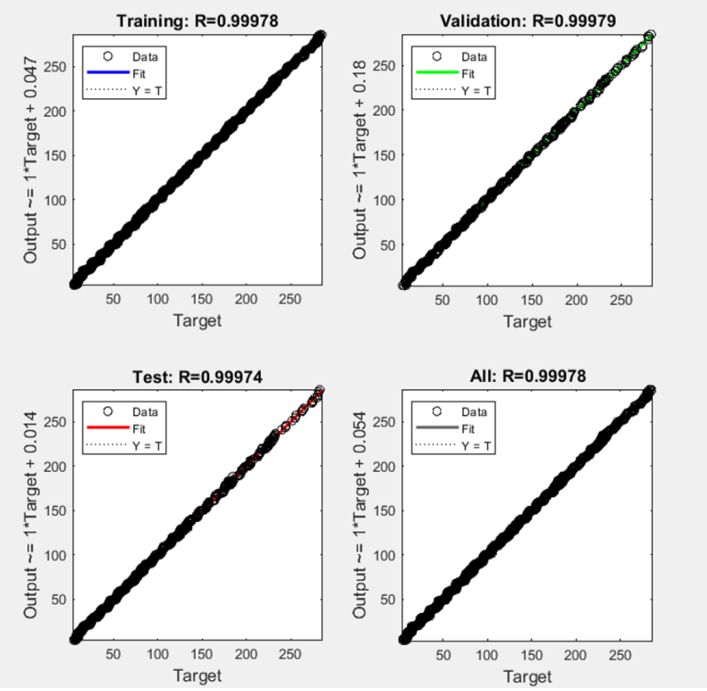

Figure 6.9: Correlation Coefficient R for network performance (network with 10 hidden neurons)

The second structure which is a network with 15 hidden neuron has the values of MSE for each set of the training, validation and testing and the values of R as shown in figure 6.10 and figure 6.11,

Results

	Samples	MSE	R
Training:	2250	1.96859e-0	9.99783e-1
Validation:	300	1.97620e-0	9.99769e-1
Testing:	450	2.12699e-0	9.99785e-1

Figure 6.10: MSE for the FFNN data with 15 hidden neuron

As a result, we observe that the MSE is decreased and the value of R is increase which means that this structure of the network is better than first structure with 10 hidden neuron.

The network with 20 hidden neurons has the results for MSE and R

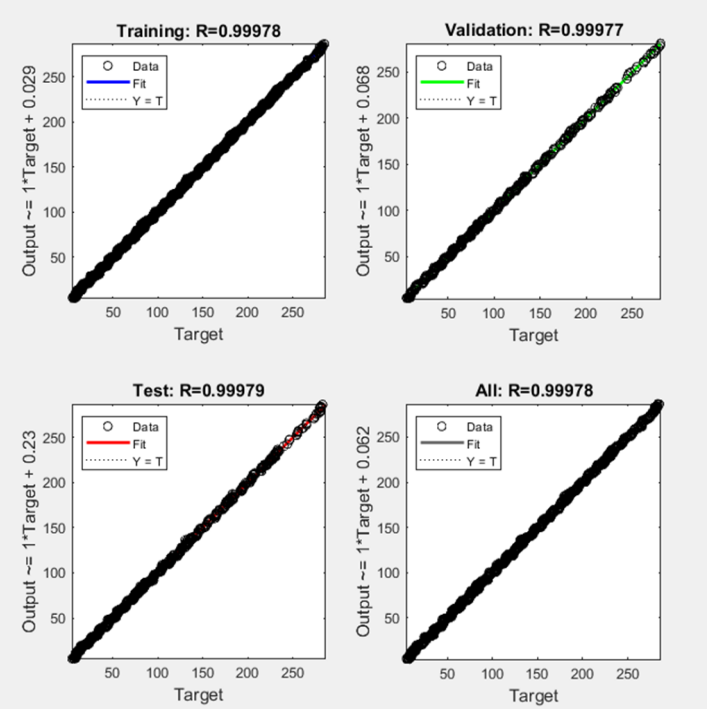

Figure 6.11: Correlation Coefficient R for network performance (network with 15 hidden neurons)

as illustrated in figure 6.12 and 6.13, respectively.

	Samples	MSE	R
Training:	2250	1.44416e-0	9.99841e-1
Validation:	300	1.54475e-0	9.99834e-1
Testing:	450	1.42572e-0	9.99844e-1

Figure 6.12: MSE for the FFNN data with 20 hidden neuron

Table 6.5 Summarizes the results of different networks. performance.

As a result, from table 6.5 we observe that the best network structure is a FFNN with 20 hidden neuron since the MSE has the lowest value comparing to the other FFNN structures and with the different DNN structurs.

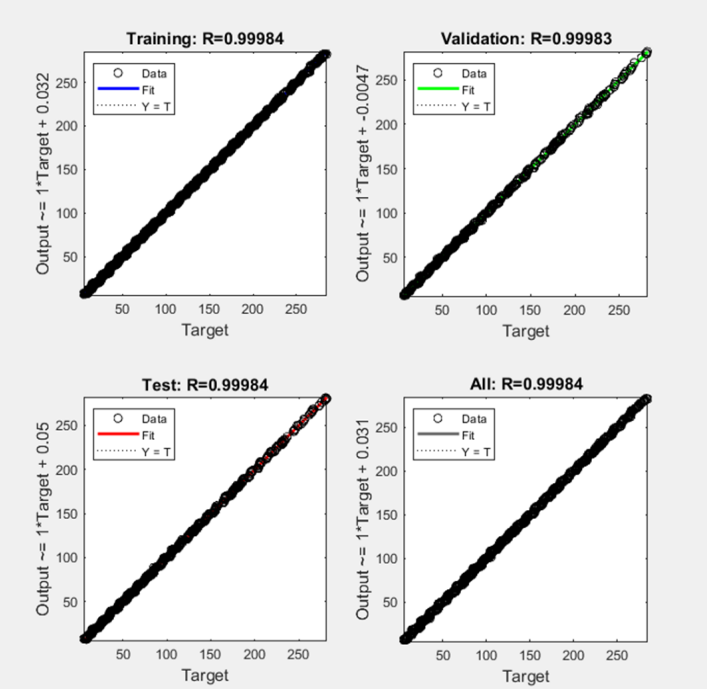

Figure 6.13: Correlation Coefficient R for network performance (network with 20 hidden neurons)

Table 6.5: Summary of different networks evaluated to yield the criteria of network performance.

hidden neurons	Training algorithm	Training error	R
10	LM	2.03107	0.99779
12	LM	2.10060	0.999771
15	LM	1.96859	0.999783
20	LM	1.44416	0.999841
25	LM	1.44641	0.999842
30	LM	1.53435	0.999833
20	GD	7.85491	9.99144
10, 10	LM	2.0539	0.99979
20, 20	LM	1.7592	0.99984

Now another important thing that we have the ability to choose it, is the training algorithm, so we apply Levenberg-Marquardt training algorithm and gradient descent.

