UNIVERSITY
OF GÄVLE

DEPARTMENT OF TECHNOLOGY AND BUILT ENVIRONMENT

Checking the integrity of Global Positioning Recommended Minimum (GPRMC) sentences using Artificial Neural Network (ANN)

Tayyab Hussain

August 2009

Master's Thesis in Geomatics

Supervisor: Dr. Hans Hauska
Examiner: Prof. Anders Östman

Abstract

In this study, Artificial Neural Network (ANN) is used to check the integrity of the Global Positioning Recommended Minimum (GPRMC) sentences. The GPRMC sentences are the most common sentences transmitted by the Global Positioning System (GPS) devices. This sentence contains nearly every thing a GPS application needs. The data integrity is compared on the basis of the classification accuracy and the minimum error obtained using the ANN. The ANN requires data to be presented in a certain format supported by the learning process of the network. Therefore a certain amount of data processing is needed before training patterns are presented to the network. The data pre processing is done by the design and development of different algorithms in C# using Visual Studio.Net 2003. This study uses the BackPropagation (BP) feed forward multilayer ANN algorithm with the learning rate and the momentum as its parameters.
The results are analyzed based on different ANN architectures, classification accuracy, Sum of Square Error (SSE), variables sensitivity analysis and training graph. The best obtained ANN architecture shows a good performance with the selection classification of 96.79 % and the selection sum of square error 0.2022. This study uses the ANN tool Trajan 6.0 Demonstrator.

Keywords: Global Positioning System (GPS), Global Positioning Recommended Minimum (GPRMC), Artificial Neural Network (ANN), Back-Propagation (BP)

Acknowledgments

I would like to thank ALMIGHTY ALLAH for helping me to finish my thesis.

I would like to dedicate my work to my parents for always supporting and believing in my abilities, without them I cannot accomplish any thing in my life.

I would like to thank especially my father Matloob Hussain and brother Moazzam Hussain for their constant support throughout my thesis work; and editing and revising my research.

I would like to thank my supervisor Dr. Hans Hauska for his encouragement from the beginning and enthusiastic support throughout the thesis. I can never thank him enough for his academic support and help.

I would like to thank Mr. Jim Förlander, Market Imperator, GeoIntel AB for providing professional working environment in the company and research data for my thesis work.

I would like to thank Prof. Ander Östman for providing thesis idea to work on artificial neural network.

I would like to thank my academic teachers Dr. Anders Brandt, Prof. Anders Östman, Stig-Göran Mårtensson , Pia Ollert Hallqvist and lab assistants Peter Fawcett, Lise Lotte Dahlberg for my Geomatics studies in the University of Gävle, Sweden.

I would like to thank my friends Abdul Waqas, Willems Tims, Chris, Ioannis Tolika, Maryam Kordi, Saman Tavakoli, Salahaldin Shoshtari for making my studies enjoyable in the University of Gävle, Sweden.

Table of Contents

1. Introduction

The Artificial Neural Network (ANN) has seen a lot of interest from past few years. It has been successfully applied to a wide range of domains such as finance, medicine, engineering, geology and physics. The ANN is often used for solving problems of prediction, classification and control. This study will investigate the ANN classification technique to check the integrity of the Global Positioning Recommended Minimum (GPRMC) sentences.

1.1. Problems and Objectives

The National Marine Electronic Association (NMEA) has developed specification for the communication between marine equipment such as sonars, anemometers, gyrocompasses, Global Positioning System (GPS) receivers and many other types of marine instruments. There are several types of GPS sentences that provide different information like GLL (latitude and longitude data), RMC (recommended minimum data for GPS), GSA (overall satellite data), etc.

The GPS receiver sends information to computers and marine equipment in the form of sentences like GLL, GSA and RMC etc. Organizations that involve monitoring thousands of GPS receivers get millions of GPS sentences in a minute. When using the data, checking the integrity of each GPS sentence is a crucial task.

The GPRMC sentence is the most common data transmitted by the GPS receivers. The GPRMC sentence is composed of thirteen data items separated by comma (,). Each data item represents unique information. The GPRMC sentence error can be either in any or all of the thirteen data items. The GPRMC sentence errors can be missing values, incorrect positions, incorrect directions, incomplete sentences and erroneous values. The data integrity will be compared on the basis of the classification accuracy and the minimum error obtained using the ANN classification technique.

