

KADIR HAS UNIVERSITY
GRADUATE SCHOOL OF SCIENCE AND ENGINEERING



LOCATION ESTIMATION OF BASE STATION OF MOBILE
PHONES USING ARTIFICIAL NEURAL NETWORKS

RAMAZAN CENGİZ

January, 2016



Ramazan Cengiz

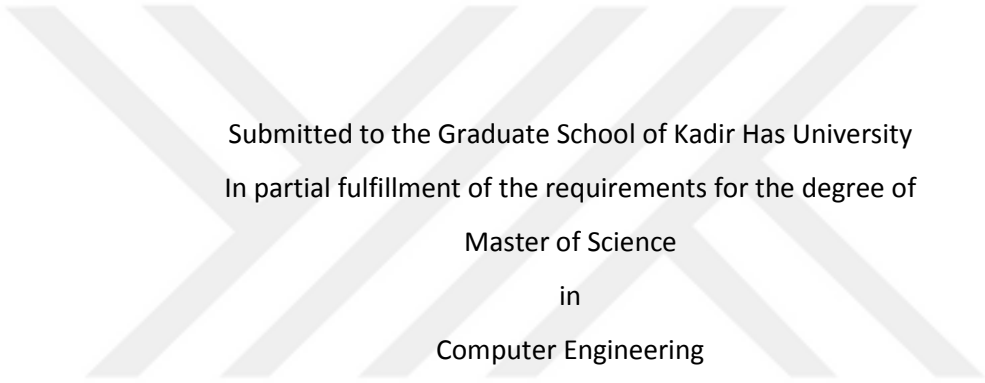
MS Thesis

2016

LOCATION ESTIMATION OF BASE STATION OF MOBILE PHONES USING ARTIFICIAL NEURAL NETWORKS

RAMAZAN CENGİZ

B.S., Computer Engineering, Kadir Has University, 2012



Submitted to the Graduate School of Kadir Has University
In partial fulfillment of the requirements for the degree of
Master of Science
in
Computer Engineering

KADIR HAS UNIVERSITY

January, 2016

KADIR HAS UNIVERSITY
GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

LOCATION ESTIMATION OF BASE STATION OF MOBILE
PHONES USING ARTIFICIAL NEURAL NETWORKS

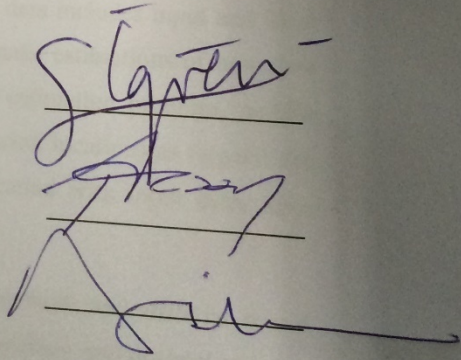
RAMAZAN CENGİZ

APPROVED BY:

Asst. Prof. Dr. Arif Selçuk Öğrenci
(Thesis Supervisor)

Asst. Prof. Dr. Taner Arsan

Asst. Prof. Dr. Atilla Özmen



APPROVAL DATE: 7.1.2016

LOCATION ESTIMATION OF BASE STATION OF MOBILE PHONES USING ARTIFICIAL NEURAL NETWORKS

Abstract

Finding the location of the base station of mobile phones is a problem. There are different approaches and methods for finding location in the literature. A solution to this problem is using “supervised learning” method. Recently, supervised learning technology is used in solving problems. For example, computer is trained and fed by necessary data. This data includes input and ideal targets. After computer is trained properly, it can make estimations of new data and give reasonable results. For the base station estimation program, mobile phone data(as input) of mobile phones and base station locations(as targets) are given to software. Software can estimate base location with input data, after training software

Matlab is a good program for supervised learning methods and artificial neural networks. For finding the optimal neural network topology and optimal training methods, Matlab is used in this project. Experiments are made with different data,different neural networks and different training methods. Results and outputs are observed. After finding best neural network topology, experiments are made with real data. Raw data are collected from mobile phones and after converting it into useful numbers and after normalizing these numbers, performance of the program is evaluated. Finally, this solution is implemented in Java platform.

In this thesis, we present base location estimation program. This program takes as input a training data and test data. It uses training data to estimate the output of the test data. As output it creates a report that shows the estimated base station location.



MOBİL CİHAZLARIN BAZ İSTASYONLARININ YERİNİ SİNİR AĞLARI KULLANARAK BULMA

Özet

Mobil cihazların sinyal aldığı baz istasyonlarının yerini bulma bir problemdir. Literatürde yer bulma için çeşitli yaklaşımlar ve yöntemler vardır. Bu probleme yönelik bir çözüm de "gözetimli öğrenimdir". Günümüzde "gözetimli öğrenim" birçok problemin çözümünde kullanılır. Örneğin, bilgisayar ilgili veri ile eğitilir. Bu veri giriş verisi ve hedef verisini içerir. Bilgisayar bu veri ile eğitilmesinin ardından bir dahaki sefere yeni giriş verisi geldiğinde mantıklı çıktılar (sonuçlar) verebilir. Baz istasyonu yeri tahmin programı için giriş verisi mobil kullanıcıdan alınan giriş verisi ve hedef verisi (baz istasyonları yeri) programa verilir. Eğitim sonrasında yeni gelen giriş verisi bilgisayara verildiğinde, bilgisayar baz istasyonu lokasyonu tahmini yapabilir.

Bu tezde, Matlab uygulamasından faydalanılmıştır. Matlab gözetimli öğrenimin yapay sinir ağları üzerinde uygulanması için iyi bir programdır. En iyi yapay sinir ağları topolojisini bulmak, en iyi eğitim metodunu bulmak için bu projede Matlab kullanılmıştır. Farklı verilerle, farklı yapay ağlarla ve farklı eğitim metodlarıyla deneyler yapılmıştır. En iyi yapay ağ teknolojisi bulunduktan sonra yapay veriden gerçek verilerle deney yapılmaya geçilmiştir. Cep telefonlarından ham veri toplanmıştır. Bu veri gereksiz bilgilerden ayıklandıktan sonra, sadeleştirilerek ve özeti alınarak anlamlı veri haline getirilmiştir. Sonrasında bu veri normalize edilerek programın çalıştırabileceği hale ve kullanıcıya kolaylık sağlar hale getirilmiştir. Sonrasında bu programın performansı deneylerle ölçülmüştür. En sonunda program Java platformunda implement edilmiştir.

Bu tezde baz istasyonu yeri tespit etme programı anlatılmıştır. Bu program girdi olarak eğitim ve test datası alır. Bu program eğitim verisini kullanır ve kendisine verilen test verisinin çıktılarını buna göre tahmin eder. Çıktı olarak baz istasyonu yerlerini gösteren bir rapor verir.



Acknowledgements

I would like to express my deep-felt gratitude to my advisor, Assistant Professor Selçuk Öğrenci of the Electronic Engineering Department at Kadir Has University. This thesis would not have been possible without his valuable guidance, constant motivation and constructive suggestions. He always gave me his time and support led me with constructive suggestions. I wish all students would benefit from his experience and ability.

Besides, I would like to thank Assistant Professor Taner Arsan of the Computer Engineering Department at Kadir Has University, for all supports they provided me throughout my all academic and graduate career.

I thank to my dear friend İlktan Ar and my all other friends who supported me during the thesis. And I thank Aykut Çayır who made the data collection mobile program.

Lastly, I thank to my family for their supports and understanding on me in completing this project. Without any of them mentioned above, I would face many difficulties while doing this thesis.

Table of Contents

Abstract	i
Özet	iii
Acknowledgements	v
Table of Contents	vi
List of Tables	viii
List of Figures	ix
List of Abbreviations	xi
1 Introduction	1
2 Overview of Geolocation methods	5
2.1 Location accuracy methods.....	5
2.2 Geolocation implementations.....	8
3 Overview of Neural Networks and Training methods	15
4 Data collection	26
5 Algorithms and Source Code	39
6 Tests and Results	63
7 Conclusion	82
References	83
Curriculum Vitae	85

List of Tables

Table 3.1: Training algorithms comparison

Table 3.2: Training algorithms' performances

Table 6.1: Matlab test results sample1

Table 6.2: Matlab test results sample2

Table 6.3: Encog test results sample1

Table 6.4: Encog test results sample2

Table 6.5: Encog test results sample3

Table 6.6: Encog test results sample4

Table 6.7 Encog test results sample5

Table 6.8: Resilient Propagation Comparisons

Table 6.9: Levenberg Marquard Comparisons

LIST OF FIGURES

- Figure 2.1: Cell of origin
Figure 2.2: Time of arrival
Figure 2.3: Time difference of arrival
Figure 2.4: Angle of arrival
Figure 4.1 : Database queries of data
Figure 4. 2:Raw data
Figure 4. 3:Processed data
Figure 4. 4:Final data
Figure 4.5:Cell 6700 map
Figure 4.6:Cell 29515 map
Figure 4.7:Cell 40143 map
Figure 4.8:Cell 37442 map
Figure 4.9:Cell 53665 map
Figure 4.10:Cell 35610 map
Figure 4.11:Cell 43078 map
Figure 5.1:Flowchart of two outputs
Figure 5.2:Neural network topology for two outputs
Figure 5.3:Flowchart of one output
Figure 5.4: Neural network topology for one outputs
Figure 5.5: BrowseTrain Button
Figure 5.6: BrowseTest Button
Figure 5.7: LoadNetwork Button
Figure 5.8: TrainandTest Button
Figure 5.9: LoadandTest Button
Figure 5.10: Levenberg-Marquard radio button
Figure 5.11: Resilient Propagation radio button
Figure 5.12: Hidden Layer Neuron InputText
Figure 5.13: Output Neurons InputText

Figure 6.1: Base locations on map
Figure 6.2: Cell 6700 matlab estimation map
Figure 6.3: Cell 11749 matlab estimation map
Figure 6.4: Cell 29515 matlab estimation map
Figure 6.5: Cell 32571 matlab estimation map
Figure 6.6: Cell 29554 encog estimation map
Figure 6.7: Cell 25625 encog estimation map
Figure 6.8: Cell 32430 encog estimation map
Figure 6.9: Cell 35610 encog estimation map
Figure 6.10: Distance-Hidden Layer Comparisons in Resilient Propagation
Figure 6.11: Distance-Hidden Layer Comparisons in Levenberg Marquard

LIST OF ABBREVIATIONS

Angle-of-Arrival: (AOA)

Cell identification: (Cell ID)

Global Positioning System: (GPS)

Round Trip Time :(RTT)

TDOA:(Time Difference of Arrival)

Time-of-arrival :(TOA)

Chapter 1

Introduction

The problem is to find the base station of mobile devices by using the most feasible methods. This problem, which we refer to as localization, is a challenging one, and yet extremely crucial for many applications of very large networks of devices[1]. There are some used methods in the literature.

A lot of related articles and presentations are read and know-how is acquired. Some of them are angulation, cell of identity, lateration and received signal strength methods. The first step is to investigate and find which method will help us more in the process. The most widely known, using the internal hardware of the cellphone, is satellite positioning using GPS but WiFi, Bluetooth, and augmented sensor networks can also be employed [2], [3], [4]. The cost, the easiness to implement and the performance of the methods are examined and comparisons are discussed.

The given input is the data of the mobile devices. The data gives the coordinates and received signal strength of the mobile devices. Boundary of the base station cell can be estimated by using this data . There are various methods for estimating the location of the base station. Firstly, location can be found by benefitting from the signal strength lines. Received signals can form a map in which various signal lines exist. Strong signal lines are closer to base station while weak signal lines are further to the base station. From this map the location of the base station can be approximated. In order to benefit from this data, data should be normalized. Then, it must be preprocessed. And after using the data, it should be post-processed.

An improvement to the map model is to use artificial neural networks and learning methods. These methods help us predicting the values. This helps us make the

estimation process faster and more accurate. Moreover with these methods data can be visualized and controlled so good that data can be interpreted easier and better. These methods benefit from neural networks.

As an input that comes from the mobile phone is the longitude, latitude, cell identity and rscp values of the phone are taken and as output the longitude and latitude(location) of the base station is estimated. Neural networks are very helpful in achieving this aim. Once appropriate neural network is fed by appropriate data, if new data comes to trained artificial neural network, it can estimate the desired results. In this study artificial neural networks and supervised learning methods are used to find base location of mobile phones.

As the first step, a project planning is made. The steps that have to be taken are determined. Time planning is made. According to this plan, project progresses. Firstly, Matlab is used for the project with artificial data, then the project is realized by Java with real values. Raw data can't be used for the experiments on the neural networks. Some redundant data should be stripped from the main data. Furthermore data should be normalized. The normalization and cleansing of the data is one of the steps. By normalization, meaningful data is obtained.

Finding the best method for training is also a step. Firstly, the most suitable neural network has to be determined. Secondly, the most suitable training method has to be determined. The number of neurons in the input, output and hidden layers are determined. The optimum neuron numbers in the hidden layer will give best results. The most suitable transfer function in the hidden layers is important

Solution by Supervised Learning Method

1. Obtain data and data cleansing

For three different cell regions(dense urban, urban and rural) obtain data for the neighbour 10-20 cell information. After that, redundant and unnecessary data must be cleansed and finally, necessary and helpful data will be obtained

2. Data Preprocessing

Before we use the data in the program, data must be preprocessed. After the data is preprocessed, we can get better results easier.

- For every cell, record and base station data is normalized. There are 3 different methods.
- Normalize according to cell's lower right corner.
- Normalize based on cell's geometrical middle point.
- Normalize based on cell's gravity center.
- Grouping : Cells form training, validation and test sets.

3. Realize algorithm by using Matlab

An algorithm must be made if we want to train data in Matlab. Matlab codes are used to realize the algorithm which trains data.

4. Realize algorithm by using Java

An algorithm must be made if we want to train data in Java. Encog library and Java codes are used to realize the algorithm which trains data.

5. Training and Results

Final project stage is training. After enough trainings are made, results of the trainings will be discussed.

- Dense urban training, urban training and rural training
- Utilization of various artificial neural network topology. (Neuron numbers in the hidden layer, hidden layer neuron numbers).

Objective: To find results for the test set using back-propagation training for data whose base station is known.

Input: Normalized longitude,latitude values.

- Output:Denormalized longitude,latitude values

Chapter 2

Literature Review

Overview of Geolocation methods

In this chapter, literature review about the geolocation methods will be explained. These methods are about locating a place on earth. They have already been used before. Each method has its own way to calculate the position of a place on earth. Each of them uses separate parameters to find location. After explaining theory of these methods, implementations of these methods are discussed.

An Analysis of Base Station Location Accuracy within Mobile-Cellular Networks

Received Signal Strength Indication (RSSI) measurements

RSSI benefits from the received signal strength and its relation between the distance from the base stations(BS) to the mobile station. Due to the complex propagation mechanisms accuracy of this method is decreased. This problem, which we refer to as localization, is a challenging one, and yet extremely crucial for many applications of very large networks of devices. Radio propagation models [5] in various environments have been well researched and have traditionally focused on predicting the average received signal strength at a given distance from the transmitter (large scale propagation models), as well as the variability of the signal strength in close spatial proximity to a location (small scale or fading models).

Angle-of-Arrival (AOA)

It locates the mobile by using the angle-of-arrival(AOA) of a signal from many Base Stations(BSs) from the mobile. A line of bearing between the and mobile estimates the AOA distance. When multiple LOBs intersect, mobile position is estimated.

GPS (Global Positioning System)

The most accurate locationing method is GPS. It uses satellite signals and it provides accurate estimation. GPS solves the problem of localization in outdoor environments for PC class nodes. But when signals are prevented, for example in indoors setting, Assisted GPS method needs hardware development. In addition new algorithms have greatly improved the accuracy and efficiency with which a cellphone can calculate its position [6], [7].

Time-of-arrival (TOA)

The fourth category determines mobile location by measuring the time-of-arrival (TOA) which provides the estimate of the distance between the BS and the mobile since electromagnetic waves propagate at the speed of light[8]. In this method, the mobile location is determined by measuring the time-of-arrival(TOA). The distance between the BS and the mobile is estimated by using electromagnetic wave propagation speed. When multiple TOAs intersect, mobile location is estimated. The most simple technic for GSM to measure the TOA is “time advance”(TA). TA measures run-trip time between the BS and the mobile.

TDOA(Time Difference of Arrival)

TDOA(Time Difference of Arrival), is another technique based on propagation delay time. It calculates the differences in TOAs of a mobile signal at multiple pairs of BSs. Each TDOA forms a hyperbolic curve in which mobile may lie.

UL-TOA (Uplink TOA) and E-OTD (Enhanced Observed Time Difference) are standard TDOA techniques for GSM networks. NLOS propagation and time synchronization between stations are potential disadvantages for propagation-time-based techniques.

Most feasible methods for location estimation of a cellphone within a mobile-cellular networks depends on the location of base stations(BSs) as known reference points for calculating the estimated position of the cellphone. The other techniques are WiFi,Bluetooth and augmented sensor networks. The accuracy of techniques depend on the technology,line-of-sight(LOS), and sensor network coverage. Assisted-GPS(A-GPS) uses mobile network information in combination with internal hardware of the cellphone. A-GPS uses network resources in the case

of poor signal reception. Location methods based primarily on mobile-cellular network information is popular.

Cell identification (Cell ID)

Cell identification (Cell ID) is the simplest location estimation method available, but also the least accurate[9]. A wedge shaped area, comprising roughly a third of the cell(for three sectorized sites) is at best estimated. But if omnidirectional antennas in low-density single sector cells are used, entire circular area for sites can be included.

Round Trip Time (RTT)

Round Trip Time(RTT) is a measure of the distance. It measures the time taken by a radio signal to travel from the base station to cellphone and back. It reduces drastically the estimated location area compared to the Cell ID method for the same site.

Cell ID and RTT combine methods to provide an location estimation where these areas overlap

Location Accuracy

In addition to accuracy degrading, these challenges can also increase the cost of location estimation. These challenges are non-line-ofsight and multi-path propagation of radio waves, base station density(or lack of) and accuracy of base station locations.

The methods of location estimation consist of two groups. The first group doesn't depend on base station location and aren't affected by the accuracy with which these locations are known. These methods are A-GPS, probabilistic fingerprinting,bulk map-matching and centroid algorithm. The second group has methods that estimate location of the cellphone relative to the base station location and depend on the accuracy of network base station location. This group includes Cell-ID based methods,Cell ID and RTT,The time of arrival(TOA) and its enhancements.

Location Tracking Approaches(Implementations)

Location tracking and positioning systems are classified by the measurement techniques. These techniques are used to find the mobile device location(localization). Real Time Location Systems (RTLS) are grouped into four categories. They find the position on the basis of the following:

1. Cell of origin(nearest cell)
2. Distance(lateration)
3. Angle(angulation)
4. Location patterning(pattern recognition)

A RTLS designer can choose to implement one or more of the above techniques.

Cell of Origin

This method indicates the cell with which the mobile device is registered so it finds the position of the mobile device. Cell origin doesn't need any complicated algorithm and thus its positioning performance is rapid. All cellular-based RF systems and cell-based WLANs can be easily and cost-effectively adapted to cell of origin positioning. This approach's coarse granularity is a drawback. Some users who want more precise results also implement lateration, pattern recognition and angulation besides this technique for better results. Cell of origin figure is shown at Figure 2.1.

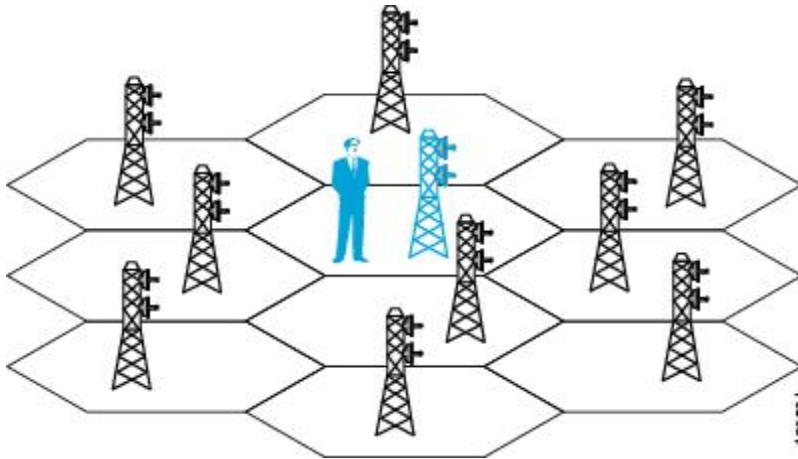


Figure 2.1: Cell of Origin

Distance-Based (Lateration) Techniques

Time of Arrival(ToA) systems measure the arrival time of a signal transmitted from a mobile device to several receiving sensors. Signals travel with a known velocity(approximately the speed of the light.)The ToA requires that all receiving sensors and the mobile device are synchronized with a precise time source. Distance can be thought as a radius of a circle area estimates the mobile location. Three sensors are implemented in ToA tri-lateration and this increases mobile location estimation accuracy. Three circular area of sensors give the estimated location . Large amount of multipath, interference, or noise creates error in ToA implemented positioning systems. The Global Positioning System (GPS) is a kind of ToA system. Timing is provided by atomic clocks precisely which is necessary in ToA systems. Figure 2.2 shows the mechanism.