The results of the MSE and R for the network with 20 hidden neuron

training with gradient descent algorithm is shown in figure 6.14

Results

	Samples	MSE	R
Training:	2100	7.85491e-0	9.99144e-1
Validation:	450	8.18078e-0	9.99091e-1
Testing:	450	8.08337e-0	9.99120e-1

Figure 6.14: MSE for the FFNN data using Gradient descent training algorithm

We notice that the MSE using Gradient descent algorithm is grater than the network that used the LM training algorithm shown previously in figure 6.12.

So we obtain that the best structure for the network is a FFNN with 20 hidden neuron applied the LM algorithm. This network performance is shown in figure 6.15.

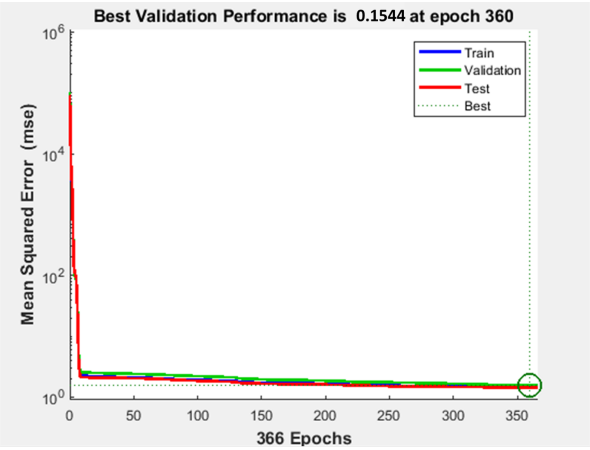

Figure 6.15: Mean square error (MSE) against epochs

the value of the MSE is 0.1544 with 360 epoch, this is consider a good performance.

6.3 Results and discussion

In this study, the best structure for the neural network was decided to be a FFNN with five input neurons, one hidden layer with 20 neurons and 2 outputs neurons since it has the minmum error and the best performance as we explained in the previous section.

For evaluating the network and find the optimum solution, MSE value and R-value were used. Smallest MSE value with the largest R-value reflects the best network performance. A regression analysis between the predicted and the desired outputs was carried out to investigate the network performance in a detailed way and the correlation coefficient R was computed to reflect the relation between the target and predicted network outputs.

Levenberg Marquardt algorithm was selected to train the network since it gives better results than the other training algorithms. In addition, Hyperbolic Tangent function (tanh) activation function for the neuron in the hidden layer was chosen to be used.

Consequently, the network structure (5-20-2) is considered satisfactory since it is the only network from all the other trained network that is achieved the minimum value of MSE with the highest correlation between the predicted and the desired output values which implies that this model is succeeded to predict the best location for the sensor in WSNs.

After we get the best trained neural network (FFNN), figure 6.16

shows the proposed FFNN with 20 hidden neurons.

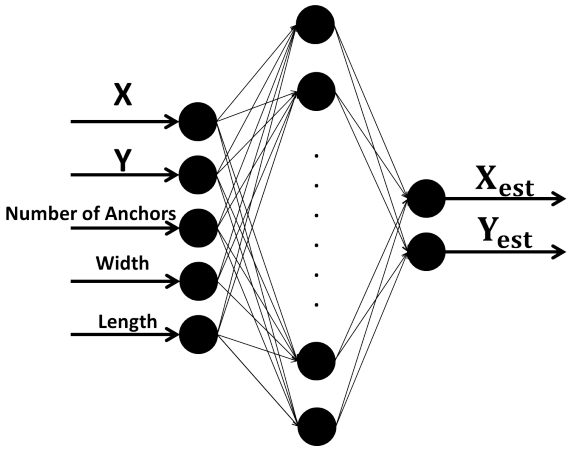

Figure 6.16: Structure of the proposed FFNN

We used this network to purpose of the prediction for the location of the unknown sensors nodes in WSNs, after we apply the weighted centroid localization algorithm based on RSSI to get a good location for the unknown sensor nodes with the help of the anchor node information. We propose to use the FFNN to predict a more accurate results than the locations obtained from the WCL.

A testing data set of 37 sensor node used to compare between the predicted locations observed from the WCL algorithm and FFNN.

37 unknown node were distributed randomly in a WSN with 100 anchor nodes which is uniformly distributed within the area 100 $\times$ 100 m^2 and the locations for the unknown nodes were determined using the WCL as shown in figure 6.17.

The red stars represents the actual locations for the sensors and the hollow squares represents the estimated location using the WCL algorithm, and the average location error is equal to 0.1002 m.