This study aims to test hypothetical deductive research methodology for checking the integrity of GPRMC sentences. The ANN requires the GPRMC sentences to be presented in a certain format supported by the learning process of the network. Therefore, the data processing is necessary to convert variables to such a form that the ANN can best utilize them. The data pre processing is done by the design and development of different algorithms in C# using the Visual Studio.Net 2003. These algorithms prepare the GPRMC sentences according to the ANN requirements. This study uses the BackPropagation (BP) feed forward multilayer ANN algorithm with the learning rate and the momentum as its parameters. The results are analyzed based on the different ANN architectures, classification accuracy, Sum of Square Error (SSE), variables sensitivity analysis and training graph. This study will use the ANN tool Trajan 6.0 Demonstrator.

1.2. Previous studies

The ANN classification means assigning data (cases or observations) to one of a number of classes based on their characteristics. If the data satisfies a certain set of criteria then it is assigned to the class that corresponds to those criteria (Mas & Flores, 2008). Several studies have been made to measure the accuracy of the ANN classification in variety of ways. Some of the history of researches in the context of measuring the ANN classification accuracy is as follows.

In medicine, Chiu, et al. (2008) used the ANN to extract and classify characteristics of arterial pulse waves in diabetics and heypertensive patients, and the results revealed that there are 13 kinds of arterial pulse waveforms in healthy subjects with the classification accuracy of 73% diabetics and 90% hypertensive patients. Erol, et al. (2008) used the ANN for the structural classification of thyroid diseases, and the experimental results showed that the predictions of ANN models are very satisfying for learning data sets. Kantardzic, et al. (2002) used the ANN classification in data mining to improve the Polycythemia Vera diagnosis. Pasquier and Hamodrakas (1999) achieved the classification accuracy of 100% by classifying transmembrane proteins using the hierarchical ANN. Dey, et al. (2008) used the ANN classification technique for diagnosing diabetes mellitus.

In engineering, Mudroch, et al. (2009) used ANN for the weather classification by remote and local atmosphere sensing using different RF propagation. Palade, et al. (2003) used the ANN classification to predict the toxicity of chemicals with respect to Quantitative Structure-Activity Relationship (QSAR). Zhichen, et al. (2009) achieved the ANN classification accuracy of 78% by classifying the weed species based on color leaf texture feature. Wei, et al. (2009) performed the comparison study using two classifiers, the Maximum Likelihood Classifier (MLC) and the ANN to identify selected internal wood, and found that the ANN produced high classification accuracy (97.6%) compared with the MLC (80.9%).

In geology, Müller and Eagles (2007) performed the seabed ANN classification based on shallow water backscatter mosaics and the results show that the ANN classified successfully unknown portions of the backscatter mosaic ranging from 77% to 92%. Marsh and Brown (2008) performed the ANN based seabed classification scheme using the beam level angular response of backscatter strength and the bathymetric attributes, and the results showed that the competitive ANN provide the most accurate method for clustering swath acoustic data. Besaw, et al. (2009) performed stream classification using the ANN and the results show that the ANN provide standardized approach for the adaptive watershed management to classify the river network in various contexts.

The ANN is an excellent tool to develop real-world applications. The learning ability from data, classification capabilities, generalization and noise tolerance are few advantages that can be successfully used in this study to check the integrity of the GPRMC sentences (Pasquier and Hamodrakas, 1999).

1.3. Organization of the thesis

This study is organized in the following manner. Chapter one presents an introduction to the research background. It outlines the importance of ANN and its classification technique, a brief history of the researches in the context of the ANN classification, and

the introduction to the GPRMC sentence and its errors. Furthermore, the focus of the study is described, including the scope and the objectives.

Chapter two provides important guidelines for the design of ANN. It presents an introduction to the ANN, the network architectures and the activation functions. It describes in detail the working of backpropagation algorithm and its parameters, the learning rate and the momentum. Topics like epoch, gradient descent, training modes and sum of square error are discussed in detail.

Chapter three presents detail about data preparation of the GPRMC sentence for the ANN modeling. It presents detail about study area, data description and data coding. The details about different kinds of errors in the GPRMC sentences are discussed. A brief overview of the different features of ANN packages is also discussed.

Chapter four presents the building of ANN. It shows a flow diagram that describes different steps involved in the design of ANN for processing the GPRMC sentence. Some of the screen shots of the implementation of Trajan 6.0 Demonstrator for the ANN modeling of the GPRMC sentences are also shown.

Chapter five discusses the results of the study. The results are analyzed based on the different ANN architectures, the classification accuracy, the sensitivity analysis and the training graph.

Chapter six presents the discussions and the chapter seven draws the conclusions. Finally, chapter eight points to possible future work.