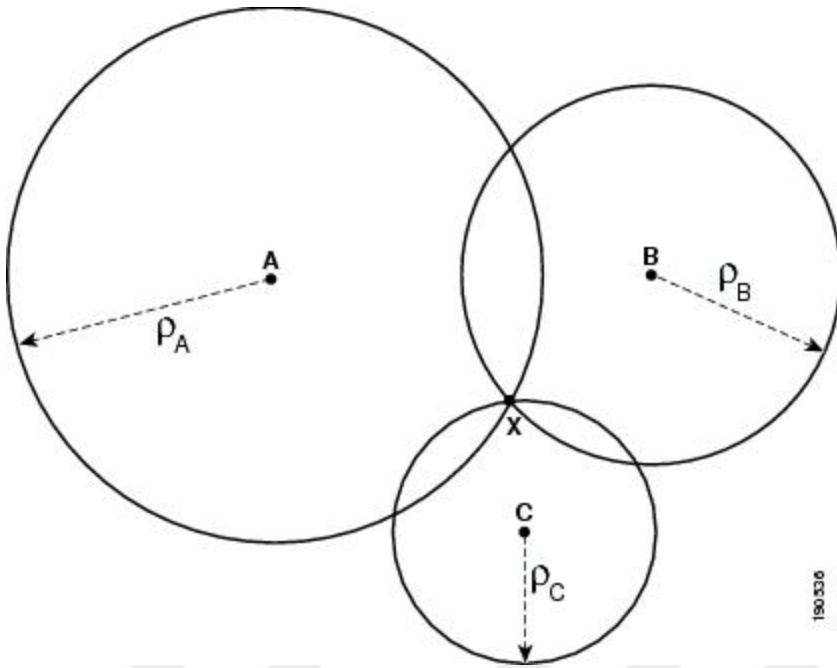


Figure 2.2: Time of Arrival

Time Difference of Arrival (TDoA)

TDoA benefit from relative time measurement for each receiving sensor. Therefore TDoA doesn't require a synchronized time source at the point of transmission as in the ToA systems. In TDoA systems only receivers need synchronization.

TDoA systems are implemented based on a mathematical concept known as hyperbolic lateration. In this method, at least three time-synchronized receiving sensors are needed. ToA and TDoA are similar. Both of these techniques proved to be successful for large-scale outdoor positioning systems. Figure 2.3 shows the hyperbols

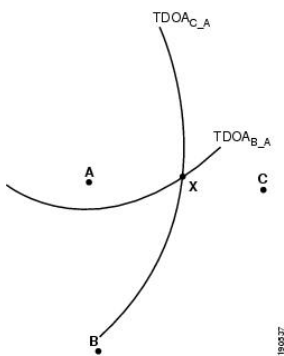


Figure 2.3: Time Difference of Arrival

Received Signal Strength (RSS)

Location can also be realized by using received signal strength (RSS) in place of time. RSS is calculated by either the mobile device or the receiving sensor. With this technique mobile device or receiving sensor measures the RSS. Transmitter output power, cable losses, antenna gains and path loss model allows to solve for the distance between two stations.

$$PL = PL_{1\text{Meter}} + 10 \log(d^n) + s$$

PL means path loss between the receiver and sender in dB

- o • $PL_{1\text{Meter}}$ means the reference path loss in dB for the desired frequency when the receiver-to-transmitter distance is 1 meter.
- d means the distance between the transmitter and receiver in meters.
- n means the path loss exponent for the environment.
- S shows the standard deviation related with the degree of shadow fading in dB.
- Path loss (PL) is the difference between the transmitted signal level and the received signal level. Path loss shows signal attenuation level present in the environment that is caused by free space propagation, reflection, diffraction and scattering.

RSS location techniques have a cost advantage because they don't require any specialized hardware at the mobile device or network infrastructure locations

Angle-Based (Angulation) Techniques

Angle of Arrival (AoA)

The Angle of Arrival is also called Direction of Arrival. This method benefits from the angle of incidence at which receiving sensor takes the signal. Geometric relationships are used to determine location by using the intersection of two lines of bearing(LoBs) which is formed by a radial line to each receiving sensor. At least two receiving sensors are necessary but three or more receiving sensors increase accuracy.

Directional antennas deployed at the receiving sensors are adjusted to the signal with highest signal strength. The positioning of the antennas determine the LoBs and measure the angles of incidence.

Multiple tower sites obtain the AoA of the cellular user's signal and use this info to perform tri-angulation. This info is converted to user location and latitude and longitude coordinates by the switching processors AoA works well with direct line of sight but its accuracy and precision decreases when confronted with signal reflections from objects. In practice this method requires expensive antenna arrays, which limit its feasibility despite its potential for high accuracy [9]. Figure 2.4 shows the angles.

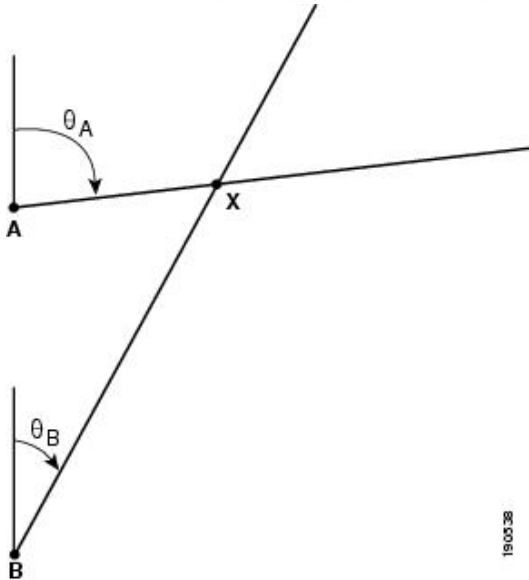


Figure 2.4: Angle of Arrival

Location Patterning (Pattern Recognition) Techniques

Location patterning basically samples and records radio signal behavior patterns in specific environments. A location patterning solution doesn't require specialized hardware. Location patterning can be implemented fully in software therefore it reduces cost and complexity comparing to angulation or lateration systems.

Location patterning techniques require two fundamental conditions

- Each potential device location possesses a distinct and unique RF "signature".
- Each floor or subsection has unique signal propagation features.

Generally location patterning solution benefit from received signal strength(RSSI), pattern recognition may benefit from ToA,AoA or TDoA-based RF signatures as well. Patterning based positioning system consist of two phases:

- Calibration phase
- Operation phase

During the calibration phase a database of RF signals are created. And during the operational phase the RF signature of the tracked device are matched with the database. The radio maps or calibration databases which is used by pattern recognition method are very specific to the areas used in their creation and it isn't suitable for re-use. The radio maps or calibration databases which is used by pattern recognition method are very specific to the areas used in their creation and it isn't suitable for re-use.

Okumura Model

The Okumura model is generally used in Urban Areas. It is a Radio propagation model. It is used for signal prediction. Its frequency range is 200Mhz-1900Mhz. Its distance range is 1km-100km. Antenna heights whose ranges 30m-300m is effective for base station. Okumura model has a 10-14db standard deviation. Its the deviation between the path loss model which is predicted by the model and the real measured path loss. Correlation factors related to terrain are calculated to improve the model accuracy.

Hata Model

The Hata model is an empirical formulation of data given by the Okumura. It is valid between 150-1500MHz. Hata model simplifies the path loss calculation since it has a closed form formula. Besides it doesn't use empirical curves for the different parameters.

Hata model doesn't use any specific path correlation factors. The Hata model gives more accurate results for distances $d > 1$ km. Hata models don't capture indoor environments. Hata model is good at first generation cellular systems but it doesn't model propagation well in new cellular systems with smaller sizes and high frequencies.

Okumura-Hata model (OH) [13] :

$$\text{path_loss} = 158.3 - 13.82 \log(H_{BT}) + (44.9 - 6.55 \log(H_{BT})) \log(d)$$

Where (H_{BT}) is BS antenna height. ΔH_b is difference between BS antenna height and MS height (1.5m) and (d) is the BS-MS distance[14].

CHAPTER 3

Overview of Neural Networks

In this chapter, artificial neural networks and training will be explained. This is the theory part of the project. Various neural network types are discussed. Besides, various learning methods are discussed. Furthermore, various training methods are discussed. The details of training, test, train validation sets and improving results are discussed for an optimal neural network training.

Artificial Neural Networks

In computer science artificial neural networks are computational models that resemble animals' biological central nervous system. Scientists who examined the central nervous systems inspired from them. Basically a class of statistical model which consist of set of adaptive weights, a learning algorithm with numerical parameters and which can approximate a function of their inputs can be called artificial neural network. Learning updates the weights. An activation function converts input to output. In artificial neural networks simple artificial nodes are called "neurons", "units" or processing elements. The adaptive weights are used in training and prediction phases. They are connection strengths between neurons.

Learning paradigms

There are three major learning paradigms.

1-Supervised learning

2-Unsupervised learning

3-Reinforcement learning

Supervised learning

In supervised learning we have to infer the mapping from a given function, using the implied data.

As the name suggests, we have prior knowledge from the labeled data about the problem domain.

Basically we have to find a function from a given set of example pairs(x,y) to find $f:X \rightarrow Y$ matching examples. Difference between our mapping and the data gives us the cost function.

Some of application areas of supervised learning are pattern recognition(classification) and regression(function approximation). We use previous solutions as feedback in supervised learning.

Training: Basically, neural network's function is to predict an output pattern when an input pattern is given. After a neural network is trained, it is able to recognize similarities when it is presented with a new input pattern. Therefore it results in a predicted output pattern.

Clustering: Clustering algorithm finds resemblances between patterns and puts similar patterns in its cluster.

Pattern recognition(Classification): When an input pattern is given, it is assigned to a class between some classes. This task is called pattern recognition.

Function approximation: Function approximation creates an estimate of the unknown function $f()$ subject to noise.

Prediction/Dynamical Systems: When a time-sequenced data is given, some future values are estimated. This task is called prediction. Prediction is used widely in decision support systems and datawarehouses. The difference between prediction and function approximation is time factor. Prediction is a dynamical system and gives different results for the same input data but different system state(time).

Learning rate: The learning rate is value between 0 and 1. Weight adjustments size is controlled by it. Learning process speed is affected by it. In addition, precision rate of network is affected by it.

Difference between supervised learning and unsupervised learning: In supervised training, inputs and outputs are given. The network uses the inputs and compares its results with the desired outputs. Errors are calculated and according to this, the system adjusts the weights which control the network. This process is repeated many times and weights are changed.

In unsupervised training the inputs are given to the network but outputs aren't given to the network. The system itself decides the features which is needed to group the input data. This is called self-organization(adaption). For example learning process is usually unsupervised.

Supervised Learning

In supervised learning a labeled training set is given. The class of inputs is provided and known.

Unsupervised Learning

In unsupervised learning a set of patterns are given from n-dimensional space. But no/little information about their classification and evaluation is given

Tasks:

Vector Quantization: N-dimensional space S is divided into a small set of regions.(It is also useful in clustering pattern sets.)

Feature extraction: Feature extraction reduces the dimensionality of n-dimensional space S by removing unimportant features that don't help clustering

Bias: Two different kind of parameters are modified during ANN training, the weights and the t values. "t" parameter is the amount of incoming pulses that is needed to activate a real neuron. This is not practical and it would be easier if only one parameter is modified. Bias neuron is invented to solve this problem.

Perceptrons

Perceptron has many definitions but one of the simplest is this “A single layer network which produces a correct target vector when presented with the input vector.” [14]. This single layer does this training by changing the weights and biases of the network.

The training technique for the perceptrons is perceptron learning rule.

The perceptron contains a single layer. It is connected to R inputs with a set of weights. A perceptron is created with the newp function. `net = newp(P,T)`

The network includes zero weights and biases. If you want different weights and biases with values other than zero, you have to create them with command manually.

Set the two weights and the one bias to -1, 1, and 1

```
net.IW{1,1}=[-1 1]; net.b{1}=[1];
```

Now use `init` to reset the weights and bias to their original values:

```
net = init(net);
```

A learning rule means a procedure to modify the weights and biases of a network. It is also called training. Learning rule is applied to network to do a specific task.

If `epochs` is set to 1, `train` goes through the input vectors 1 time.

To set the parameter

```
time. net.trainParam.epochs = 1;
```

```
net = train(net,p,t);
```

The outputs aren't equal to the targets yet, so the network needs to be trained for more than one pass. More epochs are needed to find more accurate results.

```
net.trainParam.epochs = 1000;
```

The default training function for networks created with `newp` is `trainc`. This fact can be verified by executing `net.trainFcn` (You can find this by executing `net.trainFcn`.)

Perceptron networks are generally trained with adapt function. Adapt function presents the input vectors to the network one at a time. After that corrections and adjustments are made to the network by the result of each input vector presentation. In this way, it is guaranteed that any linearly separable problem is solved in finite steps training presentations.

Backpropagation

Back propagation is the most widely used algorithm for supervised learning with multi-layered feed-forward networks[15]. The basic idea of the back propagation learning algorithm [16] is the repeated application of the chain rule to compute the influence of each weight in the network with respect to an arbitrary error function E:

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial s_i} \frac{\partial s_i}{\partial net_i} \frac{\partial net_i}{\partial w_{ij}}$$

w_{ij} represents the weight from neuron j to neuron i , s_i is the output, and net_i represents the weighted sum of the inputs of neuron i .

Once the partial derivative for each weight is known, the aim of minimizing the error function is achieved by performing a simple gradient descent:

$$w_{ij}(t + 1) = w_{ij}(t) - t \frac{\partial E}{\partial w_{ij}}(t)$$

Backpropagation means backward propagation of errors. This is a training method used with gradient descent optimization method. The aim of the method is to calculate the gradient of a loss function according to the weights in the network. The gradient is used by the optimization method and optimization method uses it to update the weights.

Typically a network is given input and output. This way network learns. And if a new input is given it will produce output that is similar to the learnt correct output.

Neurons use a differentiable transfer function f to generate their output. Firstly, a feedforward network is created. The third argument is an array containing the sizes of each hidden layer.

In matlab `net = newff(houseInputs,houseTargets,20);` does this job.

While two-layer feed-forward networks can be trained and can learn almost any input-output relationship, feed-forward networks with more layers might learn complex relationships more quickly.

The weights and biases of the network are adjusted so that the error is minimized. This performance error function is `net.performFcn` in matlab. If any performance function isn't specified, default function for it is mean square error (The average squared error between the network outputs a and the target outputs t) "mse" in matlab

During training the weights and biases of the network are iteratively adjusted to minimize the network performance function `net.performFcn`. The default performance function for feedforward networks is mean square error mse—the average squared error between the network outputs a and the target outputs t . `net = newff(p,t,3,{'traingd'})`;

More optional arguments can be provided for the feedforward backpropagation method in the matlab. For instance, the fourth argument represents a cell array which contains the names of the transfer functions to be used in each layer. The fifth argument represents the name of the training function to be used. If only three arguments are supplied, the default transfer function of hidden layers is `tansig` and the default for the output layer is `purelin`. The default training function is `trainlm`.

```
net = newff(p,t,3,{'trainrp'});
```

```
net = newff(houseInputs,houseTargets,20);
```

```
net = train(net,p,t);
```

Training,Validation,Test Datasets

In multilayer networks, firstly the data is divided into three subsets. The first subset is training subset. In training subset the gradient is computed and the network weights and biases are updated.

The second subset is the validation subset. The error on the validation set decreases during training phase with training set error. When network overfits the data,

validation set error starts to rise. At minimum validation set error the network weights and biases are saved.

The test set error is used for comparing different models. It is used when plotting the test set error during training.

In matlab four functions are used to divide data into training, validation and test sets. These are `dividerand`(the default), `divideblock`, `divideint` and `divideind`.

Training Set

The training dataset trains or builds a model. For instance, in a linear regression model the training dataset fits the linear regression model. Besides it is used to obtain regression coefficients. The training dataset finds the network weights in a neural network model.

Validation Set

After a model is built based on the training data, the accuracy of the model on unseen data is needed. Therefore data needs to be used on a dataset that wasn't used in the training process(This dataset has the actual value of the target variable) The small difference between the actual value and the predicted value of the target variable is the error in prediction. Some form of average error(for example MSE) measures the overall accuracy.

The training data itself can't be used to compute the accuracy of model fit because model fit process ensures the training data is very accurate and therefore very optimistic estimates are obtained. So a part of the original data needs to be partitioned for realistic estimate with unseen data. This dataset is called validation dataset. After the model is fit on the training dataset, its performance on the validation dataset is measured.

Test Set

The validation dataset is used in fine-tuning of models. It can be used in various architectures. When finally a model is chosen after comparisons of architectures it may still give optimistic estimates because the final model is winner among the other models based on the validation dataset accuracy.

A portion of the original data that is neither used in training nor in the validation phase is set aside. This dataset is called test dataset.

The most realistic performance estimate of the model on completely unseen data is on the test data set.

Improving Results

After training the network, if it isn't accurate enough, the network can be initialized and it can be trained again. Each time a feedforward network is initialized, the network parameters change and it may produce different solutions.

This can be done in matlab by using init command.

```
net = init(net);
```

```
net = train(net,houseInputs,houseTargets);
```

Secondly ,the hidden neuron number can be increased above 20. Larger neuron number gives the network more flexibility. Because the network can optimize more parameters. However too large hidden layers can optimize more parameters than data vectors which constrain these parameters.

Thirdly, a different training function can be used. For example Bayesian regularization training with trainbr sometimes produces better results than early stopping.

Eventually, using additional training data is good for better training. Feeding network with additional data produces a network that generalizes better to new data.[17]

Posttraining Analysis (regression)

The R value indicates the relationship between the outputs and targets. If $R=1$, this is the indication that outputs and targets have exact linear relationships. If R is close to zero this means there is no linear relationships between outputs and targets.

After training the network the performance can be measured by the errors on the training, validation and test sets. However sometimes investigating the network response in more detail is needed. A good approach is a regression analysis between the network outputs and the corresponding targets. The regression method is designed to perform this analysis.



Training Algorithm		Problem Solution Class	Network Size
Levenberg-Marquardt	trainlm	function approximation	medium networks

BFGS Quasi-Newton	trainbfg	function approximation	medium networks
Resilient Backpropagation	trainrp	pattern recognition	large networks
Scaled Conjugate Gradient	trainscg	pattern recognition,function approximation	large networks
Conjugate Gradient with Powell/Beale Restarts	traincgb	pattern recognition,function approximation	large networks
Fletcher-Powell Conjugate Gradient	traincgf	pattern recognition,function approximation	large networks
Polak-Ribière Conjugate Gradient	traincgp	pattern recognition,function approximation	large networks
One Step Secant	trainoss	function approximation	large networks
Variable Learning Rate Backpropagation	traingdx	pattern recognition,function approximation	medium networks

Table 3.1: Training algorithms comparison

Table 3.1 compares various training algorithms. Each training algorithm is good at solving its specific problem type or types. Besides, abbreviations of training algorithms are given. Furthermore, each training algorithm is good at specific network size. These attributes are listed and compared at the table.

Training Algorithm		Performance	Storage	Computation	Special
--------------------	--	-------------	---------	-------------	---------

Levenberg-Marquardt	trainlm	Very High	High	Cheap	Jacobian matrix(easier than Hessian)
BFGS Quasi-Newton	trainbfg	High	High	Expensive	Second derivative(Hessian matrix)
Resilient Backpropagation	trainrp	Medium	Medium		weights change by derivative sign
Scaled Conjugate Gradient	trainscg	High	Low	Cheap	No line search
Conjugate Gradient with Powell/Beale Restarts	traincgb	High	Medium	Medium	Less reset points for the direction of gradient
Fletcher-Powell Conjugate Gradient	traincgf	High	Low	Medium	conjugate search direction
Polak-Ribière Conjugate Gradient	traincgp	High	Low	Medium	conjugate search direction
One Step Secant	trainoss	Medium	Medium	Medium	Compromise function between BFGS and conjugate gradient
Variable Learning Rate Backpropagation	traingdx	Medium	Medium	Cheap	adaptive learning rate

Table 3.2: Training algorithms performances

Table 3.2 lists and compares various training algorithms according to their performance, storage, computation and its special training method. Besides, abbreviations of training algorithms are given. Performance, storage and computation are important attributes in training. A training algorithm can be good in one attribute but it can be medium or poor in other attributes. User can decide a training algorithm according to his needs. “Special” column is specific method to related training algorithm.

CHAPTER 4

DATA COLLECTION

In the project, firstly raw data was studied. This raw data contained a lot of redundant information. Meaningful patterns had to be specified and an algorithm for this data had to be made. This algorithm aims to extract meaningful information from raw data. After that java parser code implements and realizes this solution. After that this meaningful data is analyzed in excel and access. Necessary normalizations are made and a training data is created.

Secondly, artificial data was studied. Artificial data is created by Dr. Selçuk Öğrenci. Real data is simulated and noise is added therefore artificial data is created. This simulated data was similar to real data. This data contains user location coordinates and rscp values. Rscp values indicate the signal strength between phone and base station. Experiments are made with this data on Matlab program and java encog library. Encog library is an open source machine learning library that contains training methods.

Finally, real data that comes from user mobile phones was studied. This data is collected by a mobile program. This program starts to collect user data as user presses “start” button. It finishes collecting data when the user presses “finish” button. A person has to go around certain locations to collect the user data at this location. This data was gathered by Dr. Selçuk Öğrenci. He wandered around certain locations with mobile phone and gathered user data. Later, programmer Ramazan Cengiz gathered user data by going around certain locations for additional data. This is real data and outputs are real values and locations. Experiments were made with this data on Matlab and Encog. This data contained data from 60-70 base stations. It contained tens of thousands of lines of data. Data was reduced to 20-30 base stations and data lines were reduced to hundreds of lines. This was realized by queries and analysis on excel and access. Finally, 21 training and 5 test sets were created.

Database code

In the project information for each cell was required. This information was concerned about cell normalization. This info has to show the estimated cell identity according to three normalization types(Gravity center coordinate,Geometric center coordinate, Minimum left point coordinate). In addition normalized coordinates(latitude,longitude values) has to be shown in the information document.

In the gravity center normalization gravity center of the cell is accepted as cell center and other points' relative distance to center is calculated as its normalized value. In the geometric center normalization, geometric center of the cell is accepted as base station and other points' relative distance to center is calculated as its normalized value. In the min normalization, minimum coordinates of the cell is accepted as base station and other points' relative distance to center is calculated as its normalize value.

So, a cell id and all coordinate points belonging to this cell id needs to be collected. After these values are collected, gravity center of it is calculated,then geometric point of the cell and min,max points of the cell are calculated. After that process each cell id(base staion) has its own geometric center,gravity center and min,max points. Later on, the coordinates(longitude,latitude values) that belong to specific cells are normalized(the distance to center is calculated) so that they have new normalized values. This process makes the job easier in the training process.