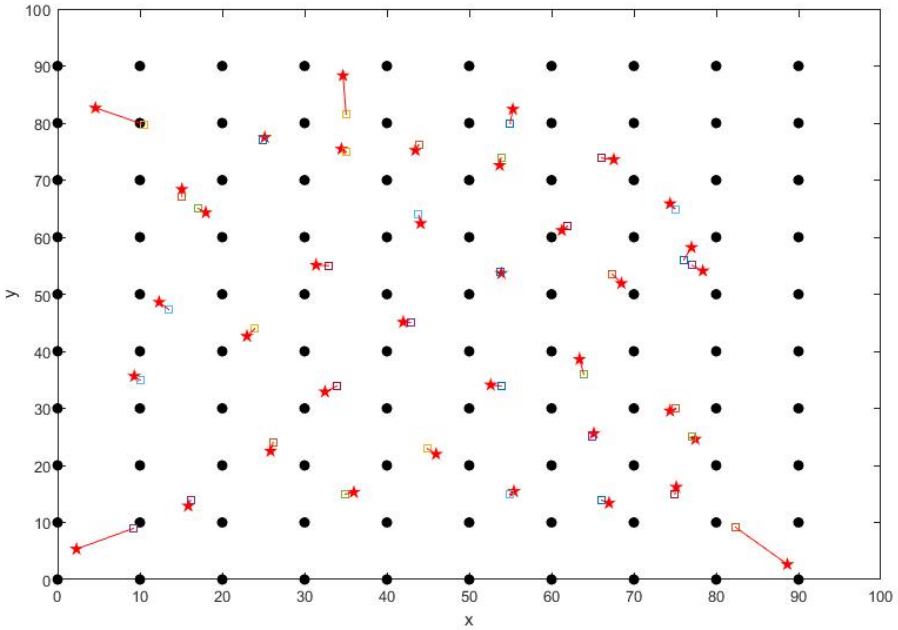

Figure 6.17: Estimated positions and real positions for the testing data set using WCL.

Bars in figure 6.18 represent the estimated error of 37 unknown nodes. such that the error is defined as the variation between the actual and estimated positions of the unknown nodes. Node 2 has the maximum displacement from its actual position so it has the maximum error, also node 22 represents the minimum error.

The case of using the FFNN, the random positions of the unknown nodes are provided as inputs for the network and then the outputs were achieved, such that the output is the estimated coordinate for the position of the unknown nodes. The first FFNN was generated using the structure of 5-20-2 using gradient descent training algorithm, and the results shows as figure 6.19 which illustrate the estimated positions of the 37 unknown nodes.

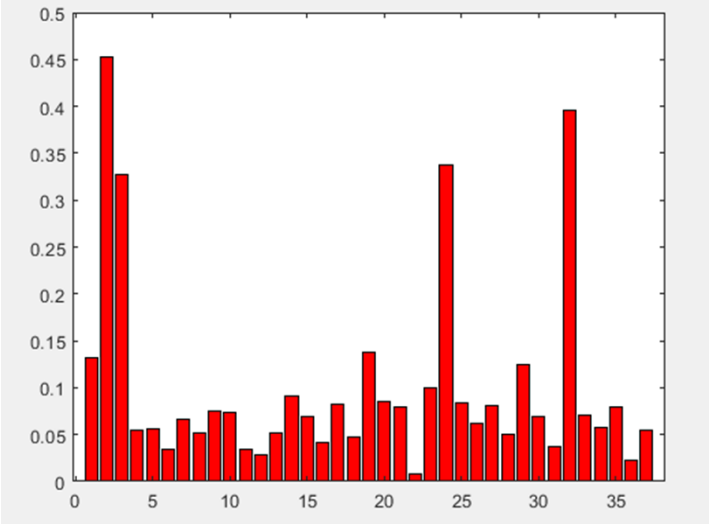

Figure 6.18: Error distribution for 37 unknown nodes using WCL algorithm

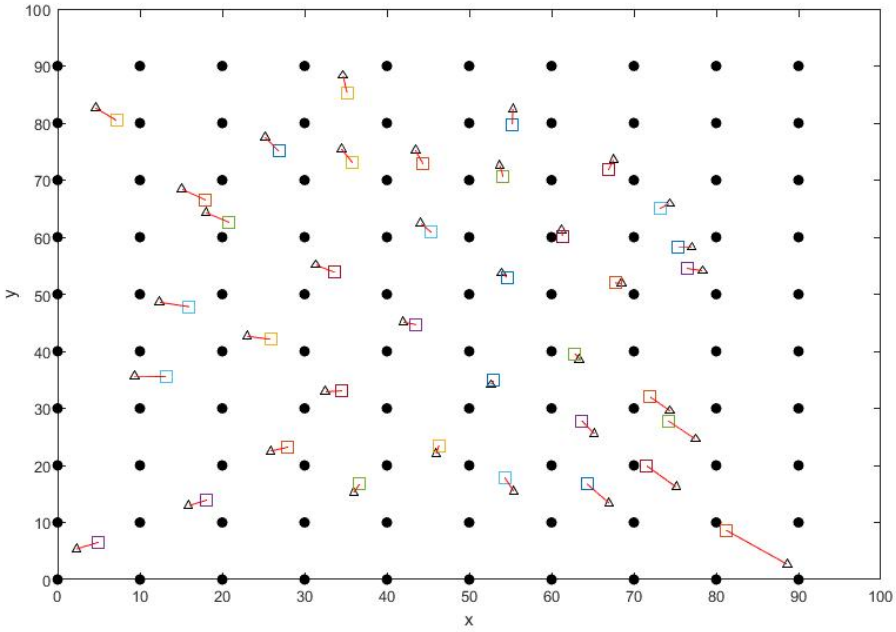

Figure 6.19: Location estimation using FFNN (using GD) for 37 nodes

The actual and predicted positions for sensor nodes indicated by the hollow triangles and squares, respectively.

In addition, error distribution for these 37 unknown nodes using FFNN which uses gradient descent training algorithm is shown in fig-

Table 6.6: Simulation parameters for FFNN

Parameters	Values
Number of neurons	5-20-2
Number of epochs	366
Training function	LM
Performance function	MSE

ure 6.22. Node 2 has the maximum error. The average estimated error for 37 unknown nodes is to 0.1341 m.

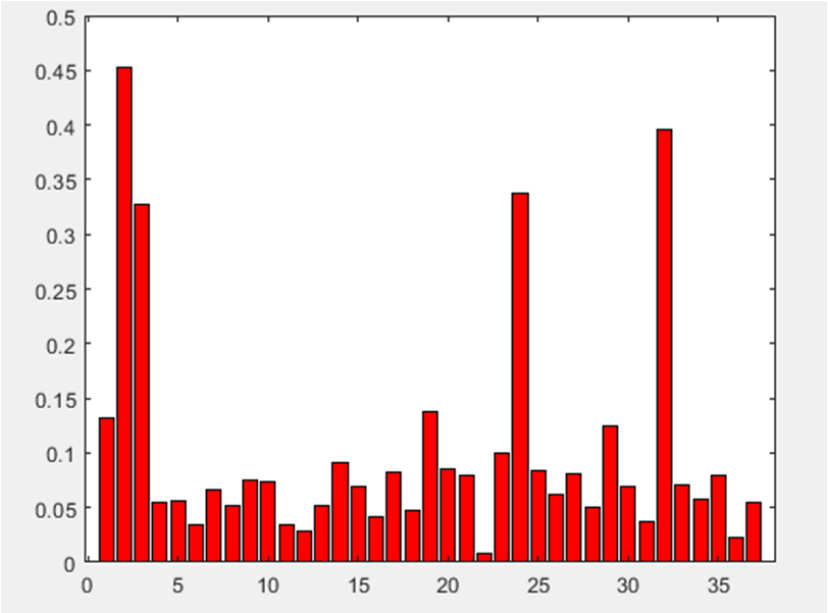

Figure 6.20: Error distribution for 37 unknown nodes using FFNN using GD training algorithm