2. Introduction to Artificial Neural Network (ANN)

Artificial Neural network (ANN) was developed to understand the biological nervous system, such as how brain processes information and emulates its strength. The human brain consists of a large number of interconnected cells called the neurons, units, cells or nodes. Neurons provide the ability to remember, think and apply our previous experiences to every action (Djarfour et al., 2008).

In an ANN, the neurons are connected to each other by a communication link where each link has an associated weight. The weight represents information that is used to solve the problem. Each neuron has an internal state called activation function or activity level. The neuron applies the activation function to its input to determine the output signal (Fausett, 1994).

For example, consider a neuron Y as shown in Fig. (1). It receives inputs X_1, X_2 and X_3 with respective weights w_1, w_2, w_3. Let x_1, x_2, and x_3 are the respective activations (output signals). The total input (y_in) is the sum of the product of weight and activation of each input as shown in Eq. (1) (Fausett, 1994).

$$y\_in = w_1x_1 + w_2x_2 + w_3x_3 \qquad (1)$$

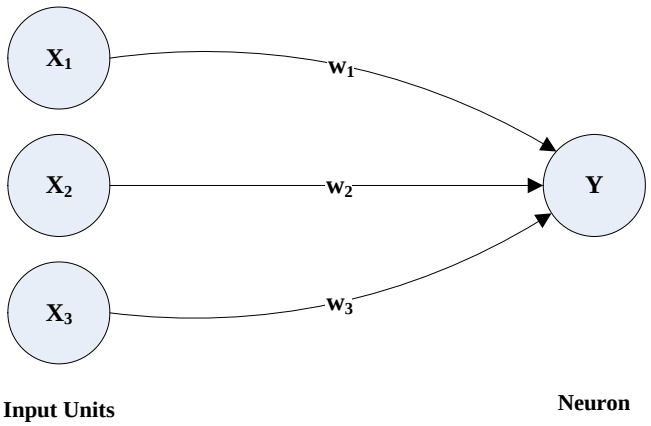

Fig. 1. A simple artificial neuron (Fausett, 1994)

Let a sigmoid function (an S-shaped curve) act as the activation function at neuron Y. Mathematically this is shown in Eq. (2) (Fausett, 1994). Details on activation functions are presented in Section (2.5).

$$f(x) = \frac{1}{1 + \exp(-x)} \qquad (2)$$

Suppose the neuron Y is connected to neurons Z_1 and Z_2 with weights v_1 and v_2 as shown in Fig. (2). The neuron Y sends its input signal y to each Z_1 and Z_2. However the values at Z_1 and Z_2 will be different depending on the weights v1 and v2 respectively.

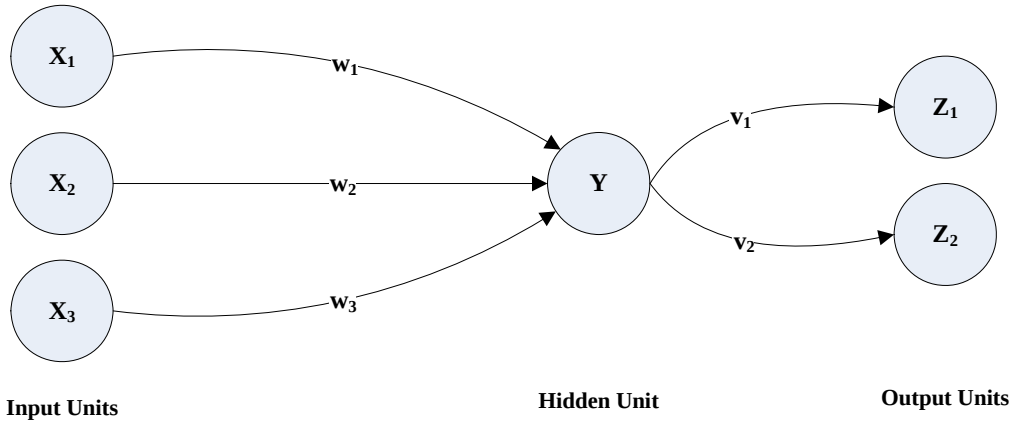

Fig. 2. Simple neural network (Fausett, 1994)

Fig. (2) shows a simple ANN involving the input layer, the hidden layer and the output layer. The input layer gets data from the outside world. The output layer predicts the output and passes it to the outside world. The hidden layer extracts and remembers useful features from the input layer and predicts the outcome to the network through output layer.

Each layer consists of sufficient number of neurons according to application requirement. The number of neurons in the input, hidden and the output layer is problem specific and there is no specific rule for selection (Rafiq et al., 2001; Haykins, 2001; Kanungo et al, 2006).