ALGORITHM

1- Main data table named “tabela” is imported into Access database. This table contains “OMA specified longitude,latitude values” and their distances in km to (0,0)point in the map. X distances, Y distances and direct distances are held in this table.

2- By running query1 on table “tabela” ,a table named “QTABLO1” is formed. This table contains detailed data. Each cell’s longitude and latitude values are held in this table.

3- By running query2 on table “tabela” ,a table named “QTABLO2” is formed. This table contains summary main data. Each cell’s avg,min,max,geo values are held in this table

4- By running NormalizeQuery each longitude,latitude with its normalized values are obtained. This query includes a relationship between QTABLO1 and QTABLO2. By using inner join ” FROM QTABLO22 INNER JOIN QTABLO11 ON QTABLO22.UC=QTABLO11.UC;” Foreign key,primary key relationship is done.

NormalizeQuery

```
SELECT QTABLO22.UC AS QTABLO22_UC,  
QTABLO11.LATIY AS LATI,  
LATI-QTABLO22.avglat AS ["normlatavg"],  
LATI-QTABLO22.geolat AS ["normlatgeo"],  
LATI-QTABLO22.minlat AS ["normlatmin"],  
QTABLO22.minlat, QTABLO22.avglat,  
QTABLO22.geolat, QTABLO22.maxlat,  
QTABLO11.LONGIX AS LONGI,  
LONGI-QTABLO22.avglon AS ["normlonavg"],  
LONGI-QTABLO22.geolon AS ["normlongeo"],  
LONGI-QTABLO22.minlon AS ["normlntmin"],  
QTABLO22.minlon,  
QTABLO22.avglon,  
QTABLO22.geolon,  
QTABLO22.maxlon  
FROM QTABLO22 INNER JOIN QTABLO11 ON QTABLO22.UC=QTABLO11.UC;
```

QUERY1(Detail table)

```
SELECT tabela.UC,  
tabela.LATIY,  
tabela.LONGIX,  
Count(tabela.UC) AS totalcell,  
Min(tabela.LATIY) AS minlat,  
Avg(tabela.LATIY) AS avglat,  
(minlat+maxlat)/2 AS geolat,  
Max(tabela.LATIY) AS maxlat,  
Min(tabela.LONGIX) AS minlon,  
Avg(tabela.LONGIX) AS avglon,  
((minlon+maxlon)/2) AS geolon,  
Max(tabela.LONGIX) AS maxlon INTO QTABLO11  
FROM tabela  
WHERE (((tabela.UC) Is Not Null) AND ((tabela.LATITUDE) Is Not Null) AND ((tabela.LATITUDE) Is Not Null))  
GROUP BY tabela.UC, tabela.LATIY, tabela.LONGIX;
```

QUERY2(Main table)

```
SELECT tabela.UC, Count(tabela.UC) AS totalcell,  
Min(tabela.LATIY) AS minlat,  
Round(Avg(tabela.LATIY),2) AS avglat,  
(minlat+maxlat)/2 AS geolat,  
Max(tabela.LATIY) AS maxlat,  
Min(tabela.LONGIX) AS minlon,  
Round(Avg(tabela.LONGIX),2) AS avglon,  
((minlon+maxlon)/2) AS geolon,  
Max(tabela.LONGIX) AS maxlon INTO QTABLO22  
FROM tabela  
WHERE (((tabela.UC) Is Not Null) AND ((tabela.LATITUDE) Is Not Null) AND ((tabela.LONGITUDE) Is Not Null))
```

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
QTABLO LATI	*normalavg	*normalat	*normalat	miniat	avglat	geolat	maxlat	LONGI	*normalavg	*normalgeo	*normalmin	minlon	avglon	geolon	maxlon	
1	65623982	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
2	65623982	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
3	6562323	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
4	65629238	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
5	65633014	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
6	6563358	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
7	65657632	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
8	65787440	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
9	65799284	4550.168	-0.002413197	0	4550.168	4550.168	4550.168	2936.002	0.002203162	0	0	2936.002	2936	2936.002	2936.002	
10	65801736	4550.263	0.002858868	0	4550.263	4550.263	4550.263	2937.377	-0.003124366	0	0	2937.377	2937.38	2937.377	2937.377	
11	65804829	4550.263	0.002858868	0	4550.263	4550.263	4550.263	2937.377	-0.003124366	0	0	2937.377	2937.38	2937.377	2937.377	
12	65805084	4551.085	0.004580421	0	4551.085	4551.085	4551.085	2936.48	-0.000480594	0	0	2936.48	2936.48	2936.48	2936.48	
13	65805085	4551.085	0.004580421	0	4551.085	4551.085	4551.085	2936.48	-0.000480594	0	0	2936.48	2936.48	2936.48	2936.48	
14	65805137	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
15	65806692	0	-1516.83	-2275.25	0	1516.83	2275.248	4550.496	-978.92	-1468.373378	0	0	978.92	1468.379	2936.759	
16	65806692	4550.496	3033.666275	2275.248	4547.714	4547.714	4547.714	2941.675	1957.838757	1468.373378	2936.758757	0	978.92	1468.379	2936.759	
17	65812290	4547.714	0.00433115	0	4547.714	4547.714	4547.714	2941.675	0.004870893	0	0	2941.675	2941.67	2941.675	2941.675	
18	65813232	0	-2274.92	-2274.92	0	2274.92	2274.919	4549.838	-1467.93	-1467.933949	0	0	1467.93	1467.934	2935.868	
19	65813232	4549.838	2274.917707	2274.919	4549.838	4549.838	4549.838	2935.868	1467.937897	1467.933949	2935.867897	0	1467.93	1467.934	2935.868	
20	65813866	4551.012	2275.501935	2275.506	4551.012	4551.012	4551.012	2936.526	1468.266098	1468.263049	2936.526098	0	1468.26	1468.263	2936.526	
21	65813866	0	-2275.51	-2275.51	0	2275.51	2275.506	4551.012	-1468.26	-1468.263049	0	0	1468.26	1468.263	2936.526	
22	65819481	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
23	65819726	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
24	65819927	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
25	65819938	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
26	65820421	4547.667	-0.023304882	-0.02382	4547.667	4547.667	4547.667	2941.784	0.054489388	0.054809252	0.109618505	2941.675	2941.73	2941.73	2941.784	
27	65820421	4547.714	0.02433115	0.023818	4547.667	4547.667	4547.667	2941.675	-0.05129107	-0.054809252	0	2941.675	2941.73	2941.73	2941.784	
28	65820514	0	-1517.02	-2275.52	0	1517.02	2275.523	4551.045	-978.95	-1468.276816	0	0	978.95	1468.277	2936.554	
29	65820514	4551.045	3034.025281	2275.523	4548.093	4548.093	4548.093	2941.652	1957.703633	1468.276816	2936.553633	0	978.95	1468.277	2936.554	
30	65820527	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
31	65820529	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
32	65820533	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
33	65821729	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
34	65822276	4548.093	0.003037605	0	4548.093	4548.093	4548.093	2941.652	0.002148944	0	0	2941.652	2941.65	2941.652	2941.652	
35	6582340	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
36	65823257	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
37	65823285	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
38	65823386	4550.263	0.002858868	0	4550.263	4550.263	4550.263	2937.377	-0.003124366	0	0	2937.377	2937.38	2937.377	2937.377	
39	65823469	4549.878	-0.001802092	0	4549.878	4549.878	4549.878	2935.817	-0.002784671	0	0	2935.817	2935.82	2935.817	2935.817	
40	65823304	4551.085	0.004580421	0	4551.085	4551.085	4551.085	2936.48	-0.000480594	0	0	2936.48	2936.48	2936.48	2936.48	
41	65825162	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
42	65825786	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
43	65825794	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
44	65825796	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
45	65825797	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
46	65825801	0	-455.15	-2275.76	4551.519	4551.519	4551.519	2940.179	-294.02	-1470.089621	0	0	294.02	1470.09	2940.179	
47	65825801	4551.519	4096.369259	2275.76	4551.519	4551.519	4551.519	2940.179	2646.158242	1470.089621	2940.179242	0	294.02	1470.09	2940.179	
48	65826228	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Figure 4.1 : Database queries of data

MOBILE PHONE DATA COLLECT SOURCE CODE

```
public void run()                //Create TelephonyManager Object
                                //GsmCellLocation gsmLoc = (GsmCellLocation)
tm.getCellLocation();

                                //Log.d("ramco", gsmLoc.getCid() + "");
List<CellInfo> gcis = tm.getAllCellInfo();
    File dir = Environment.getExternalStorageDirectory();
    Calendar calendar = Calendar.getInstance();
    SimpleDateFormat sdf = new SimpleDateFormat("dd-MM-yyyy");
        if(gcis != null){
            for(CellInfo ci : gcis){
                String logStr = "";
                if(ci instanceof CellInfoGsm){
                    //Log.d("GSM", "GSM_INFO");
                    //Log to GSM file
                    fileName = "gsmData_"+sdf.format(calendar.getTime())+".csv";
                    File logFile = new File(dir, fileName);
                    CellInfoGsm cgi = (CellInfoGsm)ci;
                    int rssi = cgi.getCellSignalStrength().getDbm();
                    long timestamp = cgi.getTimeStamp();
                    int lac = cgi.getCellIdentity().getLac();
                    int cid = cgi.getCellIdentity().getCid();
                    int mnc = cgi.getCellIdentity().getMnc();
                    int mcc = cgi.getCellIdentity().getMcc();
```

Above, mobile phone method code can be seen. Its "run" method is seen. Using the related library, cgi object is created. In cgi object, all necessary info about cell location is stored. These info is extracted and put into variables by using related methods such as getMNC, getCid, getMCC.

```

super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    //initialize & instantiate objects
    state = false;
    startBtn = (Button) findViewById(R.id.startBtn);
    stopBtn = (Button) findViewById(R.id.stopBtn);
    secText = (EditText) findViewById(R.id.sec);

    locationManager = (LocationManager) getSystemService(LOCATION_SERVICE);
    locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0, 0, this);

    tm = (TelephonyManager) getSystemService(TELEPHONY_SERVICE);
    startBtn.setOnClickListener(new OnClickListener() {

        @Override
        public void onClick(View arg0) {
            int ms;
            timer = new Timer();
            if(secText.getText() == null ||
secText.getText().toString().equalsIgnoreCase(""))
                ms = 3000;

```

Above, mobile phone code is seen. In this method GUI of the program is designed. GUI items, buttons and texts are assigned to program variables. Besides a timer object is created to specify the data collection time.

Raw Data

	A	B	C	D	E	F	G	H	I
1		LAC	MNC	MCC	Cell Id	Longitude	Latitude	Rssi	Pr Scramble
2	22964051713528,00	10321	2	286	29716	28,63986	41,00399	-91	409
3	22967084065122,00	10321	2	286	29716	28,63982	41,00416	-81	409
4	22970004620592,00	10321	2	286	29716	28,63982	41,00421	-83	409
5	22973012830572,00	10321	2	286	29716	28,63982	41,00427	-83	409
6	22976004498510,00	10321	2	286	29716	28,63986	41,0043	-83	409
7	22979013227336,00	10321	2	286	29716	28,63988	41,00434	-79	409
8	22982013440979,00	10321	2	286	29716	28,63991	41,00437	-75	409
9	22985013685141,00	10321	2	286	29716	28,63994	41,00442	-79	409
10	22988013898783,00	10321	2	286	29716	28,63996	41,00447	-81	409
11	22991014112426,00	10321	2	286	29716	28,63999	41,00451	-77	409
12	22994005810886,00	10321	2	286	29716	28,64002	41,00455	-83	409
13	22997006146608,00	10321	2	286	29716	28,64005	41,00459	-89	409
14	23000014356587,00	10321	2	286	29716	28,64008	41,00463	-79	409
15	23003126579996,00	10321	2	286	29716	28,64008	41,00466	-77	409
16	23006142816833,00	10321	2	286	29716	28,6401	41,00469	-87	409
17	23009007489503,00	10321	2	286	11908	28,64012	41,00473	-91	409
18	23012015668964,00	10321	2	286	11908	28,64013	41,00476	-83	409
19	23015007886267,00	10321	2	286	11908	28,64013	41,0048	-87	409
20	23018016065726,00	10321	2	286	11908	28,64014	41,00485	-99	409
21	23021016370931,00	10321	2	286	11908	28,64016	41,00489	-87	409
22	23024016554052,00	10321	2	286	31534	28,6402	41,00493	-91	106

Figure 4.2: Raw data

This raw data is unprocessed data that is collected by the mobile phone. It is shown at figure 4.2. This data needs to be adjusted in order to be used by the program. Redundant data should be eliminated. Besides, necessary base locations need to be added for training. And normalized values have to be computed .

Processed Data

N2														base x	
	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1							3219753	4554084					average x	average y	
2	Longitude	Latitude	Rssi	Prsc	Basex	Basey	x	y	xt	yt	interval x	interval y	base x	base y	
3	28,95666	41,02783	-97	201	28,95563	41,02658	3219835	4554089	82,69683	4,896765	-165,394	-31,4086	3219721	4553951	
4	28,95517	41,02754	-83	201	28,95563	41,02658	3219670	4554057	-82,6968	-26,5118	1,20869	0,44733	-31,4849	133,1051	
5	28,95518	41,02755	-83	201	28,95563	41,02658	3219671	4554058	-81,4881	-26,0645	0	0			
6	28,95518	41,02755	-83	201	28,95563	41,02658	3219671	4554058	-81,4881	-26,0645	2,242803	0,82029			
7	28,9552	41,02756	-81	201	28,95563	41,02658	3219673	4554059	-79,2453	-25,2442	1,061912	0,47175			
8	28,95521	41,02756	-81	201	28,95563	41,02658	3219674	4554059	-78,1834	-24,7724	1,100831	0,40293			
9	28,95522	41,02756	-75	201	28,95563	41,02658	3219676	4554060	-77,0826	-24,3695	0,403638	0,79254			
10	28,95522	41,02757	-75	201	28,95563	41,02658	3219676	4554060	-76,679	-23,577	0,331361	0,43068			
11	28,95523	41,02757	-83	201	28,95563	41,02658	3219676	4554061	-76,3476	-23,1463	0	0			
12	28,95523	41,02757	-83	201	28,95563	41,02658	3219676	4554061	-76,3476	-23,1463	0,309122	0,4107			
13	28,95523	41,02758	-83	201	28,95563	41,02658	3219677	4554061	-76,0385	-22,7356	0,101187	0,13209			
14	28,95523	41,02758	-83	201	28,95563	41,02658	3219677	4554061	-75,9373	-22,6035	0	0			
15	28,95523	41,02758	-83	201	28,95563	41,02658	3219677	4554061	-75,9373	-22,6035	0,097852	0,2331			
16	28,95523	41,02758	-83	201	28,95563	41,02658	3219677	4554062	-75,8394	-22,3704	0,027799	0,15207			
17	28,95523	41,02758	-83	201	28,95563	41,02658	3219677	4554062	-75,8116	-22,2183	0	0			
18	28,95523	41,02758	-83	201	28,95563	41,02658	3219677	4554062	-75,8116	-22,2183	0,066717	0,4107			
19	28,95523	41,02759	-79	201	28,95563	41,02658	3219677	4554062	-75,7449	-21,8076	0,054486	0,14874			
20	28,95523	41,02759	-79	201	28,95563	41,02658	3219677	4554062	-75,6904	-21,6589	0	0			

Figure 4.3: Processed data

This data is intermediate processed data. It is shown at figure 4.3. Average user x and average user y locations need to be found. According to these average values, normalization is made. This normalized values are small and more suitable for training. In addition, base x and base y locations need to be found. This is necessary for supervised learning and training data. Interval x and interval y helps us to eliminate redundant and repeated data. 0 means the distance values are the same and redundant and isn't needed for training.

Final Train Data

	A	B	C	D	E	F	G	H
1	Cell Id	x	y	rsi	base x	base y	average x	average y
2	35610,00	0,00	0,01	-83,00	-0,08	0,11	3220196,55	4553649,83
3	35610,00	0,00	0,01	-83,00	-0,08	0,11	3220196,55	4553649,83
4	35610,00	0,00	0,02	-85,00	-0,08	0,11	3220196,55	4553649,83
5	35610,00	-0,01	0,02	-87,00	-0,08	0,11	3220196,55	4553649,83
6	35610,00	-0,01	0,02	-81,00	-0,08	0,11	3220196,55	4553649,83
7	35610,00	-0,01	0,03	-87,00	-0,08	0,11	3220196,55	4553649,83
8	35610,00	-0,02	0,03	-81,00	-0,08	0,11	3220196,55	4553649,83
9	35610,00	-0,02	0,03	-85,00	-0,08	0,11	3220196,55	4553649,83
10	35610,00	-0,02	0,03	-81,00	-0,08	0,11	3220196,55	4553649,83
11	35610,00	-0,02	0,03	-87,00	-0,08	0,11	3220196,55	4553649,83
12	35610,00	0,02	-0,04	-93,00	-0,08	0,11	3220196,55	4553649,83
13	35610,00	0,02	-0,04	-93,00	-0,08	0,11	3220196,55	4553649,83
14	35610,00	0,02	-0,04	-75,00	-0,08	0,11	3220196,55	4553649,83
15	35610,00	0,02	-0,04	-75,00	-0,08	0,11	3220196,55	4553649,83
16	35610,00	0,02	-0,04	-77,00	-0,08	0,11	3220196,55	4553649,83
17	35610,00	0,02	-0,04	-77,00	-0,08	0,11	3220196,55	4553649,83
18	35610,00	0,02	-0,04	-73,00	-0,08	0,11	3220196,55	4553649,83
19	35610,00	0,02	-0,04	-73,00	-0,08	0,11	3220196,55	4553649,83
20	35610,00	0,02	-0,04	-75,00	-0,08	0,11	3220196,55	4553649,83
21	35610,00	0,02	-0,04	-75,00	-0,08	0,11	3220196,55	4553649,83
22	35610,00	0,02	-0,04	-79,00	-0,08	0,11	3220196,55	4553649,83

Figure 4.4: Final Train Data

This is final train data. Only useful data is stored in the file. It is shown at figure 4.4. This final data is ready for use for the program. The values found in the processed data are divided by 1000 to evaluate in km and to be run in java. Normalized x and y distances of the users are found. Average x and average y data is used to find denormalized locations.

User Locations and Base Stations Distributions

Below, there are tables. These tables show the cell area and shape. Information about user x coordinates distribution, user y distribution, cell type (small or large) and the ratio of x and y are shown. The Cellids are also listed. There are 4 cell types. Their distribution determines their type. If distribution is in a small area, it is small, if user coordinates are distributed around a large area its type is large and so on.