we can observed that, FFNN using the GD gives a bad results comparing with the WCL algorithm, such that it gives a larger error for each node, so now we apply the network used the Levenberg-Marquadet (LM) algorithm and we get an accurate predicted location.

Table 6.7 shows the simulation parameters for the network .

Figure 6.19 shows the estimated positions for the unknown nodes using FFNN with LM training algorithm

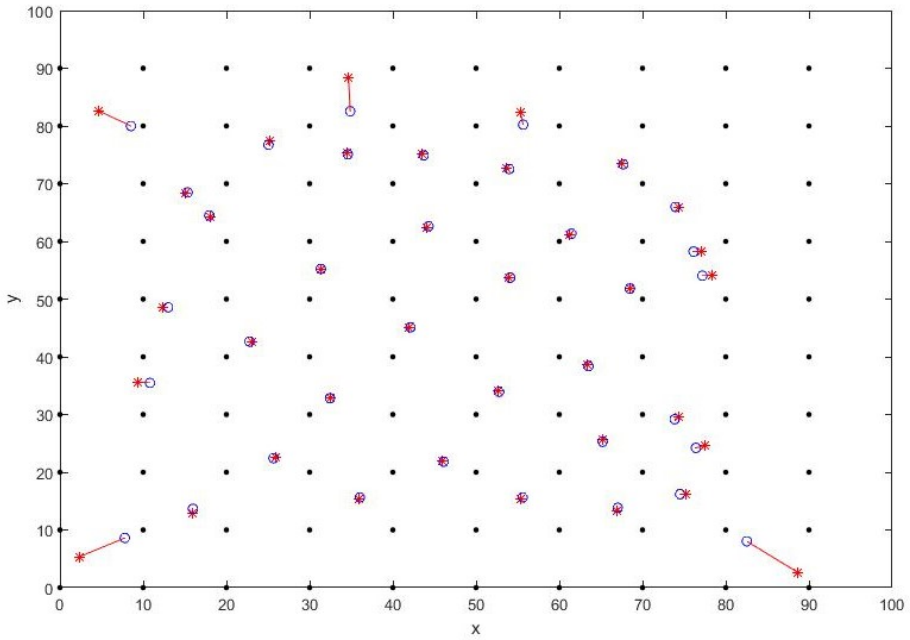

Figure 6.21: Location estimation using FFNN (using LM) for 37 nodes

In figure 6.22 the error distribution for each node were illustrated. Node 2 has the maximum error. The average estimated error is 0.056. So as we observed from the results for this FFNN that gives the lowest error and the best position estimation for each node. These results are the reasons for choosing this network structure (FFNN) insted of the weighted centroid algorithm

The average localization error for all of the unknown sensor nodes is used to evaluate the performance of the localization schemes. We observed from the previous results for all the schemes that the FFNN has a better performance than the weighted centroid algorithm, added

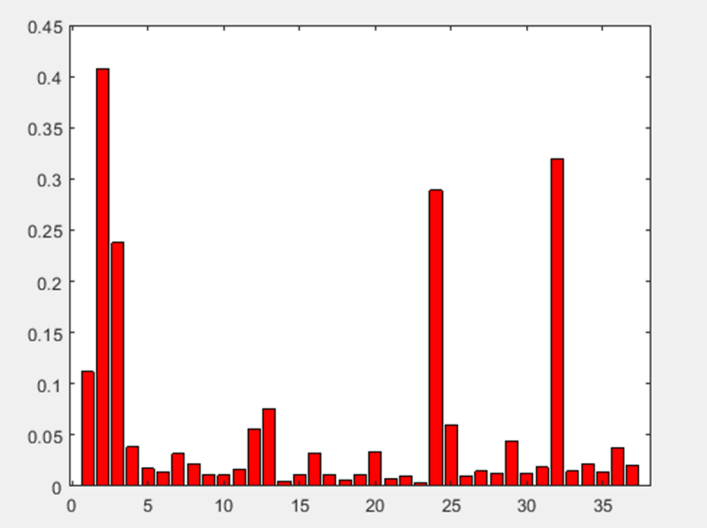

Figure 6.22: Error distribution for 37 unknown nodes using FFNN (using LM)

to that, that the neural network eliminate the need for computing the value of RSSI for each sensor node when we decide to localize it which is reduce a lot of efforts and time. Therefore, implementing FFNN in localization is a better choice to get higher localization accuracy.

Summary of performance comparison for 37 unknown nodes was shown in table 6.7

Table 6.7: Estimated error using WCL FFNN

Method	**Error in meters**
Weighted centroid localization algorithm	0.1002
Feed Forward Neural Network	0.056

Patterned Topologies for WSNs [47] provide a longer network lifetime than the randomly deployed WSNs when they used the same number of sensors, rather than this type of WSNs can efficiently save energy.

Therefore we built our FFNN model to be able to predict the sensor node locations in WSNs with Square-Based Topology areas, which is one of the important patterned topologies.

The estimated locations for the unknown nodes in different WSNs areas with Square-Based Topology and different number of anchor nodes can be obtained using our model with high accuracy as shown in figures 6.23, 6.24, 6.25, 6.26, in addition to the localization error for each sensor node.