The arrangement of layers (architecture of the network) and the training of the network (assigning weights to each link) is the crucial task in creating an effective ANN for

addressing complex problems (Fausett, 1994). Architecture and training of ANN are explained in more detail in Sections (2.1) and (2.2) respectively.

2.1. Network Architecture

The arrangement of neurons into layers and the connection patterns within and between layers is called network architecture. The ANN is classified as single and multilayer network as shown in Fig. (3) and (4).

Fig. (3) and (4) show examples of feedforward networks. In the feedforward network, signals flow from the input layer to the output layer in a forward direction (Fausett, 1994). The ANN is said to be fully connected if every neuron in the layer is connected to neurons of the adjacent forward layer; if only some of the neurons are connected then the network is called a partially connected network (Haykin, 1994).

2.1.1. Single-Layer Network

A network in which the neurons are organized in the form of layers is called *layered network*. A single layer network is the simplest form of a layered network in which input signals flow directly to the output layer. This network is strictly of type feedforward (Haykin, 1994). It contains one layer of connections weights (Fausett, 1994) as shown in Fig. (3).

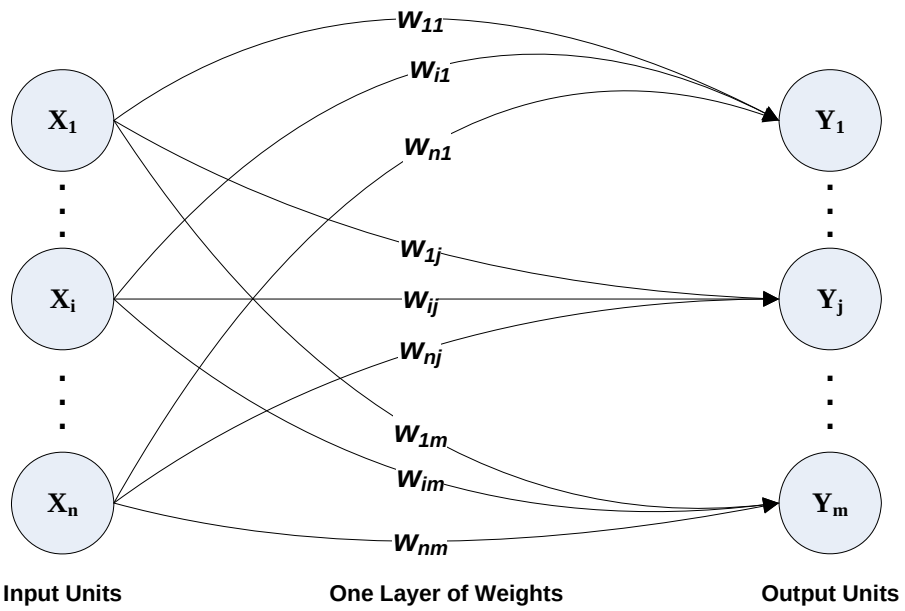

Fig. 3. A single-layer neural network (Fausett, 1994)

2.1.2. Multilayer Network

A multilayer network consists of one or more layers (so called hidden layers) between the input and the output layer. The function of a hidden layer is to connect the input layer to the output layer. Multilayer networks are used where the problem involves high order complicated mappings, but training of these networks may be more difficult than the single layer network (Haykin, 1994; Fausett, 1994). Fig. (4) shows an example of multilayer feedforward network.