Cells	CellId	dx	dy	alan	tip	dx/dy
Cell21	40143	0,21	0,13	0,0273	small	1,6
Cell26	53665	0,17	0,18	0,0306	small	0,9
Cell14	25627	0,27	0,13	0,0351	small	2,1
Cell6	29515	0,29	0,15	0,0435	small	1,9
Cell10	35610	0,29	0,16	0,0464	small	1,8
Cell15	29554	0,3	0,16	0,048	small	1,9
Cell23	40149	0,24	0,21	0,0504	small	1,1
Cell11	37442	0,78	0,07	0,0546	thin	11,1
Cell1	6700	0,21	0,33	0,0693	small	0,6
Cell7	32430	0,21	0,33	0,0693	small	0,6
Cell22	40144	0,29	0,25	0,0725	small	1,2
Cell8	32571	0,42	0,21	0,0882	small	2,0
Cell17	32431	0,98	0,09	0,0882	thin	10,9
Cell5	26297	0,37	0,27	0,0999	small	1,4

Cell25	43079	0,34	0,47	0,1598	mid	0,7
Cell19	37443	0,52	0,32	0,1664	mid	1,6
Cell13	25625	0,58	0,34	0,1972	mid	1,7
Cell12	6701	0,5	0,41	0,205	mid	1,2
Cell16	29555	0,47	0,57	0,2679	mid	0,8
Cell3	17611	1,1	0,27	0,297	large	4,1
Cell4	25617	0,86	0,37	0,3182	large	2,3
Cell24	43078	0,7	0,46	0,322	large	1,5
Cell2	11749	1,11	0,42	0,4662	large	2,6
Cell18	35053	0,86	0,7	0,602	large	1,2
Cell9	35050	0,84	0,94	0,7896	large	0,9

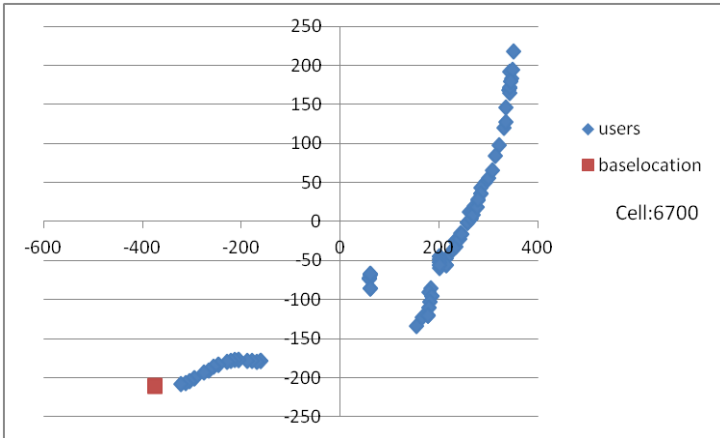


Figure 4.5:Cell 6700 map

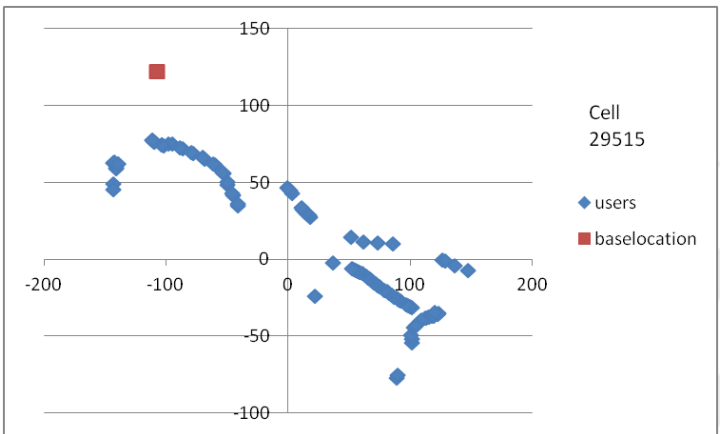


Figure 4.6:Cell 29515 map

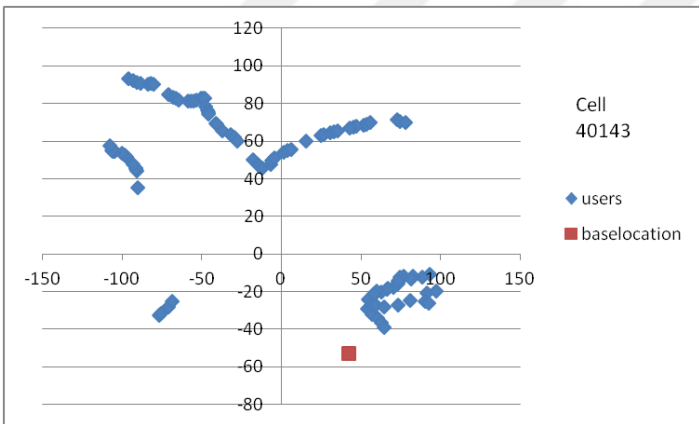


Figure 4.7:Cell 40143 map

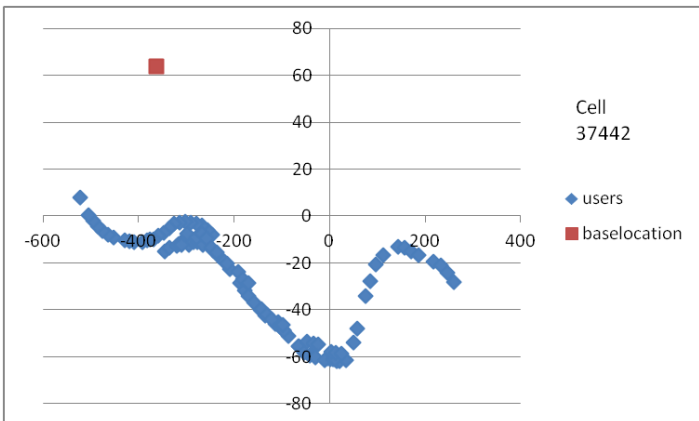


Figure 4.8: Cell 37442 map

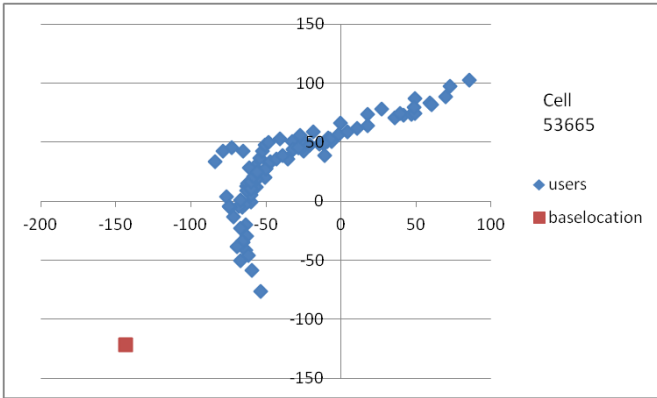


Figure 4.9: Cell 53665 map

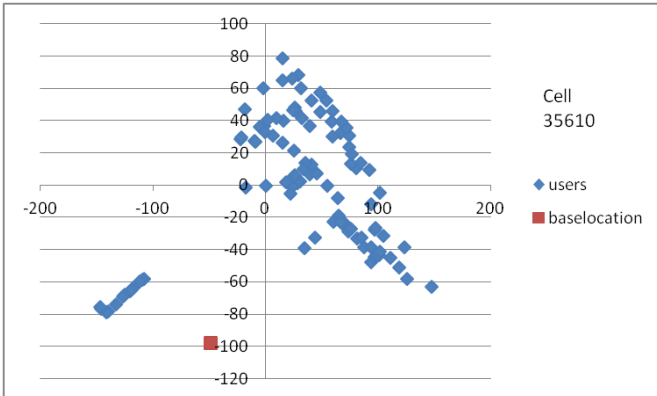


Figure 4.10: Cell 35610 map

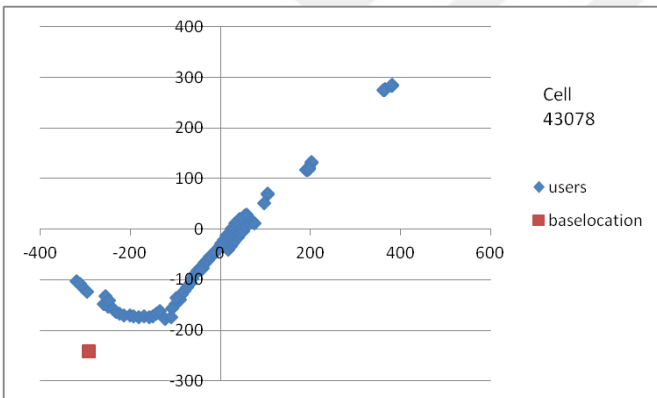


Figure 4.11: Cell 43078 map

Above in 4.5 - 4.11 figures user x-y and base station x-y distributions are listed and shown on the map. Red rectangle sign shows base station and kites are distributed user locations.

CHAPTER 5

Algorithms and Source Code

Project Flowchart

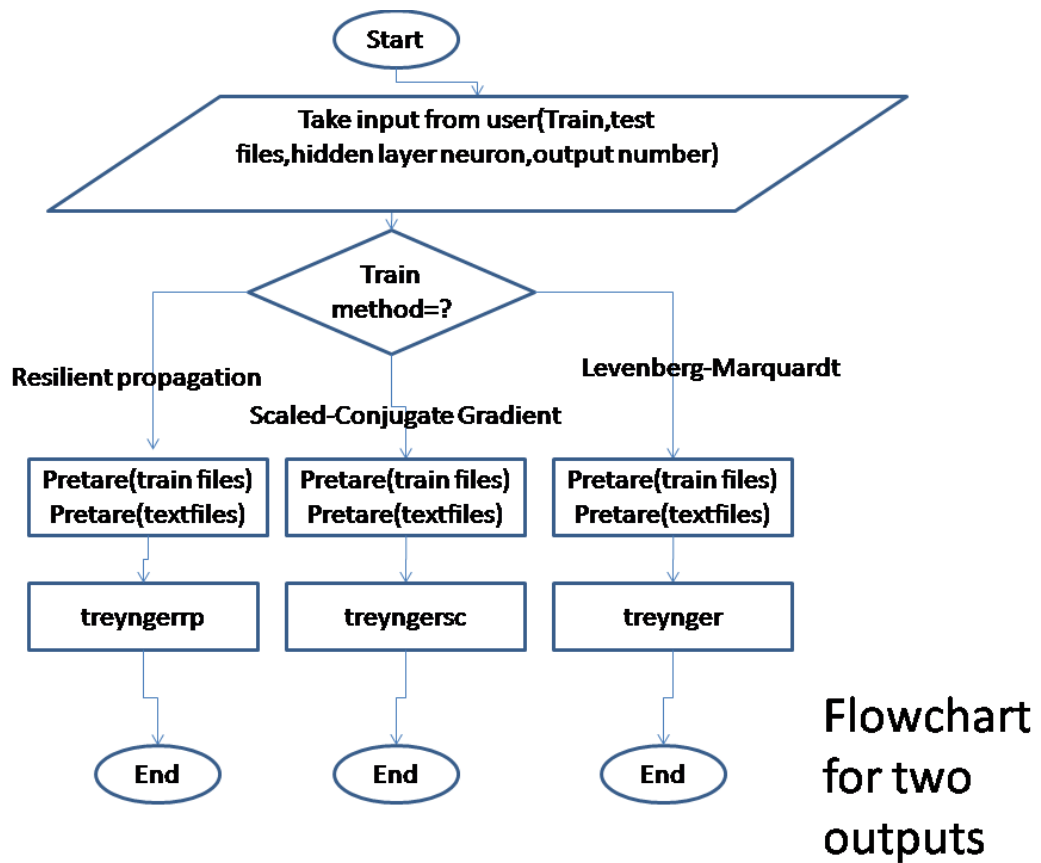


Figure 5.1: Flowchart for two outputs

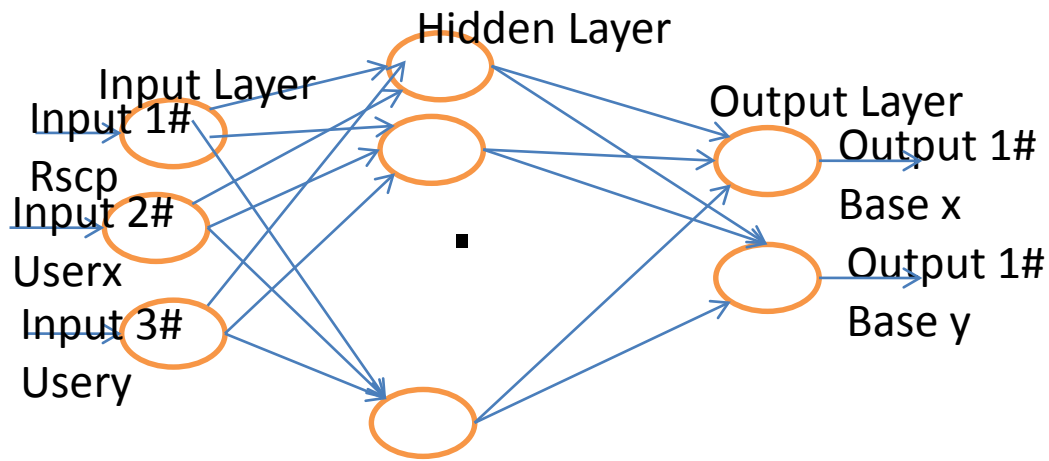


Figure 5.2: Neural network topology for two outputs

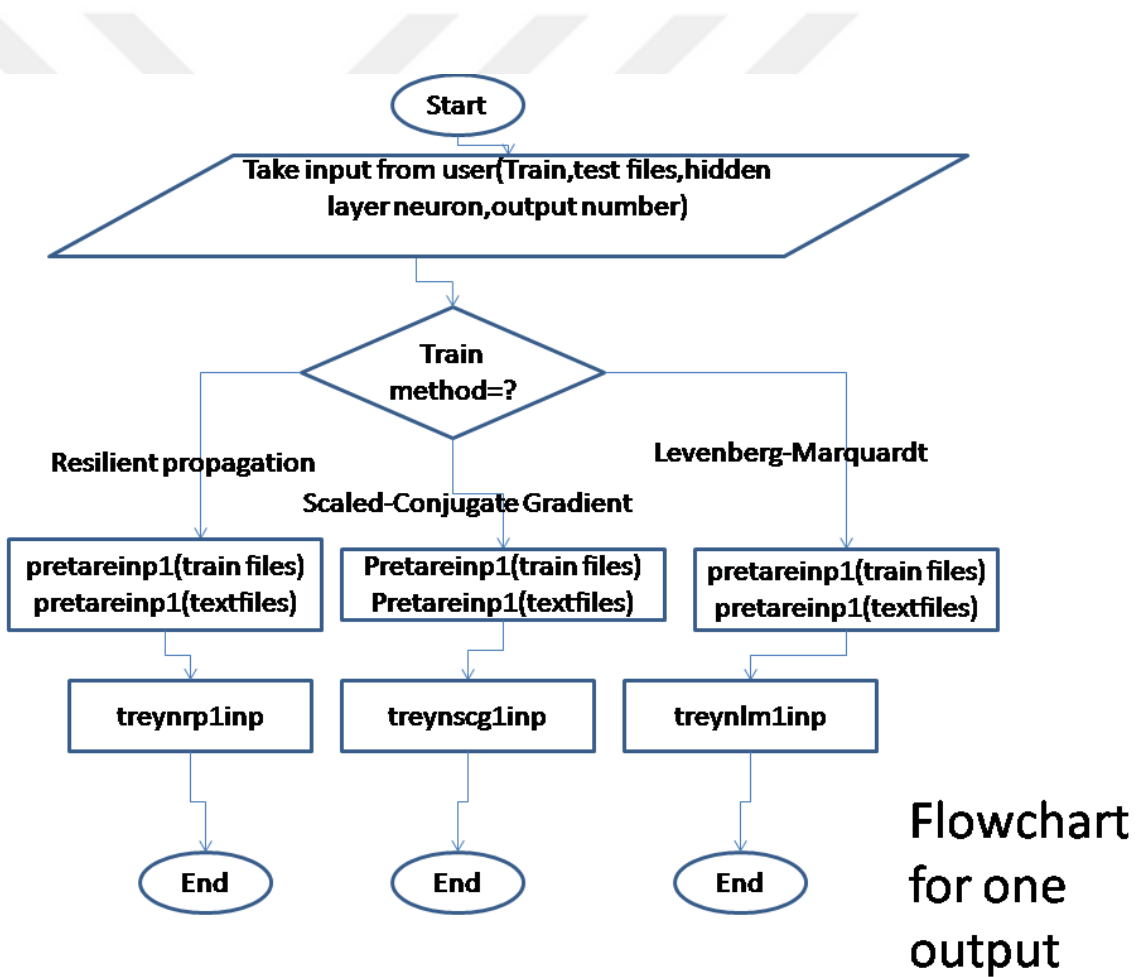


Figure 5.3: Flowchart for one output

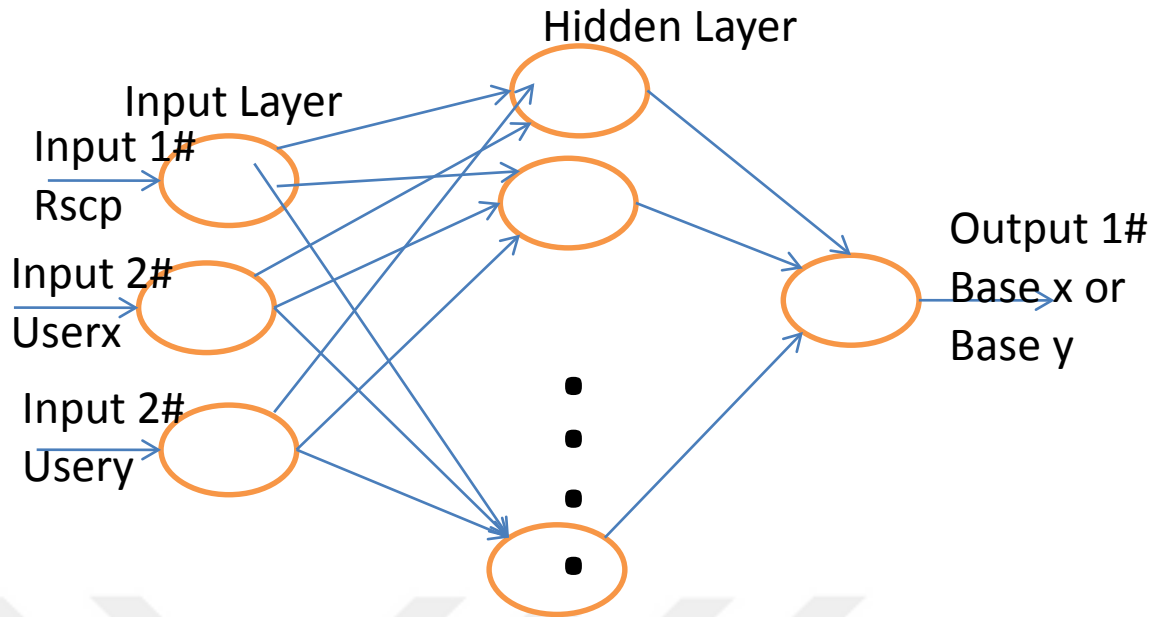


Figure 5.4: Neural network topology for one output

Flowcharts and neural network topology of the project is shown in the figures above. These figures explain the project flowchart simply and visually.

Figure 5.1 explains the flowchart for the 3 input 2 output network topology. It explains which methods will run if the user chooses 3 input 2 output network topology in its order. Neural network topology is represented in figure 5.2 visually. It briefly shows what inputs, hidden layer and outputs are taken.

Figure 5.1 explains the flowchart for the 3 input 2 output network topology. It explains which methods will run if the user chooses 3 input 2 output network topology in its order. Neural network topology is represented in figure 5.2 visually. It briefly shows what inputs, hidden layer and outputs are taken.

Below methods of the project are explained. These methods are in the flowcharts. These methods benefit from open source library "Encog Library". Encog is a jar file that contains machine learning methods. Encog library is included in the java project to use its methods in source code. For example, if we want to create a network, we can call it by typing `network1 = new BasicNetwork();` This creates a ready network from encog library. Furthermore, if we want to use a training method

it is enough to type `LevenbergMarquardtTraining train = new LevenbergMarquardtTraining(network1, trainingSet);`



Dosyeokuma Project

Readdertrain method

This method's function is to take raw data and transform it into processed data. This raw data contains cellid,userx,usery,rscp,srcm,basex and basey. Basex and basey are prepared by user in excel or other programs. This raw data is prepared by user from cellphone data in excel other programs. After being prepared as comma seperated file. Program takes values from the file using comma as delimiter.

```
value=line.split(",");  
  
for(int i=0; i<value.length; i++){  
  
cellId=value[0];  
  
outputx=value[1];
```



```

    outputy=valu[2];

    rscp=valu[3];

    srcm=valu[4];

    basex=valu[5];

    basey=valu[6];

}

```

This processed data is put into a multidimensional array. Raw data contains longitude and latitude values in degree values. But we need meter values in order to train it. Therefore program makes a calculation and gives the point's distance to (0,0) point at ecuador. Program parses the data and turns into numeric values in order to use the data.

```

longi= ((Double.parseDouble(outputx)) );

late=((Double.parseDouble(outputy)) );

baseym=(Double.parseDouble(basey));

baseym2=baseym*111;

basexm=(Double.parseDouble(basex));

latee=(late*111);

longii=longi*111.195;

basexm2=basexm*111.195;

```

Program puts data into multidimensional array

```

xl1[ind][0]= Double.valueOf(cellId);

xl1[ind][1] = longii;

xl1[ind][2]= latee;

xl1[ind][3]= Double.valueOf(rscp);

```

```

xl1[ind][4]=Double.valueOf(outputx);

xl1[ind][5]=Double.valueOf(outputy);

xl1[ind][6]=Double.valueOf(srcm);

xl1[ind][7]=basexm2;

xl1[ind][8]=baseym2;

```

Arfillgertrain method

This method's function is to create an ordered multidimensional array. It continues the work that readdgertrain method started. It takes the array that readdgertrain method created and as output it makes it ordered according to cellids. And when doing this job it benefits from a basic algorithm. If it takes a cellid value, it scans the multidimensional array from beginning to end and signs them as false so that it doesn't put it into unique cellid array. At the same time multidimensional array becomes ordered.

```

for (int i = 0; i < xl1.length; i++) {

```

```

    flag= xl1[i][0];

```

```

    if(barray[i]==true){

```

```

        z1[unind]=xl1[i][0];

```

```

        unind++;

```

```

    }

```

```

//A cell-id value is read from the array and it is compared with other values

//in the array. If it matches,this values are put into another array

// In addition boarray with this index is marked as false

//Two for loops are used to go through arrays

//An array is required for unique cell ids

    for (int j = 0; j < x11.length; j++) {

        if(flag==x11[j][0]&&barray[j]!=false){

            yl1[indx][0]=x11[j][0];

            yl1[indx][1]=x11[j][1];

            yl1[indx][2]=x11[j][2];

            yl1[indx][3]=x11[j][3];

            yl1[indx][4]=x11[j][4];

            yl1[indx][5]=x11[j][5];

            yl1[indx][6]=x11[j][6];

            yl1[indx][7]=x11[j][7];

            yl1[indx][8]=x11[j][8];

            barray[j]=false;

            indx++;

        }

    }

}

```

Objefillgertrain method

This method's function is to create cell objects. These cell objects each have a unique cell id and in addition it has its own cell's user longitude, latitude and rscp values. In short it has everything necessary and important for the cell. It makes this according to a basic algorithm. It takes the unique cellid array and ordered multidimensional array. It traverses multidimensional array by cellids and takes all data related to that cell. Then it puts this data to a cell object. All necessary and important data is stored in that cell object. After an object is finished, it goes to next cell object that is in the object array.

```
for (int i = 0; i < xl1.length; i++) {  
  
    myindex=0;  
  
    while(xl1[i]==yl1[index][0]&&xl1[i]!=0){  
  
        obje1[i].cellId=yl1[index][0];  
  
        obje1[i].longitude[myindex]=yl1[index][1];
```

```

    obje1[i].latitude[myindex]=yl1[index][2];

    obje1[i].rscp[myindex]=yl1[index][3];

    obje1[i].longer[myindex]=yl1[index][4];

    obje1[i].latger[myindex]=yl1[index][5];

    obje1[i].Scrm[myindex]=yl1[index][6];

    obje1[i].basex[myindex]=yl1[index][7];

    obje1[i].basey[myindex]=yl1[index][8];

    myindex++;

    index++;

    if(xl1[i]==0)

        break;

    if(myindex>obje1[i].longitude.length)

        break;

    if(index>yl1.length)

        break;

    } }

```

Pretare method

```

public static void pretare (String path,double[][]input,double[][]target,double[]cellldar,
double[]lngavg, double[]latavg)

```

Pretare method takes 6 parameters as input. “String path” variable takes the file path of the file which is read.”double[][]input” is entered by program as a blank array. After the function processes this array, it is filled with x,y and rscp values.” double[][]target “variable is entered

by program as a blank array. After the function processes this array, it is filled with base x and base y values. "double[]cellIdar" variable is entered by program as a blank array. After the function processes this array, it is filled with cell-id list." double[]lngavg" is a blank array entered by program. After the function processes this array, it is filled average longitude value of a specific cell." double[]latavg" is a blank array entered by program. After the function processes this array, it is filled average latitude value of a specific cell. Algorithm logic is similar in pretareInp method.