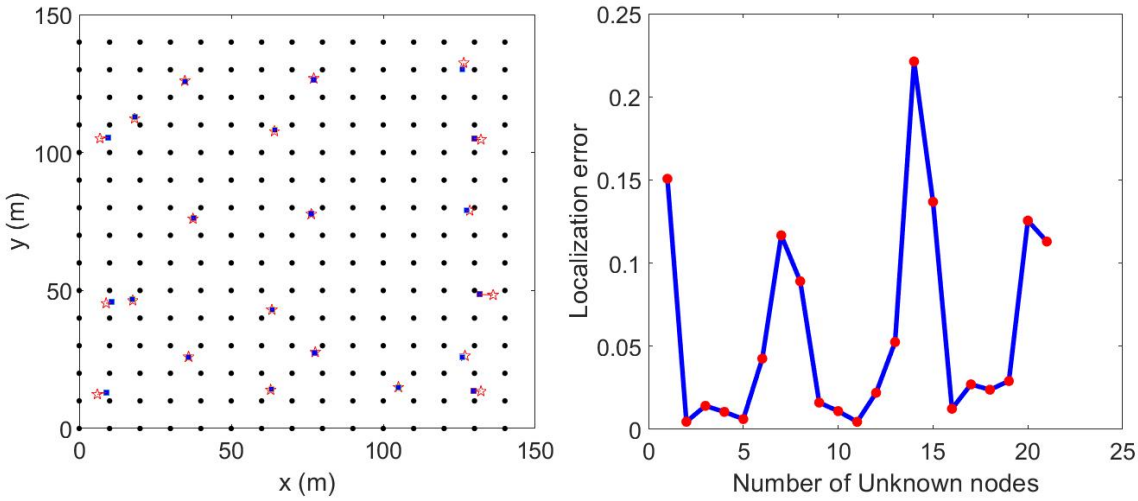

Figure 6.23: Estimated locations and localization error for 21 unknown nodes distributed in area 150 × 150 (m)

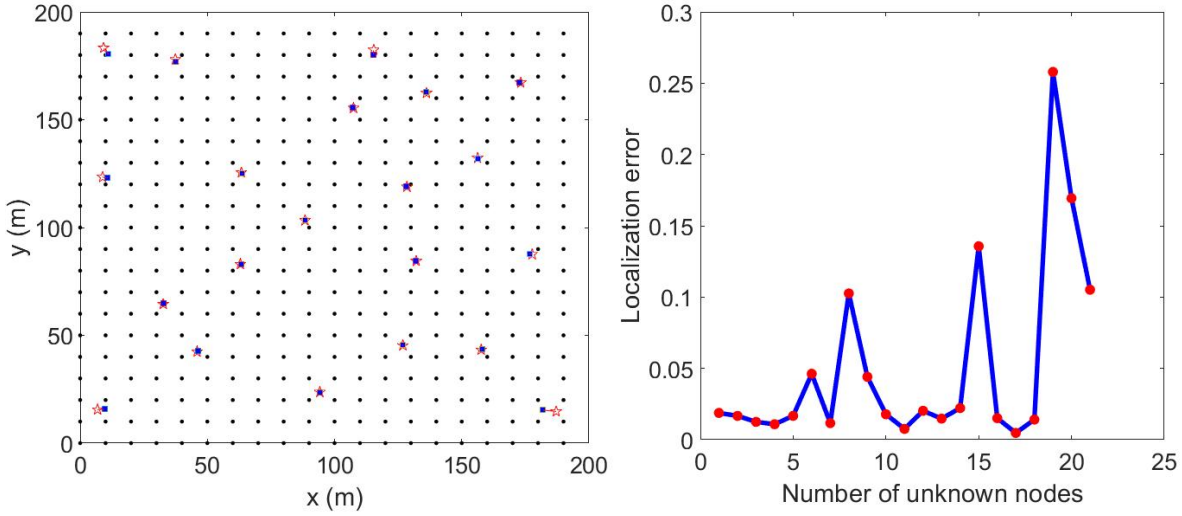

Figure 6.24: Estimated locations and localization error for 21 unknown nodes distributed in area 200 × 200 (m)

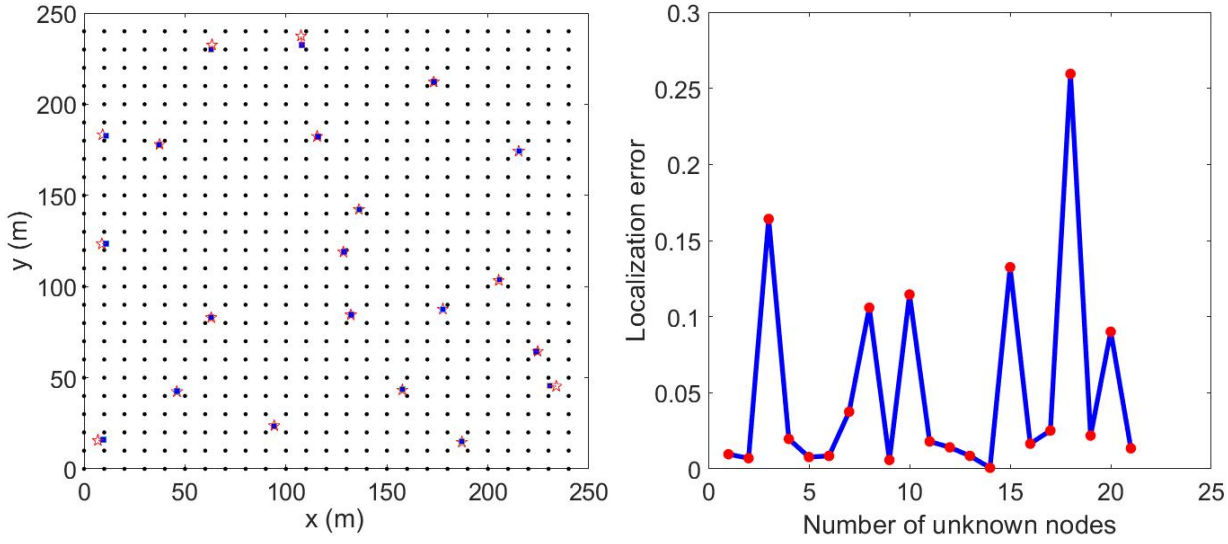

Figure 6.25: Estimated locations and localization error for 21 unknown nodes distributed in area 250 × 250(m)