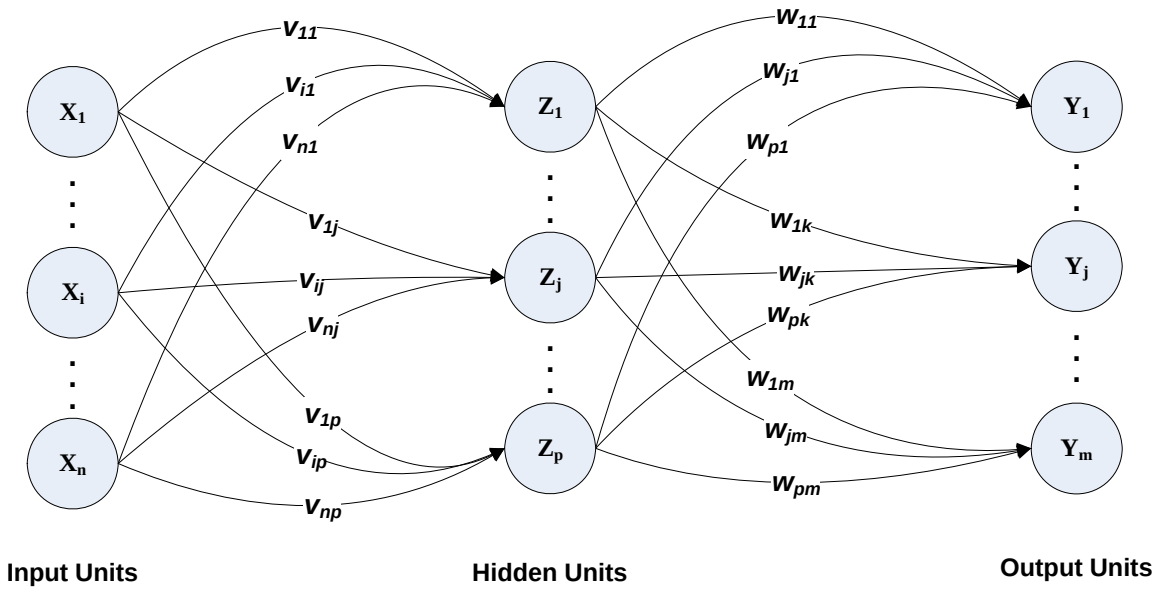

Fig. 4. A multilayer neural network (Fausett, 1994)

2.2. Setting the Weights/Training/Learning

Setting the weights (training) of the network is the important characteristic that distinguishes networks from each other. Weights between the two layers are updated by taking the derivative of the upper unit and the activation of the lower unit. The training of the network starts by selecting the initial weight. It is important to select the initial weight so that the activation or the derivative of the activation does not become zero. It is common practice to choose the weight or bias randomly between (-0.5) and (+0.5) or (-1) and (+1) (Fausett, 1994).

Choosing the right kind of training depends on the type of problem. There are two types of training i.e. supervised and unsupervised training.

2.2.1. Supervised Training

In supervised training, the network is presented with a sequence of training vectors and patterns associated with target vector (Fausett, 1994). Weights are adjusted under the combined influence of the training vector and an error signal (difference between actual and desired response). Weights are adjusted in a step-by-step fashion with the aim for the ANN to emulate the desired or target response (Haykin, 1994). This kind of training is called supervised training.

2.2.2. Unsupervised Training

In unsupervised training, input vectors showing similar characteristics are grouped together without the use of training data. In this training, input vectors are provided without specifying target output vector. The ANN adjusts the weight so that each input vector is assigned to the same output (or cluster) unit (Fausett, 1994).

2.3. Activation Functions

Activation functions transform the net input to a neuron into its activation (Fausett, 1994). The ANN use variety of activation functions such as linear, logistic, hyperbolic tangent or exponential functions etc. Some of the activation functions are explained below.

2.3.1. Identity function

The identity function returns the same value that was used in its argument as shown in Fig. (5). Mathematically it is shown in Eq. (3)(Fausett, 1994).

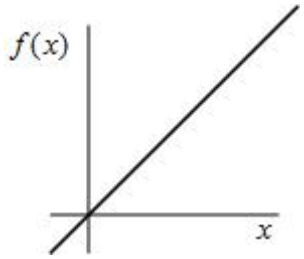

Fig. 5. Identity function (Fausett, 1994)

$$f(x) = x \qquad \text{for all } x \qquad (3)$$

2.3.2. Threshold function

The threshold function is used to convert the net input in a continuously valued variable to an output unit that is either a binary (1 or 0) or a bipolar signal (1 or -1). It is also known as binary function or Heaviside function. See Fig. (6).

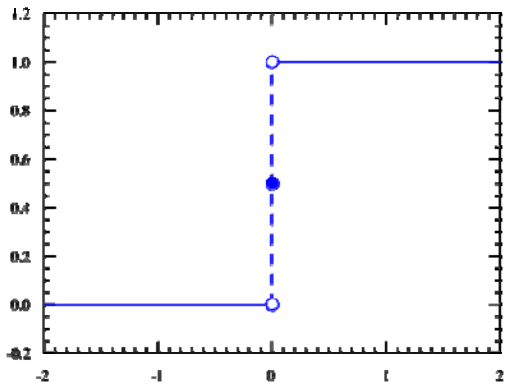

Fig. 6. The Heaviside step function (Heaviside Step Function, 2009)

2.3.3. Sigmoid function

Sigmoid functions (S-shaped curves) are the most common form of activation functions. Logistics function and hyperbolic tangent functions are the most common forms of sigmoid functions used in ANN. It is advantageous to use because the relationship between the value of the function at a point and the value of the derivative at a point reduces computational burden during training. If the output range is between 0 and 1 then it is called a binary sigmoid function or logistic function as shown in Fig. (7) for the two values of steepness parameter σ. A binary sigmoid is shown in Eq. (4) (Fausett, 1994).