GUI class or main class that called myproject class' pretare method uses static arrays for use in method. These filled in static arrays are then used for other methods. Pretare method takes three arrays and one string as arguments. String path specifies the location of the file that will be read. A line is read, then it is divided into pieces by a delimiter. Zero index piece becomes the cell id of the data, first will be x coordinate of user, second will be y coordinate of user, third will be rscp value of the user gsm. Fourth will be the x coordinate of the base station and fifth will be the y coordinate of the base station.

```
        values=line.split(";");  
  
        for(int i=0; i<values.length; i++){  
  
            cell=values[0];  
  
            outputx=values[1];  
  
            outputy=values[2];  
  
            rscp=values[3];  
  
            desiredx=values[4];  
  
            desiredy=values[5];  
  
        }
```

Zero index data piece will be put into cellid array. This cellid array will not be used for training but will be used for report later. First, second and third values will be put into multidimensional input array. Fourth and fifth data pieces will be used for multidimensional target array. Input and target arrays will be used for training and testing. They are also needed in reports as well. Pieces are strings. By DoubleValueOf method they're converted to double values. There is an algorithm that distinguishes distinct cellid values.

If the cellid value is same with the previous value, it doesn't fill in the array with the same cellid values.

```
if(ind>0&&cellIdar[ind]!=cellIdar[ind-1])  
  
    cellIdar[ind]=Double.valueOf(cell); //Check if the previous cellid is  
    same
```

Pretare1inp method

```
public static void pretare1inp(String path, double[][]x1, double[][]y1)
```

Pretare1inp method takes 3 parameters as input. "String path" variable takes the file path of the file which is read. "double[][]x1" is entered by program as a blank array. After the function processes this array, it is filled with x, y and rscp values. "double[][]y1" variable is entered by program as a blank array. After the function processes this array, it is filled with base x and base y values.

Train and Test program methods

Treynger method

```
public static void treynger(double[][]traininput,  
double[][]traintarget, double[][]testinput, double[][]testtarget, int  
hlayerneur, double[]cellIdar, double[]lngavg, double[]latavg, double[]celltrainIdar)
```

Treynger method takes 9 parameters as input. "double[][]traininput" variable is an array that pretare method prepared for treynger method. This array contains values for the training input. "double[][]traintarget" variable is an array that pretare method prepared for treynger method. This array contains values for the training target. "double[][]testinput" is an array that pretare method prepared for treynger method. This array contains values for test input. "double[][]testtarget" is an array that pretare method prepared for treynger method. This array contains values for test target. In fact test target can be entered as "0" by user. But when measuring performance of the program it is needed. "int hlayerneur" variable is entered by user in the GUI interface. It represents the hidden layer neuron number. "double[]cellIdar" variable is an array that pretare method prepared for treynger method. This array contains cell-id list of test arrays. "double[]celltrainIdar" variable is an array that pretare method prepared for treynger method. This array contains cell-id list of train arrays. "double[]lngavg" variable is an array that pretare method prepared for treynger method. This array contains average longitude of arrays. This value is used to convert normalized x values to denormalized real longitude values. "double[]latavg" variable is an array that pretare method prepared for treynger method. This array contains average latitude of arrays. This value is used to convert normalized y values to

denormalized real longitude values. Algorithm logic is similar in `treyngerpp`, `treyngersc` methods.

Treynger method's objective is to take arrays as input and write a log file as output. It firstly creates name of the report files. The date of current time is the name of the file. One report shows the whole predictions and the other report show the statistics of predictions. To create a network, encog machine learning library is implemented. Input and output layers are specified. In addition, hidden layer neuron numbers and activation function is determined.

```
network1 = new BasicNetwork();  
  
network1.addLayer(new BasicLayer(3));  
  
network1.addLayer(new BasicLayer(new ActivationTANH(),true,hlayerneur));  
  
network1.addLayer(new BasicLayer(2));  
  
network1.getStructure().finalizeStructure();  
  
network1.reset();
```

Training sets are taken from the arguments of the function. Two arrays are provided as input and target of trainset function.

```
MLDataSet trainingSet = new BasicMLDataSet(traininput, traintarget);
```

Then, training algorithm and training sets train the network that we want to be trained

```
LevenbergMarquardtTraining train = new  
LevenbergMarquardtTraining(network1, trainingSet);
```

Iteration method specifies how many iterations will network take to approximate the ideal results. At certain number ,the results become constant so it is important to obtain this number.

```
train.iteration(epoch);
```


At the end of training, trained network can be saved in a location that the user wishes. So that when user wishes to use the network it doesn't need to train it again. User just retrieves and loads the trained network.

```
EncogDirectoryPersistence.saveObject(new File(FILENAME), network1);
```

For test sets the same method that was used in training sets are used.

A dataset is created. As arguments, this dataset takes test arrays instead of train arrays this time.

```
MLDataSet testSet = new BasicMLDataSet(testinput,  
testtarget); //testinput, testtarget are test arrays
```

This method's other function is to write reports. To realize this task, test set is iterated and in a loop each input data's output is predicted by encog library's BasisNetwork.compute method

```
for(MLDataPair pair: trainingSet2 ) {  
  
    count++;  
  
    sayac++;  
  
    final MLData output = network1.compute(pair.getInput());
```

Output is of type MLData. Each prediction is predicted from output's get data method. These predictions are base station location's x and y coordinates. They are written on a string called line. Later, they're written on BufferedWriter's class' writer object.

```
myLine3=format2.format(output.getData(0))+";"+format2.format(output.getData  
(1))+";"+
```

```
format2.format(pair.getIdeal().getData(0))+";"+format2.format(pair.getIdeal().g  
etData(1));
```

object. This method also creates a log file that calculates each cell's statistics. To realize this task, a carefully designed algorithm is needed. To distinguish at which cell iterator is iterating, flags are used. If these flag values are the same with previous values, nothing happens. However if they aren't same with previous values a chunk of code is executed. This code prepares the content of second report. Each cell's average prediction values are calculated. This average value can determine the correctness of the program. After that the total variables and counter variable sayac are changed as 0. Because for new cell everything must be new.

Average predicted x and average predicted y values are put into corresponding arrays. Also the ideal targets are put into corresponding arrays. Later these values are written into a string to be put in a write file.

```
if(flag!=pair.getIdeal().getData(0)&&flag2!=pair.getIdeal().getData(1))/flag2
koy
{
    indek++;

    xide[indek]=pair.getIdeal().getData(0);

    yide[indek]=pair.getIdeal().getData(1);

    myx[indek]=totalx/sayac;

    myy[indek]=totaly/sayac;

    System.out.println("My Average"+indek+"=" +myx[indek]);

    totalx=0;

    totaly=0;

    sayac=0;

}
```

These values are put into some calculations to calculate the distance between predicted and real values.

```
myLine2="Cell"+(i+1)+";"+"CellId"+cellIdar[i]+";"+format.format(xide[i])+
"+format.format(yide[i])+";"+format.format(myx[i+1])+";"+format.format(myy
[i+1])+";" //Comparison of predictions and desired targets
```

```
Math.sqrt((Math.pow((Math.abs(xide[i]-
myx[i+1])*1000),2))+(Math.pow((Math.abs(yide[i]-
myy[i+1])*1000),2))))//Distance calculation
```

values. Function ends by closing the bufferedwriter objects. This is essential to release resources for other code pieces.

```
writer.close();
writer2.close();
```

Treynlm1inpmethod

```
. public static void treynlm1inp(double[][]traininput,
double[][]traintarget,double[][]x12,double[][]y12,int hlayerneur,double[]cellIdar)
```

Treynlm1inp method takes 6 parameters as input.” double[][]traininput” variable is an array that pretareinp1 method prepared for treynlm1inp method. This array contains values for the training input. “double[][]traintarget” variable is an array that pretareinp1 method prepared for treynlm1inp method. This array contains values for the training target.” double[][]x12” is an array that pretareinp1 method prepared for treynlm1inp method. This array contains values for test input. “double[][]y12” is an array that pretareinp1 method prepared for treynlm1inp method. This array contains values for test target. In fact test target can be entered as “0” by user. But when measuring performance of the program it is needed. “int hlayerneur” variable is entered by user in the GUI interface. It represents the hidden layer neuron number.” double[]cellIdar “ variable is an array that pretare method prepared for treynlm1inp method. This array contains cell-id list of test arrays. Algorithm logic is similar in treynlm1inp,treynrp1inp,treynscg1inp methods.

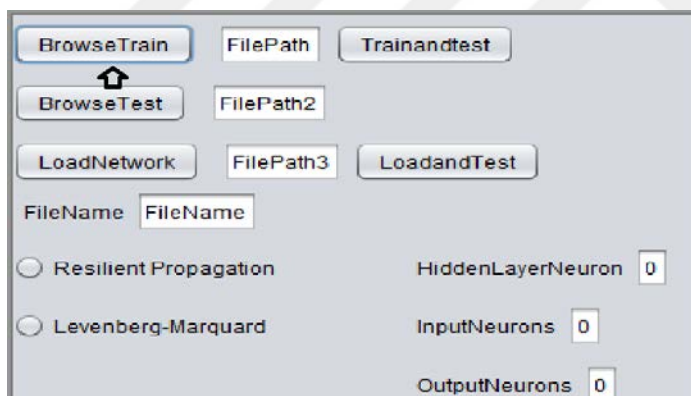


Figure 5.5: BrowseTrain Button

BrowseTrain Button: When this button is clicked as shown in figure 5.5, user is prompted to select a file from the file browser window. This file that user selected , as the button's name indicates, is the training file. This training file should be a text file. This text file should contain data that is separated by comma or semicolon. One row should contain five columns each separated by semicolon or comma. These columns form the input and target of the training program. The first three columns: user x,user y and rscp form the input of the training. The last two columns form the target of the training. Once the network is trained, its network file is saved.

In addition the trained network is ready to be tested with the test file that comes from browsetest label and button.

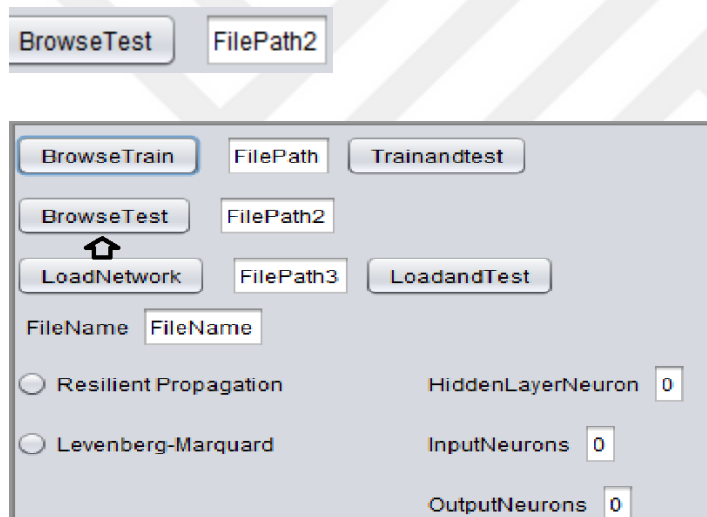


Figure 5.6: BrowseTest Button

BrowseTest Button: When this button (as shown in figure 5.6) is clicked, user is prompted to select a file from the file browser window. This file that user selected , as the button's name indicates, is the test file. This test file should be a text file. This text file should contain data that is seperated by comma or semicolon. One row should contain five columns each seperated by semicolon or comma. These columns form the input and target of the training program. The first three columns: user x,user y and rscp form the input of the text. The last two columns aren't needed for test. But for the sake of easiness, training files can also be used as test files, five column test file can be used. In practice the last two columns are never used in the

program. But every training file that is prepared can also be used as test file as well. When LoadNetwork and Test button or Train and Test button is clicked this selected file will be used as test file. As output, program will give us the computer's predictions. Once these buttons are

clicked a prediction report will be created. On this report there will be predicted results and ideal targets. Therefore we will be able to measure the accuracy of predictions.

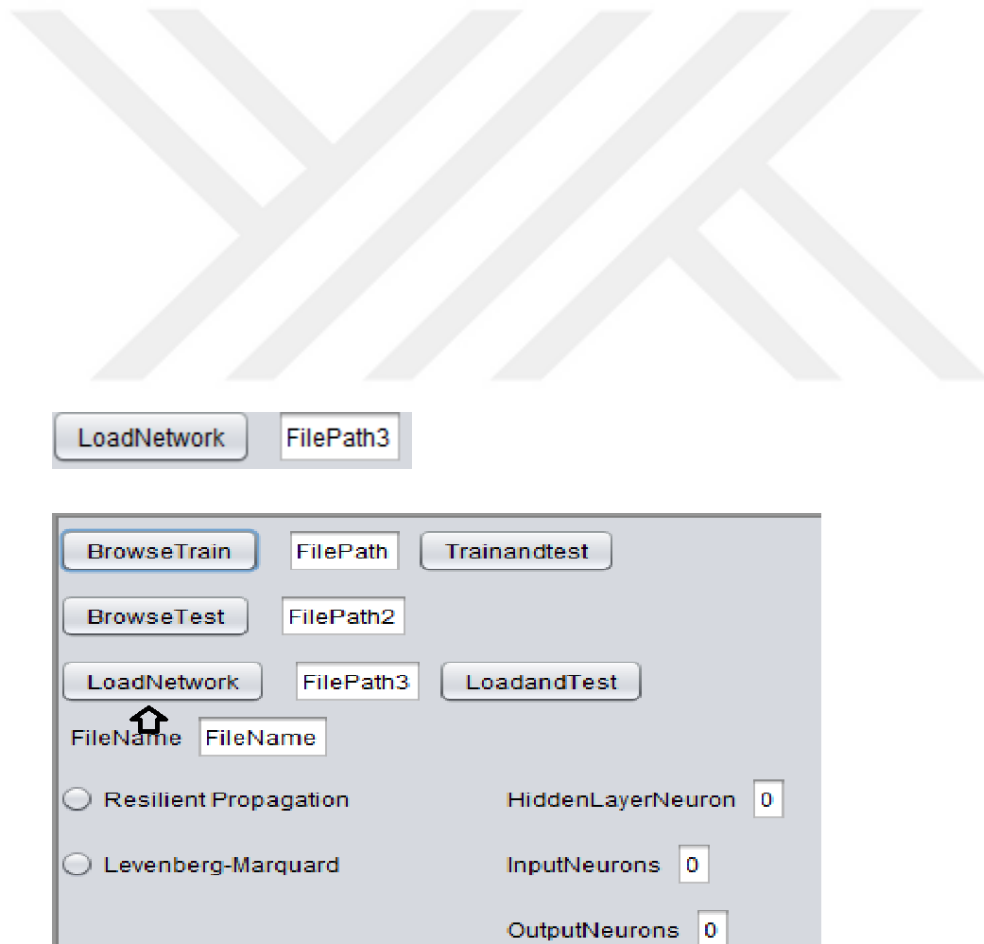


Figure 5.7: LoadNetwork Button

LoadNetwork Button: When this button(as shown in figure 5.7) is clicked user is prompted to select a file from the file browser window. This file that user selected , as the button's name indicates, is the saved network file. This file should be a file

that has the records of the saved network. When file is selected, its name is put into text input file. And when LoadNetwork and Test button is clicked, filename that is in the text input file that LoadNetwork Button selected is used for the needed network.

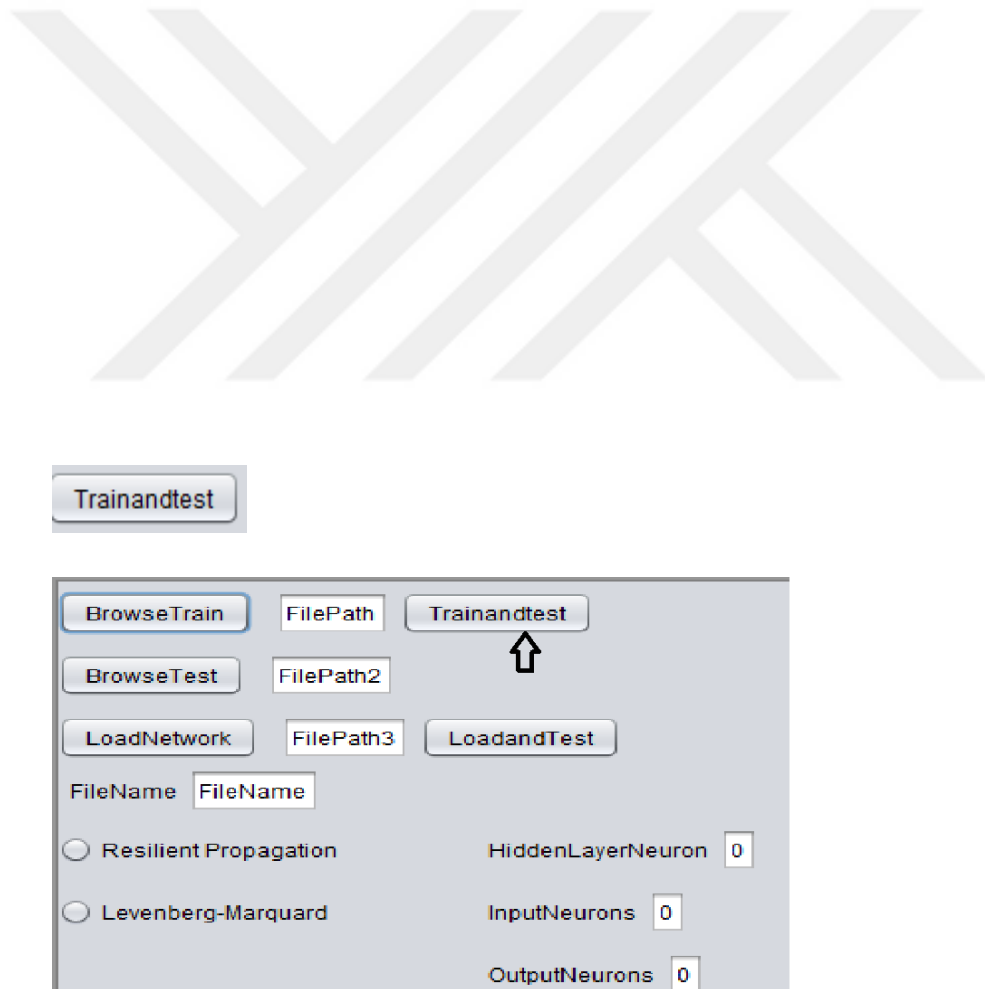


Figure 5.8: TrainandTest Button

TrainandTest Button: When this button(as shown in figure 5.8) is clicked, the computer takes the train filename that is in the train text input file and the test

filename that is in the test text input file. Program firstly trains the network with the provided train file. Later, it tests this trained network with the provided test file. Finally, it creates a prediction report file. On this report there are outputs of given test file for every inputs. For every user x (user location x), user y (user location y) and rscp inputs computer estimates a base location x and base location y . Program realizes this task according to its biases and weights. These are calculated in the training phase. Network trains itself with user location inputs and ideal base location targets.

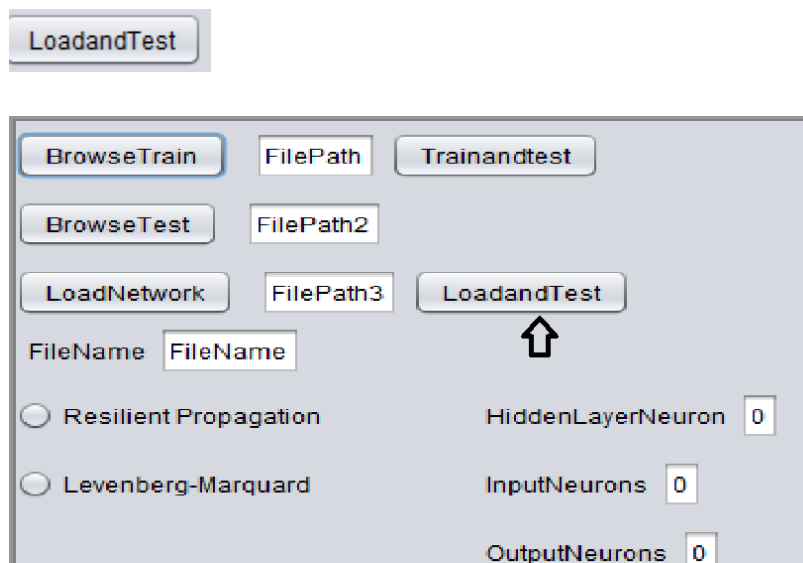


Figure 5.9: LoadandTest Button

LoadandTest Button: When this button(as shown in figure 5.9) is clicked, the computer takes the network filename that is in the loadnetwork text input file and the test filename that is in the test text input file. Program takes the network with the provided network file. This network is trained with given inputs earlier and saved somewhere in the computer. Program takes the test file that is in the testfile filepath. This way, it gives the test input to saved network and as output takes a prediction report. On this report there are outputs of given test file for every inputs.