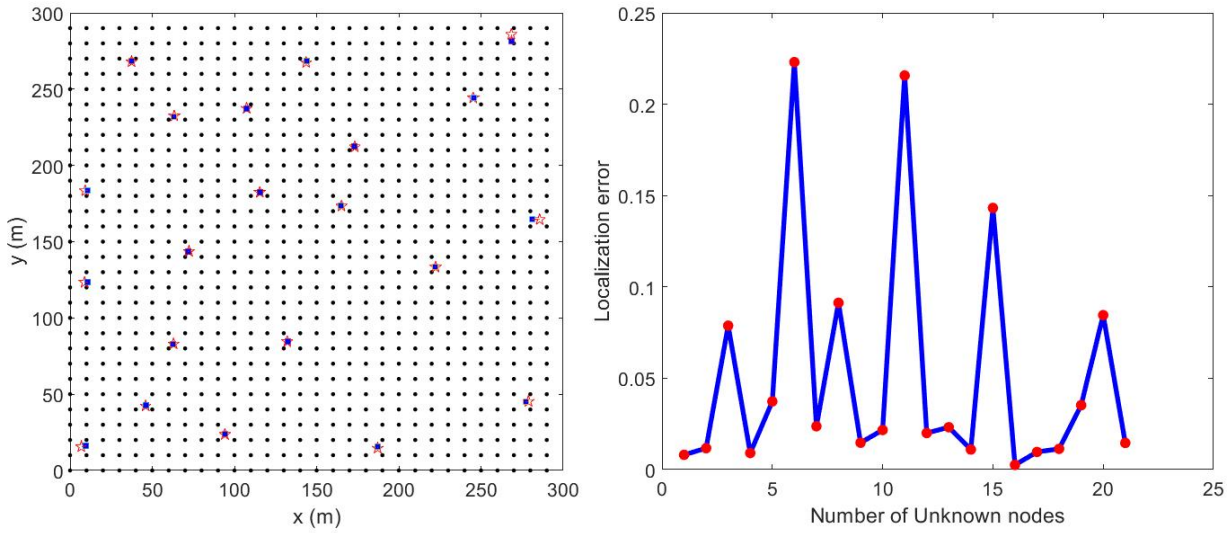

Figure 6.26: Estimated locations and localization error for 21 unknown nodes distributed in area 300 × 300(m)

The red stars represents the the actual locations for the unknown nodes and the blue squares is the estimated locations using FFNN model.

Table 6.8: Estimated localization error using FFNN in different squared areas

Squared area	Error in meters
150 × 150	0.0585
200 × 200	0.0507
250 × 250	0.0415
300 × 300	0.0369

As we obtain from the previous figures that our proposed FFNN with 20 hidden neuron have the ability to predict the sensors locations in different Square-Based Topology areas.

Conclusion

One of the critical issue in the Internet of Things environment is gathering data in an accurate and fast way. the most important thing helping us to guarantee this is the locations of the distributed sensors in the IoT environment. These sensors are the tool to gather the data in the IoT environment, so, when we have the ability to choose the best locations forthese sensors to cover all the interesting area and gather the huge amount of data in an accurate way. The weighted centroid localization algorithm based on RSSI is one of the most important localization algorithms for sensor node in WSNs. We applied this algorithm and got good result. A new and a better and accurate approach is to use machine learning for wireless sensor node localization process, discussed in this thesis. We choose one of the common used machine learning algorithms in the artificial neural network, and we trained it to be able to predict the location of any sensor nodes added to the network in a fast and accurate way. The proposed FFNN model with the structure of 5-20-2 that produces the best correlation coefficient R= 0.99984 compared to other structures. The value of R did not increase when the number of hidden neurons was more than 20. Mean square errors is equal to 1.444 for the training samples which is the smallest value between the other structures. Therefore, the best locations for the sensors node were observed using FFNN with20 neuron in one hidden layer. Also, it is better than the deep learning neural network. These results helped us to the conclusion that the FFNN model is the best way to predict the sensor locations in WSNs in an accurate way. In addition

to the ability for our model to estimate the sensor node locations in different Square-Based topology areas which give the ability to the IoT environment to gather data in an accurate way.

References

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. **Wireless sensor networks: a survey**. Computer networks, 38(4):393– 422, 2002.

[2] P. Bahl, V. N. Padmanabhan, V. Bahl, and V. Padmanabhan. **Radar: An in-building rf-based user location and tracking system**. 2000.

[3] M. H. Bakr and M. H. Negm. **Modeling and design of high-frequency structures using artificial neural networks and space mapping.** In Advances in Imaging and Electron Physics, volume 174, pages 223–

260. Elsevier, 2012.

[4] R. Battiti, N. T. Le, and A. Villani**. Location-aware computing: a neural network model for determining location in wireless lans.** Technical report, University of Trento, 2002.

[5] F. Benbadis, T. Friedman, M. D. De Amorim, and S. Fdida. **Gps- free-free positioning system for wireless sensor networks.** In Second IFIP International Conference on Wireless and Optical Communi- cations Networks, 2005. WOCN 2005., pages 541–545. IEEE, 2005.

[6] J. Blumenthal, R. Grossmann, F. Golatowski, and D. Timmermann. **Weighted centroid localization in zigbee-based sensor networks.** In 2007 IEEE international symposium on intelligent signal processing, pages 1–6. IEEE, 2007.

[7] M. Bokare and A. Ralegaonkar. **Wireless sensor network.** International Journal of Computer Engineering Science (IJCES), 2(3), 2012.

[8] N. Bulusu, J. Heidemann, D. Estrin, et al. **Gps-less low-cost outdoor localization for very small devices**. IEEE personal communications, 7(5):28–34, 2000.

[9] H. Chen, K. Sezaki, P. Deng, and H. C. So**. An improved dv-hop localization algorithm with reduced node location error for wireless sensor networks.** IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, 91(8):2232–2236, 2008.

[10] D. Christin, A. Reinhardt, P. S. Mogre, R. Steinmetz, et al. **Wireless sensor networks and the internet of things: selected challenges.** Proceedings of the 8th GI/ITG KuVS Fachgespräch Drahtlose sensornetze, pages 31–34, 2009.

[11] H. Demuth and M. Beale. **Neural networks toolbox user's guide: For use with matlab.** The MathWorks, Inc, 2002.