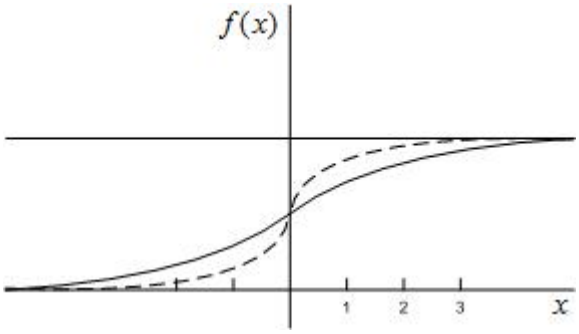

Fig. 7. Binary sigmoid. Steepness parameter σ=1 and σ=3 (Fausett, 1994)

$$f(x) = \frac{1}{1 + \exp(-\sigma x)} \qquad (4)$$

A logistic sigmoid function can be scaled to have any range depending upon the problem. If the range of a logistic function is scaled between -1 and +1 then it is called a bipolar sigmoid function as shown in Fig. (8) for $\sigma = 1$. See Eq. (5) (Fausett, 1994).

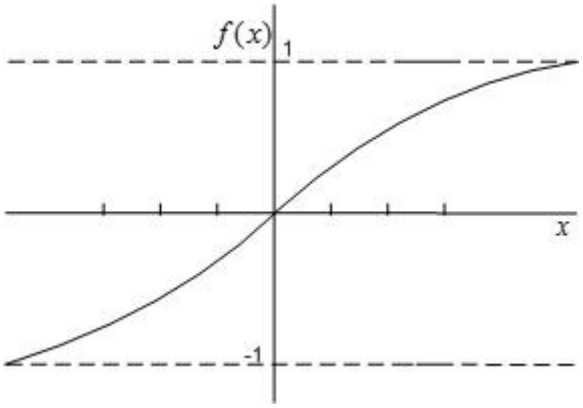

Fig. 8. Bipolar sigmoid (Fausett, 1994)

$$g(x) = 2f(x) - 1 = \frac{2}{1 + \exp(-\sigma x)} - 1$$

$$g(x) = \frac{1 - \exp(-\sigma x)}{1 + \exp(-\sigma x)} \qquad (5)$$

2.4. Backpropagation

Backpropagation came due to the failure of solving XOR (Exclusive OR) problem and the training of multilayer network. Parallel Distributed Processing Group led by psychologists David Rumelhart and James McClelland first publicized the use of back propagation learning and the training of Multilayer Perceptrons (MLP) (Haykin, 1995; Fausett, 1994).

Backpropagation training of a network is divided into three stages (Fausett, 1994):

2. Feedforward of the input training pattern.
3. Calculation and backpropagation of error.
4. Adjustment of weights.

Fig. (9) represents a feedforward ANN with input units (X units), hidden units (Z units) and output units (Y units). W_{0k} and V_{0j} represent bias having output 1 on unit Y_k and Z_j respectively (Fausett, 1994).

During feedforward in stage 1, each input unit $\mathbf{X}_i$ receives input signal and broadcasts to hidden units $\mathbf{Z}_j$. The hidden units compute activation and send signals to output units $\mathbf{Y}_k$. The output units compute activation and produce the output signal of the net from the input pattern (Fausett, 1994).

In stage 2, calculation and backpropagation of error is performed. Each output unit computes its activation y_k and compares it with the target output t_k to determine the associated error. Based on the error, the factor δ_k is computed. The factor δ_k distributes error to the output layer and the hidden layer. The weights are then updated between the output layer and the hidden layer. The hidden layer then computes error factor δ_j, but it is not necessary to distribute error to the input layer. The factor δ_j is only used to update the weights between the hidden and the input layer (Fausett, 1994).

In stage 3, after all the error factors are computed, the weights of all layers are adjusted simultaneously. The adjustment of the weights from the hidden layer to the output layer is based on the error factor and the activation of hidden units. The adjustment of weights from the input layer to the hidden layer is based on the error factor and the activation of the input layer (Fausett, 1994).