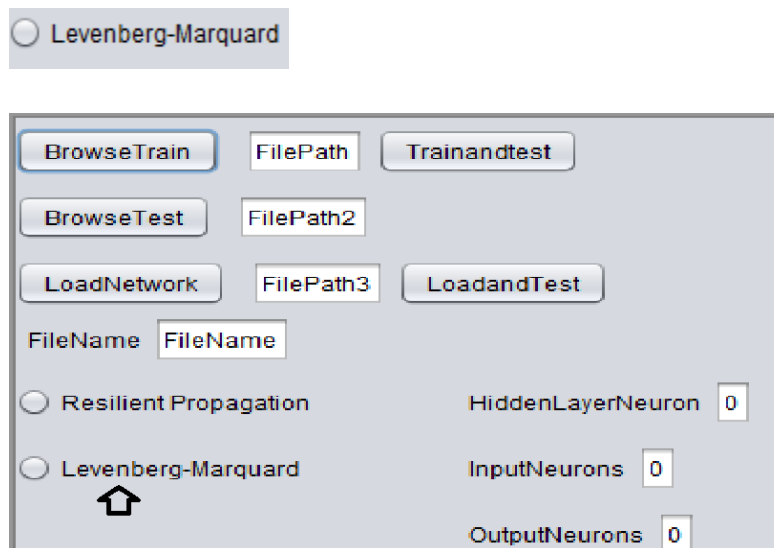


Figure 5.10: Levenberg-Marquard radio button

Levenberg-Marquard radio button: This radio(as shown in figure 5.10) button determines the training method of the train file as Levenberg-Marquard. This training method gives one of the best results for the base location estimation problem. Network is trained by Levenberg-Marquard method. Training method is one of the parameters of the training. In the program user can either choose Levenberg-Marquard training method or Resilient Propagation training method. When TrainandTest or LoadandTest button is clicked,it is used as the training method of the training

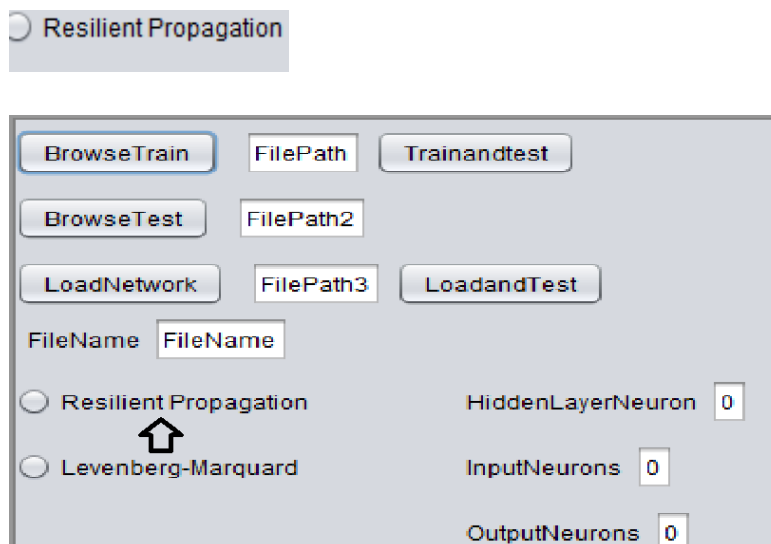


Figure 5.11: Resilient Propagation radio button

Resilient Propagation radio button: This radio button(as shown in figure 5.11) determines the training method of the train file as Resilient Propagation. Network is trained by Resilient Propagation method. Training method is one of the parameters of the training. This training method gives one of the best results for the base location estimation problem. In the program user can either choose Levenberg-Marquard training method or Resilient Propagation training method. When TrainandTest or LoadandTest button is clicked,it is used as the training method of the training

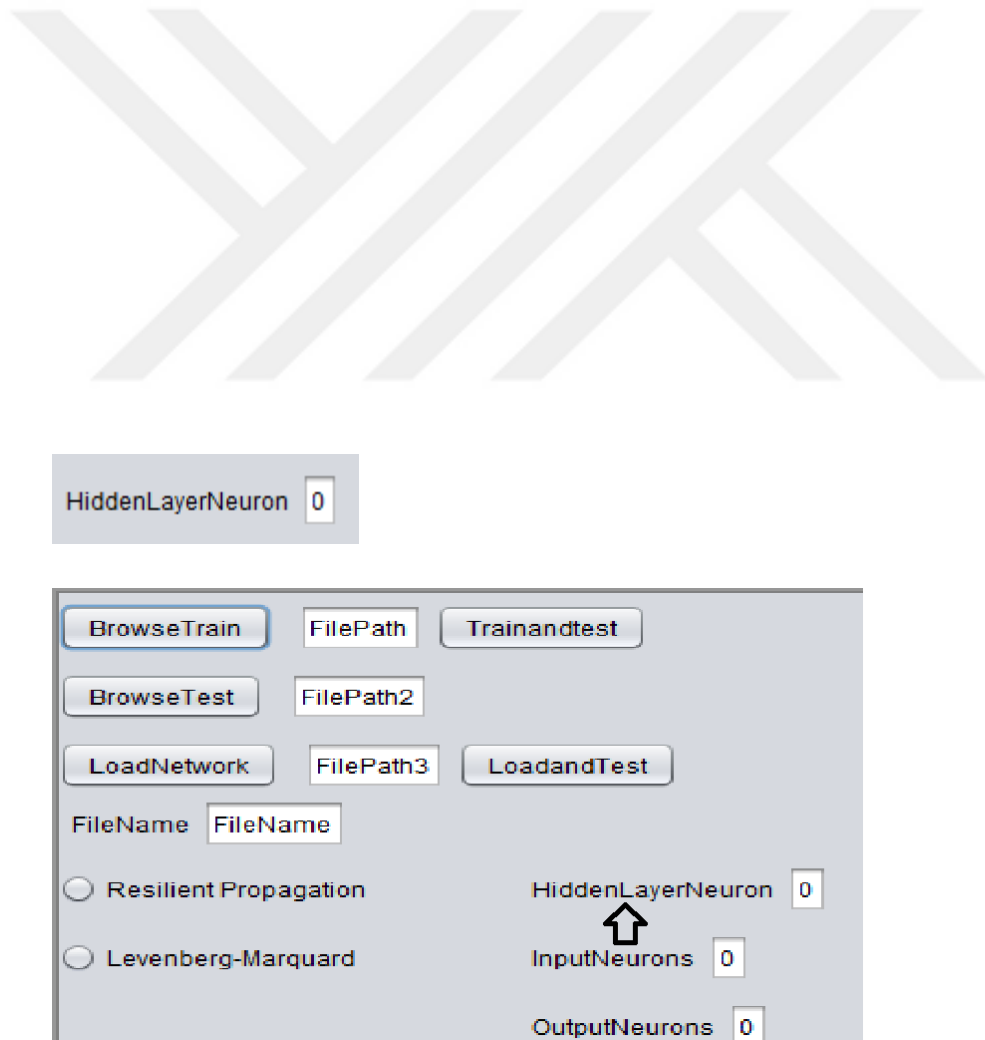


Figure 5.12: Hidden Layer Neuron InputText

Hidden Layer Neuron InputText: This inputtext(as shown in figure 5.12) determines the hidden layer neuron number of the training. User enters the hidden layer number and when TrainTest or LoadandTest button is clicked,it is used as the hidden layer neuron number of the training.

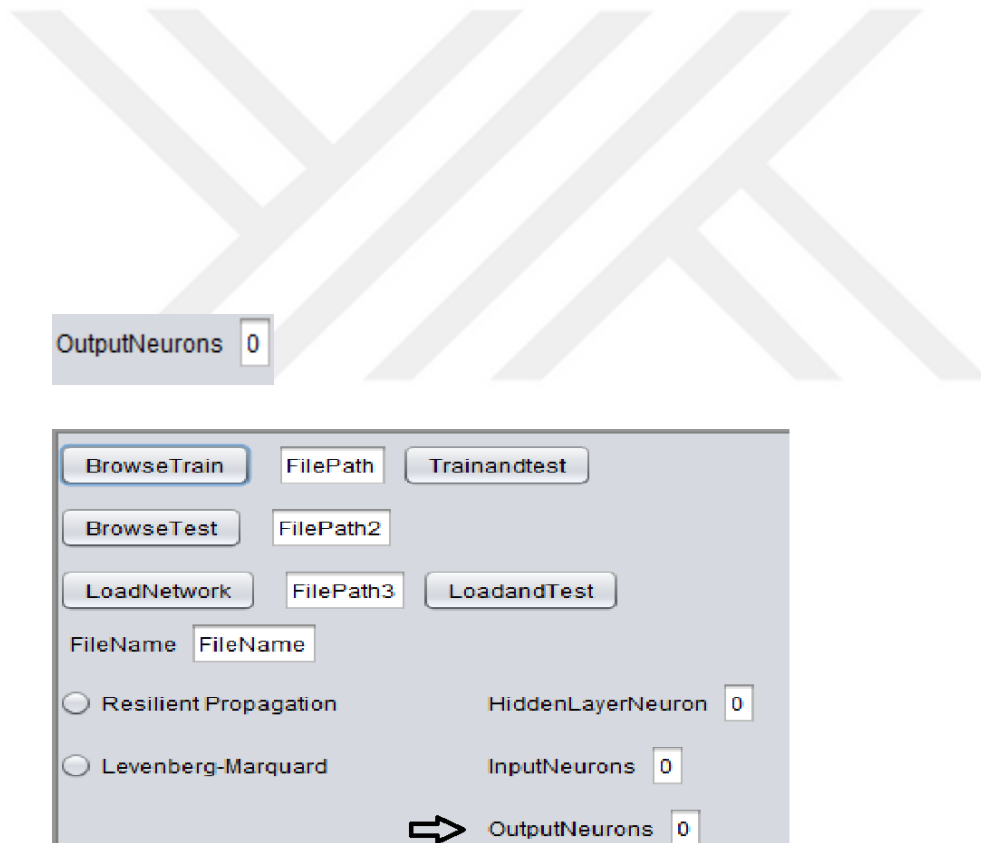


Figure 5.13: Output Neurons InputText

Output Neurons InputText: This inputtext(as shown in figure 5.13) determines the output layer neuron number of the training. User enters the output layer neuron number and when TrainTest or LoadandTest button is clicked,it is used as the output layer neuron number of the training. For the base location problem output must be

2 or 1. If you enter output as 1 x or y of the base location is trained and predicted or if you enter output as 2 x and y of the base location together is trained and predicted



CHAPTER 6

TESTS AND RESULTS

In this section, train and test of the networks with related training methods are discussed. Firstly, Matlab program is used for training and results are obtained and reported. Training sets and tests are created for this process. Built-in functions of Matlab are used for Matlab source code of project.

Secondly, Encog (open source machine learning library) supported Java program is used for training and testing. 21 training sets and 5 test sets are created for various training and testing. The aim is to get good results with a good training and test sets.

In the Matlab reports, training set cells and test set cells can be seen. In each row, the results of each cells can be seen. Furthermore, ideal base location x and ideal base location y can be seen. These values are in meters. These values are compared with the values that software evaluated and distance x and distance y values are calculated. From these distance x and distance y values, distance between real base location and predicted base location is calculated.

After Matlab trainings and testings, trainings and testings are done in Java program. Same calculations in the Matlab are done. Ideal base location and predicted base location are computed and distance between them are calculated. In each row, the results of each cells can be seen.



MATLAB RESULTS

Sample1

	myx	basex	myy	basey	distancex	distancey	distance	cell-Id
Cell1	376,4003	549,4513	109,5495	122,8933	173,0509	13,34384	173,5646	17611
Cell2	423,5533	553,5277	94,31267	143,6853	129,9743	49,37263	139,0359	11749
Cell3	-268,325	-304,629	-131,191	-124,302	36,30359	6,888846	36,95141	25617
Cell4	-187,343	-108,446	-78,4281	20,60549	78,89679	99,03354	126,6189	26297
Cell5	122,1103	309,6725	-611,819	-514,29	187,5622	97,52901	211,4036	35050
Cell6	-88,1455	-48,9495	-63,8627	-97,9007	39,19606	34,03796	51,91256	35610
Cell7	-177,956	-362,987	-55,7186	63,99094	185,0316	119,7095	220,3794	37442
Cell8	-357,42	-464,227	-111,674	-137,575	106,8074	25,90108	109,903	32430
Cell9	-358,485	-387,169	-96,1191	-91,9491	28,68406	4,170023	28,98559	32431
Cell10	-12,3413	-32,4406	-927,114	-962,383	20,09931	35,26964	40,5947	35223
Cell11	140,1116	304,9106	-614,284	-563,841	164,799	50,44288	172,3461	35053
Cell12	-159,523	-153,48	-78,9241	-113,822	6,043442	34,89755	35,41698	25625
Cell13	-8,43157	107,5322	-72,0504	-60,7636	115,9638	11,28675	116,5118	25627
Cell14	-345,153	-471,929	-1084,57	-1346,23	126,7759	261,6568	290,7514	52461
Cell15	-134,056	42,24743	-82,4176	-52,7713	176,303	29,64638	178,7783	40143
Cell16	-38,6841	91,45344	-87,1988	-162,955	130,1375	75,75606	150,5814	40149
Cell17	-164,384	-375,655	-313,023	-210,934	211,2706	102,0889	234,6432	6700
Cell18	39,76581	-107,614	-65,7477	122,0404	147,38	187,7881	238,7158	29515
Cell19	-136,885	-138,987	-80,7944	70,92234	2,10191	151,7168	151,7313	32571
Cell20	-20,7369	-10,9377	-859,494	-956,273	9,799174	96,77926	97,2741	35222
Cell21	-133,271	-476,085	-115,646	-236,75	342,8144	121,104	363,5765	6701
Cell22	-1,33027	-83,5411	-86,0604	117,1974	82,21078	203,2578	219,2541	29554
Cell23	191,8905	225,1869	-446,524	-230,808	33,29641	215,716	218,2706	29555
Cell24	-39,0825	-159,122	-331,818	-138,682	120,0392	193,1359	227,4003	32570
Cell25	-174,511	55,73872	-62,8981	98,88657	230,2501	161,7847	281,4061	32572
Cell26	49,60828	87,21747	-68,5608	-47,8443	37,60918	20,71644	42,93741	25619
Cell27	-231,233	-1043,4	-35,496	-10,6871	812,1632	24,8089	812,542	37440
Cell28	-233,374	-1021,4	-35,9086	-4,19469	788,0275	31,71392	788,6654	37441
Cell29	-113,431	232,8673	-225,027	-73,3233	346,2988	151,7039	378,07	37443
Cell30	-70,0526	-292,045	-224,443	-241,895	221,9921	17,45158	222,677	43078
Cell31	-6,07185	-292,435	-364,597	-109,584	286,3632	255,013	383,4521	43079
Cell32	-123,357	-143,156	-87,9157	-121,937	19,79881	34,0217	39,3633	53665
Cell33	-246,766	-174,093	-101,466	124,332	72,67263	225,7984	237,2051	40141

	X error	Y error	DISTANCE
AVERAGE	165,749	95,25884	212,7552

Table 6.1: Matlab test sample 1

Sample2

	myx	myy	basex	basey	distancex	distancey	distance	Celllist
Cell1	402,6432	118,6238	549,4513	122,8933	146,8081	4,269506	146,8701	17611
Cell2	423,7697	100,5036	553,5277	143,6853	129,758	43,18174	136,7545	11749
Cell3	-266,284	-42,2915	-304,629	-124,302	38,34453	82,01015	90,53158	25617
Cell4	-229,775	-25,7939	-108,446	20,60549	121,3287	46,39942	129,8982	26297
Cell5	268,1377	-450,14	309,6725	-514,29	41,53479	64,14939	76,42174	35050
Cell6	-251,929	-20,8428	-48,9495	-97,9007	202,9795	77,05787	217,1142	35610
Cell7	-206,747	-8,96602	-362,987	63,99094	156,2404	72,95697	172,4349	37442
Cell8	-409,704	-104,06	-464,227	-137,575	54,52295	33,5153	64,00022	32430
Cell9	-131,078	-26,9142	-375,655	-210,934	244,5771	184,0196	306,0738	6700
Cell10	-46,7106	19,98059	-107,614	122,0404	60,90362	102,0598	118,8505	29515
Cell11	-250,547	-30,5028	-138,987	70,92234	111,5601	101,4252	150,7738	32571
Cell12	234,1402	-246,782	304,9106	-563,841	70,77038	317,0593	324,8616	35053
Cell13	-247,05	-27,1673	-153,48	-113,822	93,57064	86,6543	127,5321	25625
Cell14	-160,974	6,943711	107,5322	-60,7636	268,5066	67,70733	276,9117	25627
Cell15	-233,238	-42,9784	42,24743	-52,7713	275,4853	9,792831	275,6593	40143
Cell16	-143,249	6,180941	91,45344	-162,955	234,7023	169,1358	289,2959	40149

	X error	Y error	Distance
Average	140,7246	91,33715	181,499

testlist	trainlist
17611	17611
11749	11749
25617	25617
26297	26297
35050	35050
35610	35610
37442	37442
32430	32430
6700	
29515	
32571	
35053	
25625	
25627	
40143	
40149	

Table 6.2: Matlab test sample 2

ENCOG RESULTS

Sample result1

Cells		Basex	Basey	Myx	Myy	Distance	Mylon	Mylat
Cell1	6700	-0,46	-0,14	-0,338	-0,047	153.485	29.046	41.118
Cell2	11749	0,55	0,14	-0,097	-0,129	700.266	28.963	41.024
Cell3	17611	0,55	0,12	-0,14	-0,032	706.715	28.963	41.025
Cell4	25617	-0,3	-0,12	-0,211	-0,013	139.411	28.991	41.076
Cell5	26297	-0,11	0,02	-0,08	-0,111	134.434	28.998	41.077
Cell6	29515	-0,11	0,12	-0,167	-0,001	133.467	28.968	41.017
Cell7	32430	-0,46	-0,14	-0,34	-0,047	152.069	29.051	41.126
Cell8	32571	-0,14	0,07	-0,163	0,001	72.965	29.052	41.128
Cell9	35050	0,31	-0,51	0,249	-0,437	95.042	29.053	41.118
Cell10	35610	-0,05	-0,1	-0,157	0,002	147.851	28.959	41.024
Cell11	37442	-0,36	0,06	-0,232	-0,018	150.015	29.032	41.128
Cell12	6701	-0,48	-0,24	-0,143	0	413.905	29.048	41.119
Cell13	25625	-0,15	-0,11	-0,212	-0,012	115.572	28.961	41.023
Cell14	25627	0,11	-0,06	-0,176	-0,003	292.019	28.959	41.022
Cell15	29554	-0,08	0,12	-0,19	-0,007	168.016	28.968	41.017
Cell16	29555	0,23	-0,23	0,107	-0,341	165.665	28.968	41.018
Cell17	32431	-0,36	0,1	-0,187	-0,019	209.664	29.052	41.125
Cell18	35053	0,3	-0,56	0,243	-0,45	124.086	29.053	41.119
Cell19	37443	0,23	-0,07	-0,06	-0,118	294.396	29.028	41.128
Cell20	37444	-0,36	0,1	-0,185	-0,012	207.420	29.032	41.128
Cell21	40143	0,04	-0,05	-0,143	0,006	191.574	28.957	41.025
Cell22	40144	-0,14	0,07	-0,12	-0,02	92.025	28.959	41.024
Cell23	40149	0,09	-0,16	-0,099	-0,027	231.266	28.957	41.026
Cell24	43078	-0,29	-0,24	-0,131	0,009	295.863	28.965	41.022
Cell25	43079	-0,29	-0,11	0,049	-0,21	353.524	28.966	41.019
Cell26	53665	-0,14	-0,12	-0,143	0,005	125.461	29.024	41.130

Training set arrays

29515

x error y error distance

32430

average 0,159 0,113 0,195317

32571

35050

35610

37442

Table 6.3: Encog test sample 1

Sample Result2

Cells		Basex	Basey	Myx	Myy	Distance	Mylon	Mylat
Cell1	6700	-0,46	-0,14	-0,138	-0,007	347.826	29.047	41.119
Cell2	11749	0,55	0,14	0,276	0,074	281.580	28.967	41.025
Cell3	17611	0,55	0,12	0,22	0,062	334.511	28.966	41.025
Cell4	25617	-0,3	-0,12	-0,048	0,01	283.472	28.993	41.076
Cell5	26297	-0,11	0,02	-0,072	0,005	40.510	28.998	41.078
Cell6	29515	-0,11	0,12	-0,118	-0,004	123.903	28.969	41.017
Cell7	32430	-0,46	-0,14	-0,144	-0,008	342.050	29.053	41.126
Cell8	32571	-0,14	0,07	-0,087	0,002	86.056	29.052	41.128
Cell9	35050	0,31	-0,51	0,096	0,038	587.895	29.051	41.123
Cell10	35610	-0,05	-0,1	-0,123	-0,005	120.287	28.959	41.024
Cell11	37442	-0,36	0,06	-0,106	-0,001	261.352	29.033	41.128
Cell12	6701	-0,48	-0,24	-0,02	0,015	525.618	29.049	41.119
Cell13	25625	-0,15	-0,11	-0,12	-0,004	110.261	28.962	41.023
Cell14	25627	0,11	-0,06	-0,135	-0,007	250.475	28.959	41.022
Cell15	29554	-0,08	0,12	-0,135	-0,007	138.416	28.968	41.017
Cell16	29555	0,23	-0,23	-0,001	0,019	339.450	28.967	41.021
Cell17	32431	-0,36	0,1	-0,039	0,012	332.968	29.053	41.126
Cell18	35053	0,3	-0,56	0,256	0,069	630.145	29.053	41.123
Cell19	37443	0,23	-0,07	-0,079	0,004	317.776	29.028	41.129
Cell20	37444	-0,36	0,1	-0,031	0,013	340.654	29.034	41.128
Cell21	40143	0,04	-0,05	-0,138	-0,007	183.062	28.957	41.025
Cell22	40144	-0,14	0,07	-0,119	-0,004	76.679	28.959	41.024
Cell23	40149	0,09	-0,16	-0,096	0	245.945	28.957	41.026
Cell24	43078	-0,29	-0,24	-0,073	0,005	326.973	28.965	41.022
Cell25	43079	-0,29	-0,11	-0,137	-0,007	183.981	28.964	41.021
Cell26	53665	-0,14	-0,12	-0,144	-0,009	111.395	29.024	41.130

Training set arrays

6700				x error	y error	distance
11749	average	0,189	0,147	0,239811		
17611						
25617						
26297						
29515						