[12] L. Doherty, L. El Ghaoui, et al. **Convex position estimation in wireless sensor networks.** In Proceedings IEEE INFOCOM 2001. Confer- ence on Computer Communications. Twentieth Annual Joint Con- ference of the IEEE Computer and Communications Society (Cat. No. 01CH37213), volume 3, pages 1655–1663. IEEE, 2001.

[13] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar. **Next century challenges: Scalable coordination in sensor networks.** In Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking, pages 263–270. ACM, 1999.

[14] L. Gogolak, S. Pletl, and D. Kukolj. **Neural network-based indoor localization in wsn environments.** Acta Polytechnica Hungarica, 10(6):221–235, 2013.

[15] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami. **Internet of things (iot): A vision, architectural elements, and future directions.** Future generation computer systems, 29(7):1645–1660, 2013.

[16] L. Gui, A. Wei, and T. Val. **A two-level range-free localization algorithm for wireless sensor networks.** In 2010 6th International Con- ference on Wireless Communications Networking and Mobile Com- puting (WiCOM), pages 1–4. IEEE, 2010.

[17] T. He, C. Huang, B. M. Blum, J. A. Stankovic, and T. Abdelzaher. **Range-free localization schemes for large scale sensor networks.** In Proceedings of the 9th annual international conference on Mobile computing and networking, pages 81–95. ACM, 2003.

[18] B. Hofmann-Wellenhof, H. Lichtenegger, and J. Collins. **Global positioning system: theory and practice.** Springer Science & Business Media, 2012.

[19] F. Ijaz, H. K. Yang, A. W. Ahmad, and C. Lee. Indoor positioning**: A review of indoor ultrasonic positioning systems.** In 2013 15th International Conference on Advanced Communications Technology (ICACT), pages 1146–1150. IEEE, 2013.

[20] T. C. Karalar and J. Rabaey. **An rf tof based ranging implemen- tation for sensor networks.** In 2006 IEEE International Conference on Communications, volume 7, pages 3347–3352. IEEE, 2006.

[21] B. Karlik and A. V. Olgac. **Performance analysis of various activa- tion functions in generalized mlp architectures of neural networks.** International Journal of Artificial Intelligence and Expert Systems, 1(4):111–122, 2011.

[22] H. Khan, M. N. Hayat, and Z. U. Rehman. **Wireless sensor networks free-range base localization schemes: A comprehensive survey.** In 2017 International Conference on Communication, Computing and Digital Systems (C-CODE), pages 144–147. IEEE, 2017.

[23] Q. Kong, X. Yang, and X. Dai. **Research of an improved weighted centroid localization algorithm and anchor distribution.** In 2010 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, pages 400–405. IEEE, 2010.

[24] P. Kułakowski, J. Vales-Alonso, E. Egea-López, W. Ludwin, and J. Garc´ıa-Haro. **Angle-of-arrival localization based on antenna arrays for wireless sensor networks.** Computers & Electrical Engineering, 36(6):1181–1186, 2010.

[25] P. Kumar, L. Reddy, and S. Varma. **Distance measurement and error estimation scheme for rssi based localization in wireless sensor networks.** In 2009 Fifth international conference on wireless com- munication and sensor networks (WCSN), pages 1–4. IEEE, 2009.

[26] S. Kumar and S.-R. Lee. **Localization with rssi values for wireless sensor networks: An artificial neural network approach.** In Interna- tional Electronic Conference on Sensors and Applications, volume 1. Multidisciplinary Digital Publishing Institute, 2014.

[27] J. Lalis, B. Gerardo, and Y. Byun. **An adaptive stopping crite- rion for backpropagation learning in feedforward neural network.** International Journal of Multimedia and Ubiquitous Engineering, 9(8):149–156, 2014.

[28] I. Lee and K. Lee. **The internet of things (iot): Applications, investments, and challenges for enterprises.** Business Horizons, 58(4):431– 440, 2015.

[29] Y.-D. Lee and W.-Y. Chung. **Wireless sensor network based wear- able smart shirt for ubiquitous health and activity monitoring.** Sensors and Actuators B: Chemical, 140(2):390–395, 2009.

[30] H. Liu**. On the levenberg-marquardt training method for feed-forward neural networks.** In 2010 sixth international conference on natural computation, volume 1, pages 456–460. IEEE, 2010.

[31] W. S. McCulloch and W. Pitts. **A logical calculus of the ideas im- manent in nervous activity.** The bulletin of mathematical biophysics, 5(4):115–133, 1943.

[32] D. Mishra, A. Yadav, S. Ray, and P. K. Kalra**. Levenberg-marquardt learning algorithm for integrate-and-fire neuron model.** Neural Information Processing-Letters and Reviews, 9(2):41–51, 2005.

[33] S. L. Mohammed. **Distance estimation based on rssi and log-normal shadowing models for zigbee wireless sensor network.** Engineering and Technology Journal, 34(15 Part (A) Engineering):2950–2959, 2016.

[34] D. Moore, J. Leonard, D. Rus, and S. Teller. **Robust distributed network localization with noisy range measurements.** In Proceedings of the 2nd international conference on Embedded networked sensor systems, pages 50–61. ACM, 2004.

[35] M. Negnevitsky and A. Intelligence**. A guide to intelligent systems.** Artificial Intelligence, 2nd edition, pearson Education, 2005.

[36] D. Niculescu and B. Nath**. Ad hoc positioning system (aps).** In GLOBECOM'01. IEEE Global Telecommunications Conference (Cat. No. 01CH37270), volume 5, pages 2926–2931. IEEE, 2001.

[37] D. Niculescu and B. Nath. **Ad hoc positioning system (aps) using aoa.** In IEEE INFOCOM 2003. Twenty-second Annual Joint Con- ference of the IEEE Computer and Communications Societies (IEEE Cat. No. 03CH37428), volume 3, pages 1734–1743. Ieee, 2003.

[38] D. Niculescu and B. Nath**. Dv based positioning in ad hoc networks**. Telecommunication Systems, 22(1-4):267–280, 2003.

[39] T. Pradeep, P. Srinivasu, P. Avadhani, and Y. Murthy. **Comparison of variable learning rate and levenberg-marquardt back-propagation training algorithms for detecting attacks in intrusion detection systems.** International Journal on Computer Science and Engineering, 3(11):3572, 2011.