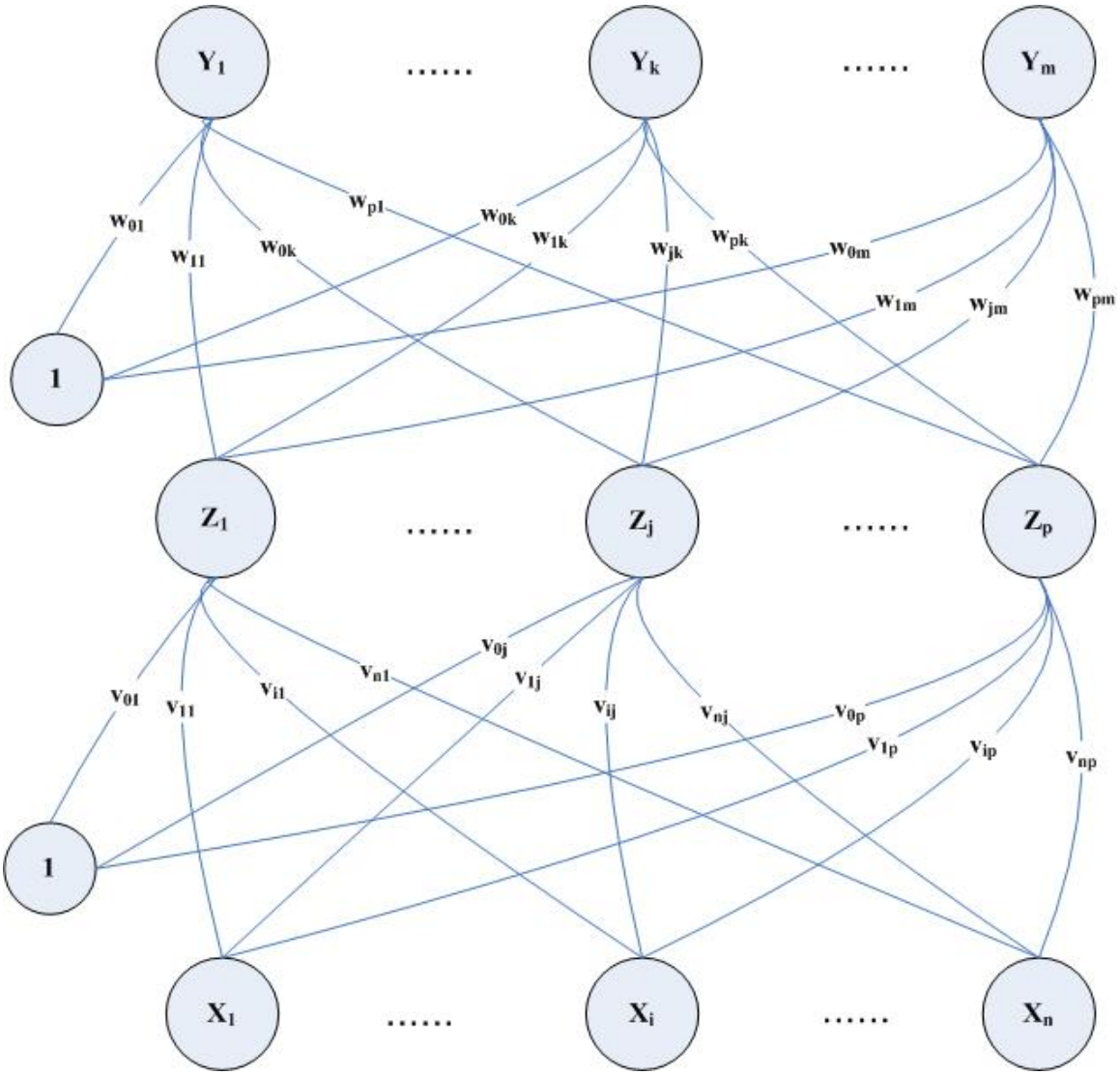

Fig. 9. Backpropagation neural network with one hidden layer (Fausett, 1994)

Table (1) shows the notations used in the backpropagation training algorithm and the detail working of backpropagation algorithm in steps.

Table (1)
Backpropagation training algorithm notations and working (Fausett, 1994)

Notations	
X	The input training vector $\mathbf{x} = \{x_1 ..... , x_i , ... x_n\}$
T	The target output vector $\mathbf{t} = \{t_1 .... , t_k , ... t_m\}$
δ_k	Error backpropagated from output unit to hidden unit.