Table 6.4: Encog test sample 2

Base stations estimations on map

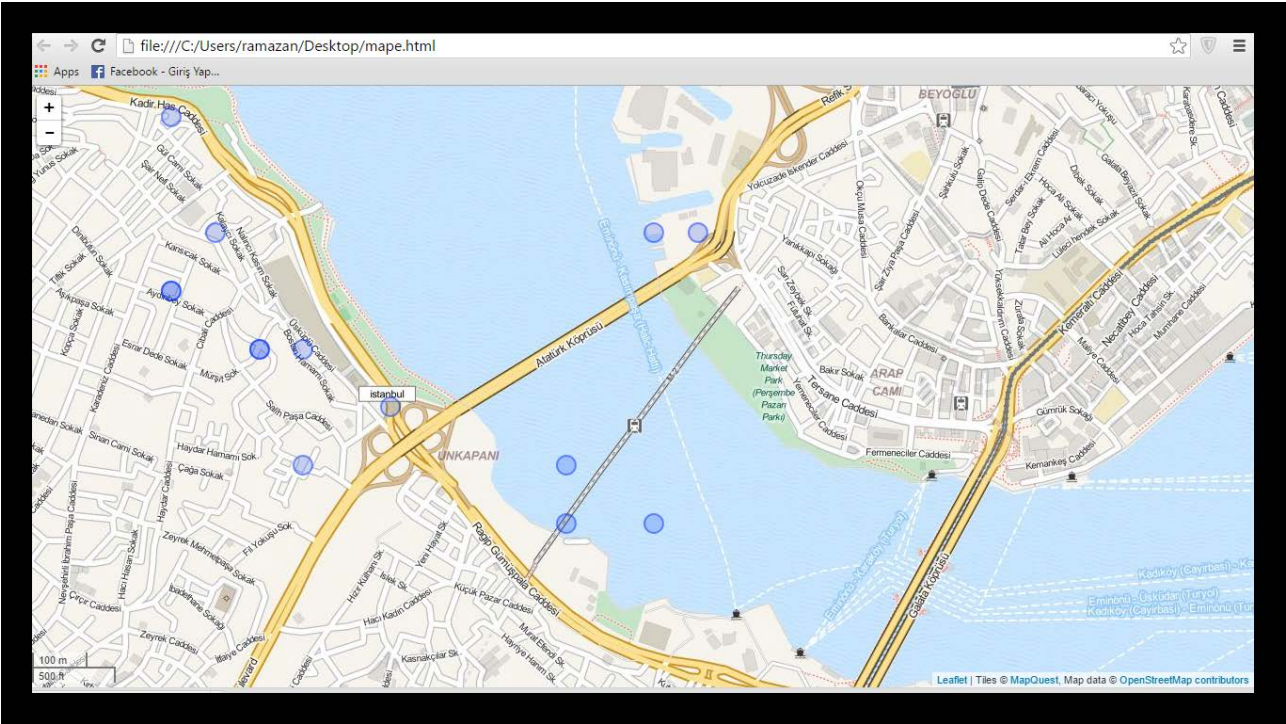


Figure 6.1: Base locations on map

Sample Result 3

Cells		Basex	Basey	Myx	Myy	Distance	Myxlon	Myxlat
Cell1	6700.0	-0.460	-0.140	-0.050	-0.004	431.554	29.048	41.119
Cell2	11749.0	0.550	0.140	-0.254	0.067	807.125	28.962	41.025
Cell3	17611.0	0.550	0.120	-0.247	0.079	798.384	28.962	41.026
Cell4	25617.0	-0.300	-0.120	-0.154	0.093	258.518	28.992	41.076
Cell5	26297.0	-0.110	0.020	0.101	-0.103	244.499	29.000	41.077
Cell6	29515.0	-0.110	0.120	-0.169	0.059	84.403	28.968	41.018
Cell7	32430.0	-0.460	-0.140	-0.047	-0.005	434.507	29.053	41.127
Cell8	32571.0	-0.140	0.070	-0.164	0.053	29.271	29.052	41.128
Cell9	35050.0	0.310	-0.510	0.178	-0.265	278.314	29.052	41.120
Cell10	35610.0	-0.050	-0.100	-0.183	0.067	213.674	28.958	41.025
Cell11	37442.0	-0.360	0.060	-0.188	0.059	171.950	29.032	41.129
Cell12	6701.0	-0.480	-0.240	-0.043	-0.005	496.191	29.049	41.119
Cell13	25625.0	-0.150	-0.110	-0.101	0.026	144.892	28.962	41.023
Cell14	25627.0	0.110	-0.060	-0.186	0.067	322.746	28.959	41.023
Cell15	29554.0	-0.080	0.120	-0.177	0.063	112.681	28.968	41.018
Cell16	29555.0	0.230	-0.230	0.165	-0.212	67.090	28.968	41.019
Cell17	32431.0	-0.360	0.100	-0.241	0.082	120.104	29.051	41.126
Cell18	35053.0	0.300	-0.560	0.264	-0.407	157.385	29.053	41.119
Cell19	37443.0	0.230	-0.070	0.193	-0.145	83.480	29.030	41.128
Cell20	37444.0	-0.360	0.100	-0.241	0.080	120.293	29.032	41.129
Cell21	40143.0	0.040	-0.050	-0.144	0.049	208.426	28.957	41.025
Cell22	40144.0	-0.140	0.070	-0.074	0.001	95.905	28.959	41.024
Cell23	40149.0	0.090	-0.160	-0.065	-0.003	220.626	28.957	41.026
Cell24	43078.0	-0.290	-0.240	-0.075	0.053	363.335	28.965	41.023
Cell25	43079.0	-0.290	-0.110	0.240	-0.159	532.234	28.968	41.020
Cell26	53665.0	-0.140	-0.120	-0.152	0.052	172.441	29.024	41.130

Training set arrays

29554

29555

32431

35053

37443

37444

average x error y error distance Table 6.5: Encog test sample 3
 0,2199 0,111154 268,078

Sample Result 4

Cells		Basex	Basey	Myx	Myy	Distance	Mylon	Mylat
Cell1	6700.0	-0.460	-0.140	-0.300	-0.054	181.761	29.046	41.118
Cell2	11749.0	0.550	0.140	0.453	0.118	99.887	28.968	41.026
Cell3	17611.0	0.550	0.120	0.373	0.103	177.607	28.967	41.026
Cell4	25617.0	-0.300	-0.120	-0.176	-0.022	158.190	28.991	41.075
Cell5	26297.0	-0.110	0.020	-0.113	0.003	17.605	28.998	41.078
Cell6	29515.0	-0.110	0.120	-0.115	-0.007	127.224	28.969	41.017
Cell7	32430.0	-0.460	-0.140	-0.306	-0.055	176.058	29.051	41.126
Cell8	32571.0	-0.140	0.070	-0.096	-0.003	85.388	29.052	41.128
Cell9	35050.0	0.310	-0.510	-0.270	0.002	773.640	29.048	41.122
Cell10	35610.0	-0.050	-0.100	-0.116	-0.006	114.916	28.959	41.024
Cell11	37442.0	-0.360	0.060	-0.132	-0.012	239.352	29.033	41.128
Cell12	6701.0	-0.480	-0.240	0.024	0.026	570.169	29.050	41.119
Cell13	25625.0	-0.150	-0.110	-0.239	-0.036	115.855	28.961	41.022
Cell14	25627.0	0.110	-0.060	-0.162	-0.020	275.410	28.959	41.022
Cell15	29554.0	-0.080	0.120	-0.208	-0.031	198.065	28.968	41.017
Cell16	29555.0	0.230	-0.230	-0.211	0.005	499.944	28.965	41.021
Cell17	32431.0	-0.360	0.100	-0.106	-0.008	275.744	29.052	41.125
Cell18	35053.0	0.300	-0.560	-0.022	0.052	690.974	29.050	41.123
Cell19	37443.0	0.230	-0.070	-0.025	0.020	270.853	29.028	41.130
Cell20	37444.0	-0.360	0.100	-0.093	-0.006	287.353	29.033	41.128
Cell21	40143.0	0.040	-0.050	-0.169	-0.020	210.811	28.957	41.025
Cell22	40144.0	-0.140	0.070	-0.118	-0.001	74.301	28.959	41.024
Cell23	40149.0	0.090	-0.160	-0.018	0.021	211.205	28.958	41.026
Cell24	43078.0	-0.290	-0.240	-0.157	-0.007	268.448	28.964	41.022
Cell25	43079.0	-0.290	-0.110	-0.265	-0.006	106.926	28.963	41.021
Cell26	53665.0	-0.140	-0.120	-0.210	-0.029	114.757	29.023	41.130

Training set arrays

6700.0
 11749.0
 17611.0
 25617.0
 26297.0
 29515.0

average x error y error distance Table 6.6: Encog test sample 4
 0,1822 0,13827 0,2287

Sample Result 5

Cells		Basex	Basey	Myx	Myy	Distance	Mylon	Mylat
Cell1	6700.0	-0.460	-0.140	-0.027	-0.068	438.444	29.048	41.118
Cell2	11749.0	0.550	0.140	-0.044	-0.073	631.199	28.964	41.024
Cell3	17611.0	0.550	0.120	-0.088	-0.089	670.914	28.963	41.024
Cell4	25617.0	-0.300	-0.120	-0.037	-0.072	267.585	28.993	41.075
Cell5	26297.0	-0.110	0.020	-0.098	-0.093	113.480	28.998	41.077
Cell6	29515.0	-0.110	0.120	-0.052	-0.077	205.125	28.969	41.017
Cell7	32430.0	-0.460	-0.140	-0.027	-0.068	439.128	29.054	41.126
Cell8	32571.0	-0.140	0.070	-0.090	-0.090	167.898	29.052	41.127
Cell9	35050.0	0.310	-0.510	-0.226	-0.139	651.825	29.048	41.121
Cell10	35610.0	-0.050	-0.100	-0.123	-0.102	73.031	28.959	41.023
Cell11	37442.0	-0.360	0.060	-0.044	-0.074	342.977	29.033	41.127
Cell12	6701.0	-0.480	-0.240	-0.089	-0.089	419.435	29.049	41.118
Cell13	25625.0	-0.150	-0.110	-0.090	-0.090	63.153	28.962	41.022
Cell14	25627.0	0.110	-0.060	-0.114	-0.099	227.590	28.960	41.021
Cell15	29554.0	-0.080	0.120	-0.092	-0.091	211.452	28.969	41.017
Cell16	29555.0	0.230	-0.230	-0.155	-0.114	402.383	28.965	41.020
Cell17	32431.0	-0.360	0.100	-0.103	-0.095	322.687	29.052	41.125
Cell18	35053.0	0.300	-0.560	-0.293	-0.163	713.198	29.048	41.121
Cell19	37443.0	0.230	-0.070	-0.136	-0.106	367.763	29.027	41.128
Cell20	37444.0	-0.360	0.100	-0.093	-0.092	328.277	29.033	41.127
Cell21	40143.0	0.040	-0.050	-0.048	-0.075	91.495	28.958	41.024
Cell22	40144.0	-0.140	0.070	-0.118	-0.100	171.576	28.959	41.023
Cell23	40149.0	0.090	-0.160	-0.100	-0.093	201.171	28.957	41.025
Cell24	43078.0	-0.290	-0.240	-0.189	-0.126	152.184	28.964	41.021
Cell25	43079.0	-0.290	-0.110	-0.174	-0.120	116.442	28.964	41.020
Cell26	53665.0	-0.140	-0.120	-0.103	-0.094	45.066	29.024	41.129

Training set arrays

40143
40144
40149
43078
43079
53665

 x error y error distance
average 0,2506 0,129 0,28194

Table 6.7: Encog test sample 5

Training Samples and Graphics

Matlab Samples

Above we have seen the results of the training. However, it is important to see the results visually. We can see the user locations on the Excel graphic. They are coordinates of the user mobile phone signals. Besides, we can see the coordinates of the base station location. Furthermore, we can see the prediction of the base station that program made. With these information, analysis can be made. The distance between the real base station location and the estimated base station location can be seen on the graphic. Moreover, the user location distribution area can be seen. The user distribution and the distance error of base station can give an idea about the performance of the program. Below are some estimations made by matlab and java encog program.

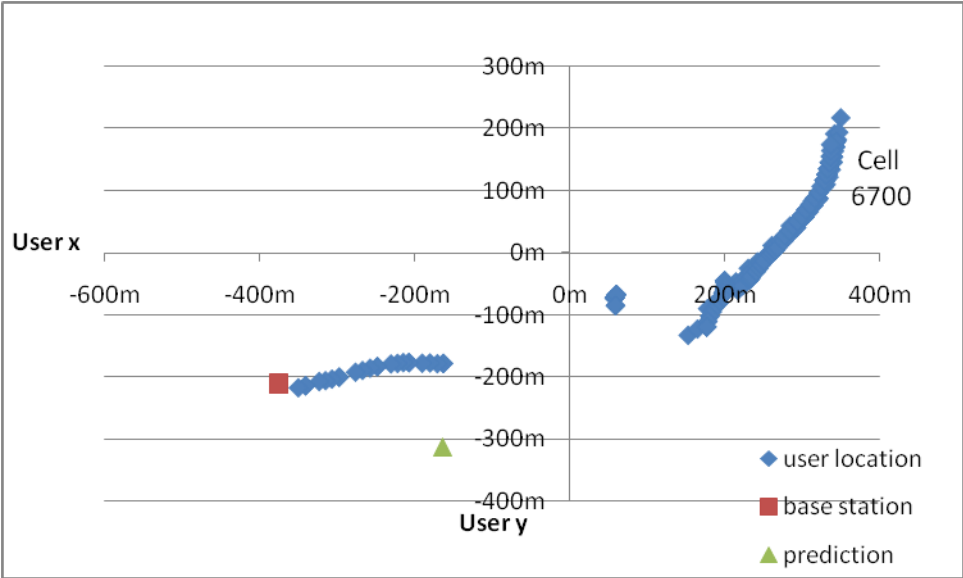


Figure 6.2: Cell 6700 matlab estimation map

User location is distributed in a 600x400 m² square area where the distance error between the real base station and predicted base station is around 234 m. Base station is represented by a square, predicted base station is represented by a triangle and user locations are represented by kites.

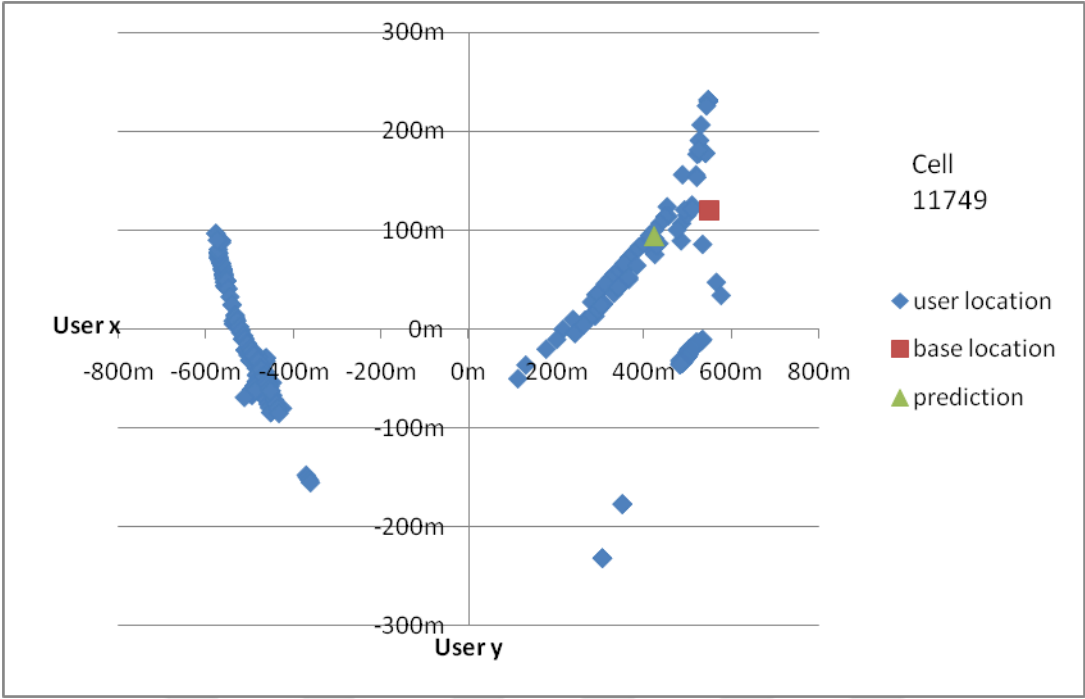


Figure 6.3: Cell 11749 matlab estimation map

User location is distributed in a 800x300 m² square area where the distance error between the real base station and predicted base station is around 139 m. Base station is represented by a square, predicted base station is represented by a triangle and user locations are represented by kites.

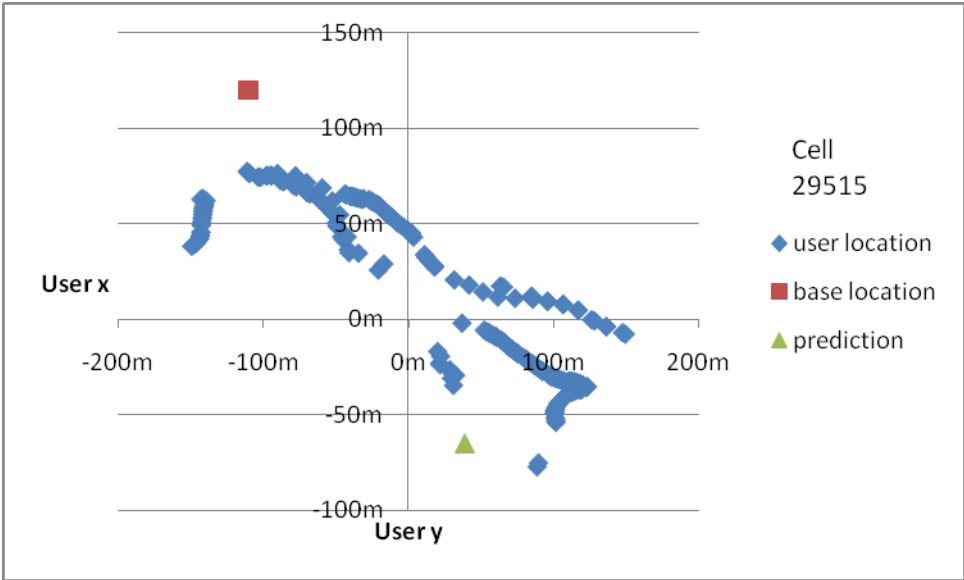


Figure 6.4: Cell 29515 matlab estimation map

User location is distributed in a 200x150 m² square area where the distance error between the real base station and predicted base station is around 238m. Base station is represented by a square, predicted base station is represented by a triangle and user locations are represented by kites.

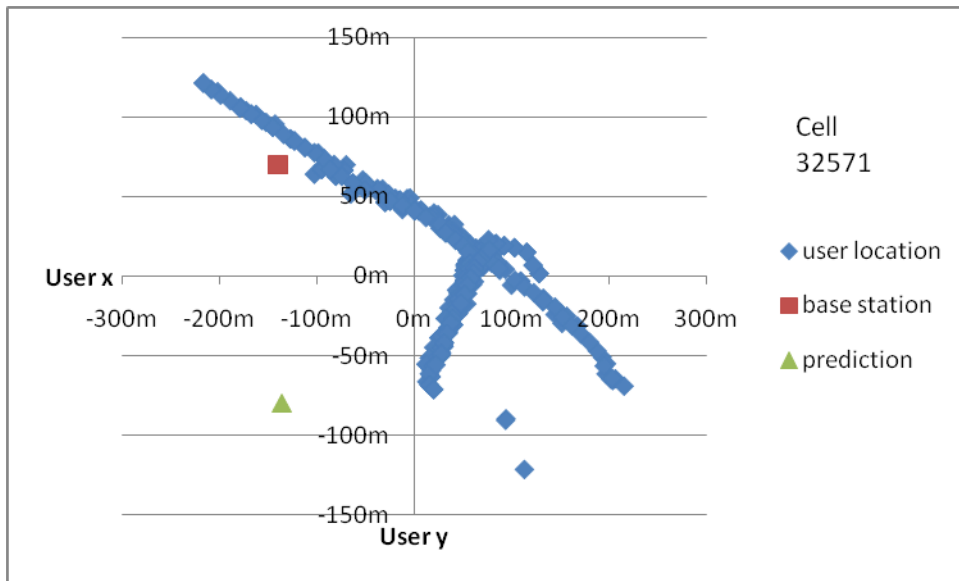


Figure 6.5: Cell 32571 matlab estimation map

User location is distributed in a $300 \times 150 \text{ m}^2$ square area where the distance error between the real base station and predicted base station is around 151 m. Base station is represented by a square, predicted base station is represented by a triangle and user locations are represented by kites.

Java Encog Samples and Graphics

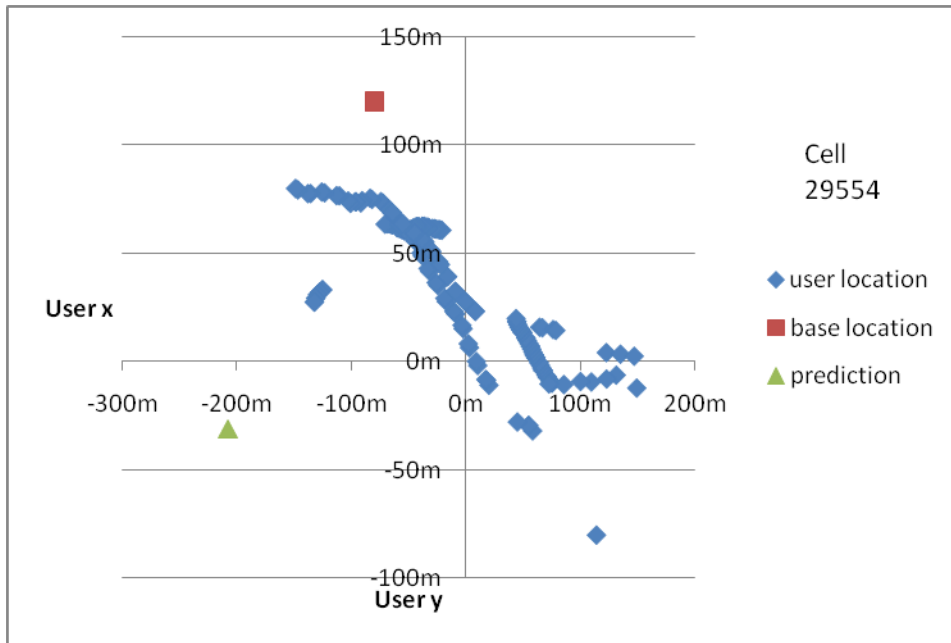


Figure 6.6: Cell 29554 encog estimation map

User location is distributed in a $150 \times 300 \text{ m}^2$ square area where the distance error between the real base station and predicted base station is around 198 m. Base station is represented by a square, predicted base station is represented by a triangle and user locations are represented by kites.