[40] M. Ramazany and Z. Moussavi. **Localization of nodes in wireless sensor networks by mdv-hop algorithm.** ARPN Journal of Systems and Software, 2(5):166–171, 2012.

[41] D. E. Rumelhart, R. Durbin, R. Golden, and Y. Chauvin. **Backpropagation: The basic theory. Backpropagation: Theory, architectures and applications**, pages 1–34, 1995.

[42] S. Samara and E. Natsheh**. Modeling the output power of heterogeneous photovoltaic panels based on artificial neural networks using low cost microcontrollers.** Heliyon, 4(11): e00972, 2018.

[43] S. Samara and E. Natsheh. **Intelligent real-time photovoltaic panel monitoring system using artificial neural networks.** IEEE Access, 7:50287–50299, 2019.

[44] A. Savvides, H. Park, and M. B. Srivastava**. The bits and flops of the n-hop multilateration primitive for node localization problems.** In Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications, pages 112–121. ACM, 2002.

[45] F. Shaker, A. H. Monadjemi, and H. YAZDANPANAH. **Comparing artificial neural networks and linear regression model in predicting soil surface temperature.** International Journal, 5(6):2305–1493, 2014.

[46] P. Singh, B. Tripathi, and N. P. Singh. **Node localization in wireless sensor networks.** International journal of computer science and information technologies, 2(6):2568–2572, 2011.

[47] H. Tian, H. Shen, and T. Matsuzawa. **Developing energy-efficient topologies and routing for wireless sensor networks.** In IFIP International Conference on Network and Parallel Computing, pages 461–469. Springer, 2005.

[48] S. Y. M. Vaghefi and R. M. Vaghefi. **A novel multilayer neural network model for toa-based localization in wireless sensor networks.** In The 2011 International Joint Conference on Neural Networks, pages 3079–3084. IEEE, 2011.

[49] M. Viswanathan. **Simulation of digital communication systems using matlab.** Mathuranathan Viswanathan at Smashwords, 2013.

[50] M. Viswanathan. **Wireless communication systems in matlab.** Independently published, 2018.

[51] Q. D. Vo and P. De. **A survey of fingerprint-based outdoor localization.** IEEE Communications Surveys & Tutorials, 18(1):491–506, 2015.

[52] Z.-M. Wang and Y. Zheng**. The study of the weighted centroid localization algorithm based on rssi.** In 2014 International Confere on Wireless Communication and Sensor Network, pages 276–279. IEEE, 2014.

[53] G. Werner-Allen, K. Lorincz, M. Ruiz, O. Marcillo, J. Johnson, J. Lees, and M. Welsh. **Deploying a wireless sensor network on an active volcano.** IEEE internet computing, 10(2):18–25, 2006.

[54] B. M. Wilamowski, N. J. Cotton, O. Kaynak, and G. Dundar**. Method of computing gradient vector and jacobean matrix in ar- bitrarily connected neural networks.** In 2007 IEEE International Symposium on Industrial Electronics, pages 3298–3303. IEEE, 2007.

[55] L. Xu, K. Wang, Y. Jiang, F. Yang, Y. Du, and Q. Li**. A study on 2d and 3d weighted centroid localization algorithm in wireless sensor networks.** In 2011 3rd International Conference on Advanced Computer Control, pages 155–159. IEEE, 2011.

[56] H. Yu and B. M. Wilamowski. **Levenberg-marquardt training.** Industrial electronics handbook, 5(12):1, 2011.

[57] J. Zheng, Y. Liu, X. Fan, and F. Li. **The study of rssi in wireless sensor networks.** In 2016 2nd International Conference on Artificial Intelligence and Industrial Engineering (AIIE 2016). Atlantis Press, 2016.

[58] D. Y. Zou, C. Li, and T. F. Han. **Research of centroid localization algorithm based on grid distribution.** In Applied Mechanics and Materials, volume 738, pages 401–404. Trans Tech Publ, 2015.

جامعة النجاح الوطنية
كليه الدراسات العليا

المبادئ والممارسات والخوارزميات الرياضية المستخدمة لتحليل بيانات انترنت الأشياء باستخدام نهج التعلم الالي

إعداد

بتول سمير سلامة سليمان

إشراف

د.محمد شرف

أ. د. ناجي قطناني

قدمت هذه الأطروحة استكمالا لمتطلبات الحصول على درجة الماجستير في الرياضيات المحوسبة بكلية الدراسات العليا في جامعة النجاح الوطنية في نابلس- فلسطين

2019

المبادئ والممارسات والخوارزميات الرياضية المستخدمة لتحليل بيانات انترنت الأشياء باستخدام نهج التعلم الالي

إعداد

بتول سمير سلامة سليمان

إشراف

د. محمد شرف

أ. د. ناجي قطناني

الملخص

بيئة انترنت الأشياء تقوم بإنتاج بيانات كثيرة في كل وقت، وهذه البيانات تحتاج الى الجمع والتحليل ومن ثم اتخاذ الحدث والتصرف المناسب. الكثير من تطبيقات انترنت الاشياء تعتبر مواقع المعلومات التي تم جمعها معلومات غاية في الأهمية، لذلك تعتبر دقة اختيار مواقع المستشعرات المسؤولة عن جمع البيانات قضية مهمة جدا. يوجد العديد من الطرق التقليدية لعملية التوطين داخل شبكة الاستشعار اللاسلكية. في هذه الأطروحة، تم اقتراح استخدام الشبكة العصبية الاصطناعية الذكية لعملية توطين المستشعرات الخاصة بجمع البيانات في بيئة انترنت الاشياء كنهج جديد، وتم تنفيذ نموذج الشبكة العصبية الذكية المقترح على مساحات مختلفة من شبكة المستشعرات اللاسلكية باستخدام برنامج الماتلاب. اخيرا تمت المقارنة بين النموذج المقترح من الشبكة العصبية الذكية ذات طبقية مخفية واحدة مع الشبكية العصبية العميقة، بالإضافة الى مقارنة نتائجنا بنتائج خوارزمية توطين النقطة الوسطى الحرجة وهي نظرية تقليدية تستخدم لعملية التوطين عادةً في شبكة الاستشعار اللاسلكية. واظهرت النتائج ان نموذجنا المقترح يقوم بتقدير المواقع للمستشعرات بدقة اكبر من المواقع التي تم تقديرها باستخدام النظريات التقليدية.