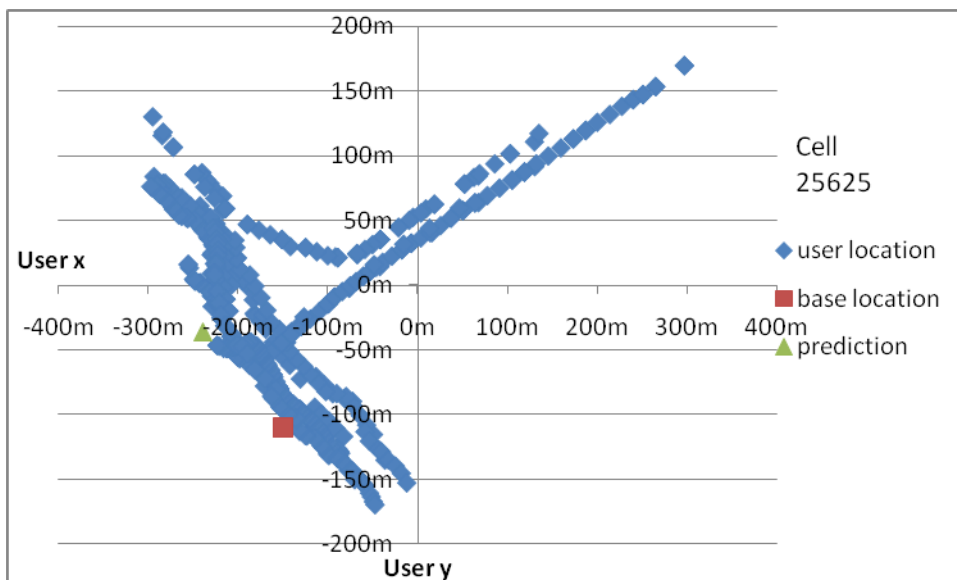


Figure 6.7: Cell 25625 encog estimation map

User location is distributed in a $400 \times 200 \text{ m}^2$ square area where the distance error between the real base station and predicted base station is around 115 m. Base station is represented by a square, predicted base station is represented by a triangle and user locations are represented by kites.

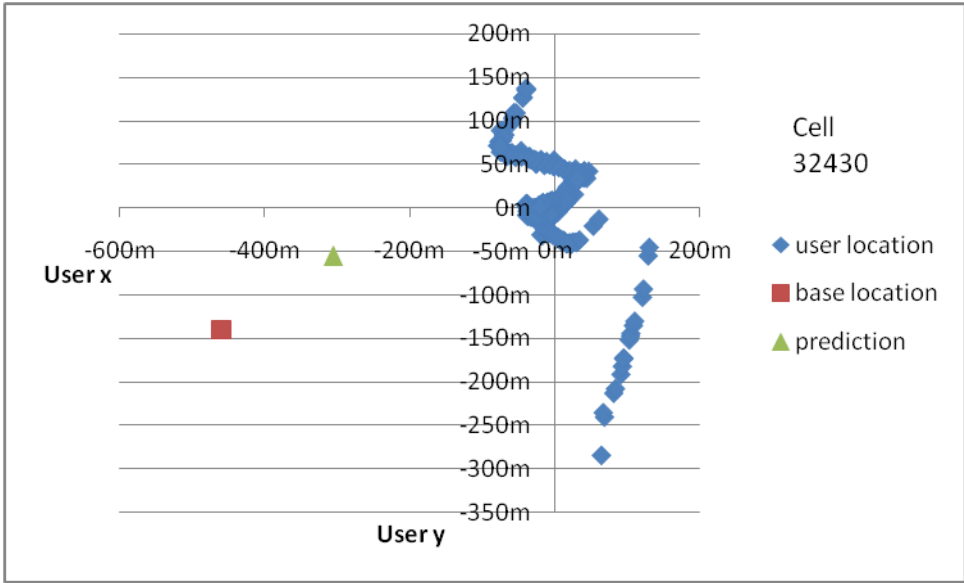


Figure 6.8: Cell 32430 encog estimation map

User location is distributed in a 600x350 m² square area where the distance error between the real base station and predicted base station is around 176 m. Base station is represented by a square, predicted base station is represented by a triangle and user locations are represented by kites.

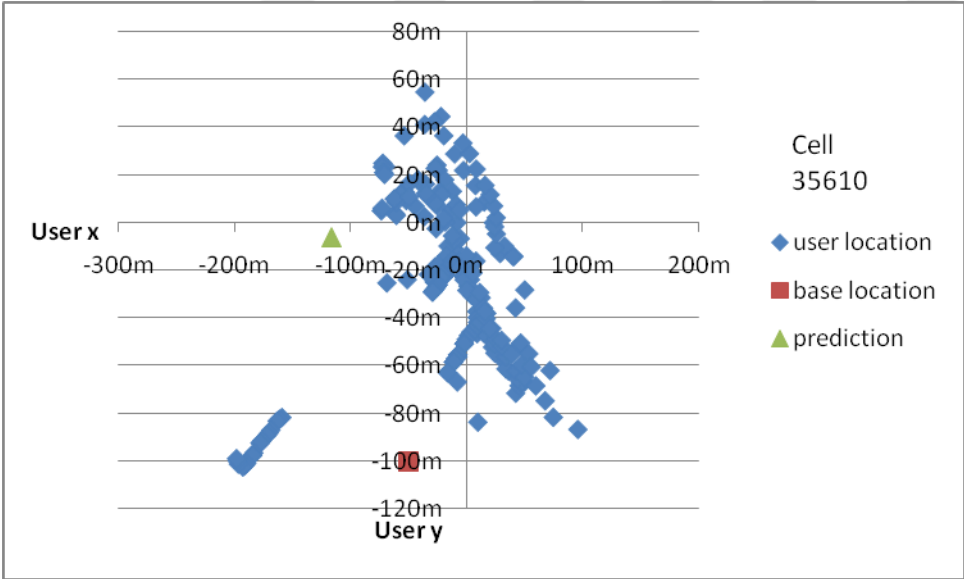


Figure 6.9: Cell 35610 encog estimation map

User location is distributed in a 300x120 m² square area where the distance error between the real base station and predicted base station is around 114 m. Base station is represented by a square, predicted base station is represented by a triangle and user locations are represented by kites.

Comparisons

Levenberg-Marquardt Algorithm

Levenberg–Marquardt algorithm (LMA), is also named as damped least-squares(DLS). LMA solves non-linear least squares problems. Least squares curve fitting problems especially contain minimization problems. LMA is used in programs for solving curve fitting problems. The LMA is an intermediate method. It uses both the Gauss-Newton algorithm(GNA) and the method of gradient descent. LMA algorithm is used as a backpropagation algorithm in training feedforward neural networks. LMA uses SSE(Sum of squared error) for error method.

Resilient Propagation Algorithm

Resilient Propagation also known as Rprop is a heuristic and it is used in feedforward neural networks as a supervised learning method. Sign of the partial derivative is used in Rprop and it acts on each weight separately. For each weight, if a sign change is detected compared to previous iteration in the partial derivative of error function, the update value for that weight is multiplied by a number smaller than 1. If there is no sign change than the update value is multiplied by a number greater than 1. This way total error function is minimised by changing update values opposite direction of that weight's partial derivative. Rprop uses MSE(Mean squared errors) for error method

Similarity between two algorithms

Rprop algorithm is one of the fastest backpropagation training algorithm besides Levenberg-Marquardt algorithm .

Differences between two algorithms

1. Two algorithms have different error calculation methods. LMA uses SSE(Sum of Squares of Error) method whereas Rprop uses MSE(Mean Squares of Error).
2. Two algorithms have different way of updating weights. Resilient propagation updates weights with partial derivative sign change whereas Levenberg-Marquardt updates weights with gradient descent method when parameters are far from their optimal value and Gauss-Newton method when parameters are close to their optimal value.
3. Levenberg-Marquard is best at small neural networks in non-linear least square problems whereas Rprop is good at large neural networks.

Comparisons

Resilient Propagation

Below training samples are discussed. CellIds are listed and each distance of cellids can be seen. Distance is the difference between the program's base location estimation and real base station location. The lesser the distance, the better results we have. These trainings are made with Resilient Propagation method. These trainings are made with 5,8 and 10 hidden layer neurons respectively. Neural networks are trained with 29515, 32430, 32571, 35050,35610, 37442 cells in Encog.

Resilient CellId	Propagation Distance			Resilient CellId	Propagation Distance		
	5neuron	8neuron	10neuron		5neuron	8neuron	10neuron
11749.0	646.830	727.242	751.102	32431.0	189.223	202.224	196.074
17611.0	677.471	709.811	759.606	35053.0	129.918	133.493	121.079
25617.0	149.162	148.954	133.816	37443.0	320.016	282.368	341.025
26297.0	173.407	171.807	127.780	37444.0	181.784	196.035	185.371
29515.0	143.812	133.540	129.843	40143.0	220.881	201.960	189.910
32430.0	210.135	147.016	191.045	40144.0	160.150	157.807	109.273
32571.0	87.049	79.695	77.967	40149.0	239.575	230.527	252.602
35050.0	112.905	115.539	68.401	43078.0	248.512	274.387	287.779
35610.0	152.743	146.711	159.518	43079.0	462.663	484.238	341.168
37442.0	146.785	164.185	126.584	53665.0	122.564	126.714	125.731
6701.0	383.410	410.115	398.323	40141.0	95.493	94.117	97.800
25625.0	107.390	96.993	123.112	40142.0	229.393	225.245	235.019
25627.0	330.377	307.877	314.423	52121.0	233.967	201.758	213.381
29554.0	195.717	177.657	179.324	52123.0	318.984	296.866	296.148
29555.0	176.086	165.437	155.173	55073.0	105.477	74.738	82.481
Average Distance				5neuron	8neuron	10neuron	
				231	229	225	

Table 6.8: Resilient Propagation Comparisons

Levenberg Marquard

Below training samples are discussed. CellIds are listed and each distance of cellids can be seen. Distance is the difference between the program's base location estimation and real base station location. The lesser the distance, the better results we have. These trainings are made with Levenberg Marquard method. These trainings are made with 5,8 and 10 hidden layer neurons respectively. Neural networks are trained with 29515, 32430, 32571, 35050,35610, 37442 cells.

Levenberg CellId	Marquard Distance		
	5 neuron	8 neuron	10 neuron
6700.0	281.373	153.485	281.372
11749.0	693.250	700.266	693.243
17611.0	746.051	706.715	746.048
25617.0	140.976	139.411	140.976
26297.0	136.876	134.434	136.886
29515.0	165.230	133.467	165.230
32430.0	281.373	152.069	281.372
32571.0	107.420	72.965	107.421
35050.0	100.751	95.042	100.752
35610.0	182.153	147.851	182.155
37442.0	166.853	150.015	166.851
6701.0	354.675	413.905	354.675
25625.0	114.672	115.572	114.674
25627.0	323.049	292.019	323.051
29554.0	184.847	168.016	184.847
29555.0	146.743	165.665	146.739

Levenberg CellId	Marquard Distance		
	5 neuron	8 neuron	10 neuron
32431.0	201.893	209.664	201.893
35053.0	121.622	123.962	121.767
37443.0	324.930	294.396	324.920
37444.0	194.747	207.420	194.740
40143.0	252.341	191.574	252.343
40144.0	115.357	92.025	115.358
40149.0	286.409	231.266	286.406
43078.0	241.860	295.863	241.861
43079.0	318.715	353.524	318.413
53665.0	128.452	125.600	128.454
40141.0	132.815	89.175	132.817
40142.0	281.275	230.300	281.277
52121.0	228.807	188.398	228.808
52123.0	330.185	280.154	330.187
55073.0	113.343	64.176	113.343

		5 neuron	8 neuron	10 neuron
average	distance	238	216	238

Table 6.9: Levenberg Marquard Comparisons

Resilient Propagation comparisons

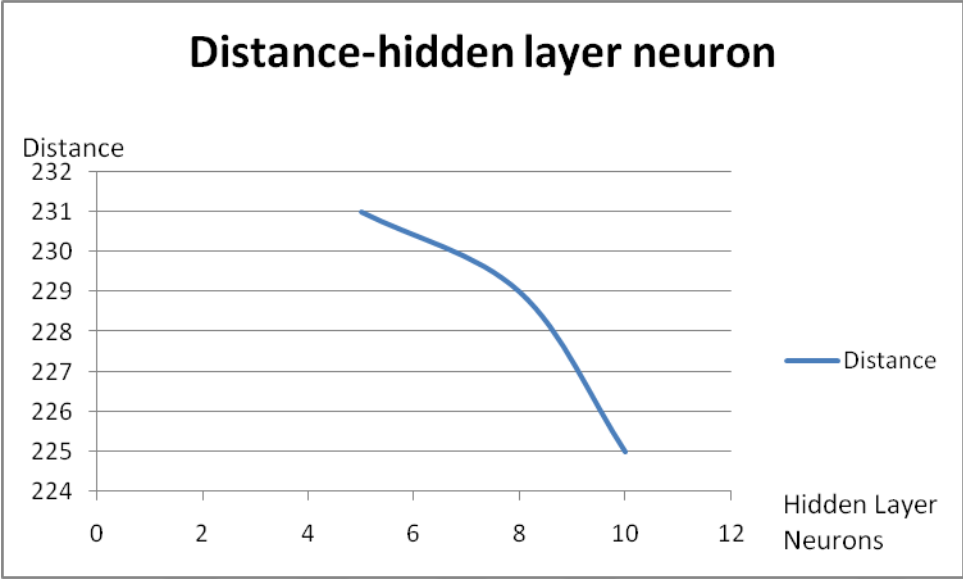


Figure 6.2:Distance-Hidden Layer Comparisons in Resilient Propagation

As the figure shows, as hidden layer neurons increase, distance decreases in the Resilient Propagation method.

Levenberg Marquard comparisons

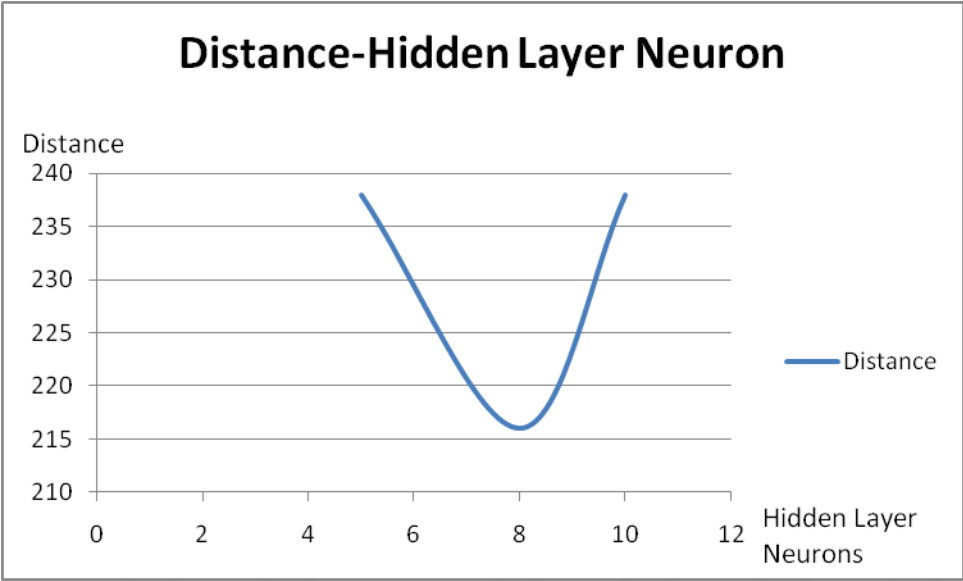


Figure 6.3:Distance-Hidden Layer Comparisons in Levenberg Marquard

As the figure shows, as hidden layer neurons increase from 5 to 8, distance decreases. However, if neurons increase from 8 to 10 distance increases in the Resilient Propagation method.

Comparison Conclusion

Comparisons are made with same training and test sets with Resilient Propagation and Levenberg-Marquard methods. As a conclusion, Resilient Propagation is better than Levenberg-Marquard training slightly. And optimal hidden layer neuron number is 8 in Levenberg-Marquard but 10 in Resilient Propagation. This shows Levenberg-Marquard is good at less neurons and Resilient Propagation is good at more hidden layer neurons.

Conclusion

In this thesis, location estimation of base station is explained. This is realized by using artificial neural networks and machine learning methods(self learning,training).

Firstly, former related works are investigated. Geolocation techniques are examined. The theory of geolocation is studied. Different technologies are examined. After that various implementations are examined.

After that, machine learning theories are examined. This involves training methods. Supervised and unsupervised learning are examined. Supervised backpropagation methods are examined. Performance and usage area of methods are discussed.

Then, algorithm of the base estimation is discussed. Plan of the project is discussed.

Data collection is discussed next. This is a difficult job. This is realized by java parser program. Later data is made neat by database applications. The source code of program is also discussed.

Later, java implementation of the program is explained. The algorithm and source code of the program is discussed. The manual of the program is discussed.

Finally,tests and results are discussed. A lot of tests are made in the production of the program. Best results are discussed. Tests in matlab and java are discussed. Performance and results of the test are discussed.

As a conclusion, the program estimates the locations successfully and consistently. The estimated locations are close to the real locations. Program gives more accurate results when inputs are similar to training sets. Though matlab estimations are slightly better than the program. Results accuracy is based on the training sets. Better training sets give better and more accurate results. Results are also optimal with a good training algorithm and a good artificial neural network topology.

References

- 1] Bulusu, Nirupama, John Heidemann, and Deborah Estrin. "GPS-less low-cost outdoor localization for very small devices." *Personal Communications, IEEE 7.5 (2000): 28-34.*
- 2] V. Zeimpekis, G.M. Giaglis, and G. Lekakos, "A Taxonomy of Indoor and Outdoor Positioning Techniques for Mobile Location Services," *SIComExch.*, vol. 3, no. 4, pp. 19–27, 2003.
- 3] M. Hazas, J. Scott, and J. Krumm, "Location-Aware Computing Comes of Age," *Comput.*, vol. 37, no. 2, pp. 95–97, 2004.
- 4] A. Kupper, "Location-Based Services: Fundamentals and Operation." Chichester: Wiley, 2005.
- 5] T. S. Rappaport, *Wireless Communications - Principles and Practice*, Prentice Hall PTR, 1996.
- 6] B. Hofmann-Wellenhof, H. Lichtenegger, and J. Collins, *Global Positioning System: Theory and Practice*, Fourth Edition, SpringerVerlag, 1997.
- 7] M. Ibrahim and M. Youssef, "A Hidden Markov Model for Localization Using Low-End GSM Cell Phones," in *Proc. 2011 IEEE Int. Conf. on Communications (ICC)*, Cairo, Egypt, 2011, pp. 1–5.
- 8] J. Paek, K. Kim, J.P. Singh, and R. Govindan, "Energy-Efficient Positioning for Smartphones using Cell-ID Sequence Matching," in *Proc. 9th Int. Conf. on Mobile Systems, Applications, and Services*, Maryland, USA, 2011, pp. 293–306.
- 9] Mobile Location Estimation Based on Differences of Signal Attenuations for GSM Systems/ Ding-Bing Lin ; Inst. of Comput. & Commun., Nat. Taipei Univ.

of Technol., Taiwan ; Juang, R.-T./ Vehicular Technology, IEEE Transactions on (Volume:54 , Issue: 4)

10] An Analysis of Base Station Location Accuracy within Mobile-Cellular Networks, Liam Smit , Adrie Stander , Jacques Ophoff, International Journal of Cyber-Security and Digital Forensics (IJCSDF) 1(4): 272-279 /The Society of Digital Information and Wireless Communications (SDIWC) 2012 (ISSN: 2305-0012)

11] J. Borkowski, "Performance of Cell ID+RTT Hybrid Positioning Method for UMTS," M. Sc. thesis, Tampere University of Technology, Finland, 2004.

12] H. Holma and A. Toskala "WCDMA for UMTS: Radio Access for Third Generation

13] Farhad . E .Mahmood,Ahmad M. A. Salama," Mobile Positioning System using Signal Strength Measurement for WCDMA System",Al-Rafidain Engineering Vol.19 No.1 February 2011.

14]Neural Networks User's Toolbox User's Guide 6/Howard Demuth ,Mark Beale ,Martin Hagan

15]A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm Riedmiller, M. Braun, H. Neural Networks, 1993., IEEE International Conference on (28 Mar 1993-01 Apr 1993)

[16] D. E. Rumelhart and J. McClelland. Parallel Distributed Processing. 1986.

17] <http://www.mathworks.com>

Curriculum Vitae

Ramazan Cengiz was born on 17 May 1986, in Istanbul. He received his BS degree in Computer Engineering in 2012 from Kadir Has University. He worked as a research assistant at the department of Computer Engineering of Kadir Has University between 2012 – 2013. Then he has been working as a Software Engineer at Kadir Has University/Vodafone project since 2014. During his master education, he has been affiliated with the Artificial Neural Networks. His research interests include data mining, relational databases and java technologies.