δ_j	Error backpropagated from hidden unit to input unit.
α	Learning rate.
$\mathbf{X}_i$	Input unit i.
v_{0j}	Bias on hidden unit j.
$\mathbf{Z}_j$	Hidden unit j.
w_{0k}	Bias on output unit k.
$\mathbf{Y_k}$	Output unit k.
Working	
Steps	**Instructions**
0	Initialize weight (Set to small random values)
1	While stopping condition is false, do steps 2-9
2	For each training pair, do steps 3-8
3	Each input layer (X_i, $i = 1,.....,n$) receives input signal x_i and broadcasts these signals to hidden units above.
4	Each hidden units (Z_j, $j = 1,....., p$) sums its weighted input signals, $z\_in_j = v_{0j} + \sum_{i=1}^{n} x_i v_{ij}$, applies its activation function to compute the output signal, $z_j = f(z\_in_j)$, Send output signal to above output layer
5	Each output unit (Y_k, $k = 1,.....,m$) sums its weighted input signal, $y\_in_k = w_{0k} + \sum_{j=1}^{p} z_j w_{jk}$, applies its activation function to compute the output signal, $y_k = f(y\_in_k)$
6	Each output unit (Y_k, $k = 1,.....,m$) receives a target pattern corresponding to the input training pattern, computes its error information term, $\delta_k = (t_k - y_k) f'(y\_in_k)$, calculates its weight correction term (used to update w_{jk} later), $\Delta w_{jk} = \alpha \delta_k z_j$, calculates its bias correction term (used to update w_{0k} later), $\Delta w_{0k} = \alpha \delta_k$, and send δ_k to units in the layer below,
7	Each hidden layer (Z_j, $j = 1,....., p$) sums its delta input from the units in above layer, $\delta\_in_j = \sum_{k=1}^{m} \delta_k w_{jk}$, calculates error information by multiplying with the derivative of activation function, $\delta_j = \delta\_in_j f'(z\_in_j)$, calculates its weight correction term (used to update v_{ij} later), $\Delta v_{ij} = \alpha \delta_j x_i$, calculates its bias correction term (used to update v_{0j} later), $\Delta v_{0j} = \alpha \delta_j$

8	Each output unit (Y_k, $k = 1,.....,m$) updates its bias and weights ($j = 0,.....,p$): $w_{jk}(new) = w_{jk}(old) + \Delta w_{jk}$, Each hidden unit (Z_j, $j = 1,.....,p$) updates its bias and weights ($i = 0,.....,n$): $v_{ij}(new) = v_{ij}(old) + \Delta v_{ij}$
9	Test stopping condition.

2.5. The Epoch and the Gradient Descent

An epoch is a cycle in which the entire set of training patterns is presented to the network. The network using many epochs are required to train the backpropagation algorithm. The number of epochs depends on many factors. The most important factors are the numbers of training data, hidden layers, neurons in hidden layers and dependent output parameters (Fausett, 1994; Rafiq et al., 2001).

The gradient descent method is the optimization technique incorporated in the standard backpropagation. The gradient is the error function and its variables are the weights of the network. Its gradient gives a direction in which error increases. The negative of the gradient shows the direction in which the error decreases (Fausett, 1994).

2.6. The learning rate and the momentum

The learning rate and the momentum are the gradient descent optimization factors. These are introduced to improve the overall efficiency of the standard backpropagation (Kanungo et al., 2006).

The learning rate controls the amount by which the weights may change during training. It can be constant or variable as the training proceeds (Fausett, 1994). It is usually a small number between 0 and 0.5 (Djarfour et al., 2008). The momentum proves advantageous in a situation where some set of training data is different from the majority of the training data or even incorrect (Fausett, 1994).

2.7. Training Modes

There are two kinds of training modes, batch mode and pattern mode. In batch mode, after an epoch is completed, a single average error is calculated over the entire set (batch) of training data. Then the weights are updated according to that average error. It requires less time and is faster to train the network. In pattern mode, error is calculated after each single pattern presentation and the weights are updated, hence taking more computation time.

Backpropagation adjusts the network weights after each patter presentation and therefore it is based on pattern mode algorithm (Mas & Flores, 2008). Selecting the proper mode for training is problem specific. Experience shows that it is advantageous to use batch mode for initial training (Rafiq et al., 2001). Training algorithms such as conjugate gradient descent, quasi-Newton or quick propagation are batch mode algorithms (Mas & Flores, 2008).

2.8. Sum of square error function (SSE)

The SSE function is calculated by squaring the difference between sets of target and actual values, and adding these together. This is the standard error function used in regression problems. It may also be used for classification problems, giving robust performance in estimating discriminant functions. The confidence levels here are not generated as probabilities. Its advantage is that training is often faster and more stable, and the final performance levels are sometimes better.

3. Data preparation

3.1. Study Area

The study investigates GPS navigation data in Islamabad, Pakistan. Fig. (10) shows a vector map of the Islamabad region between latitude 33.7182 N and 33.7179 N, and longitude 73.0713 E and 73.0713 E. An algorithm was developed using C# in Visual Studio.Net 2003 that checks and stores the data. The code is shown in Appendix (1).

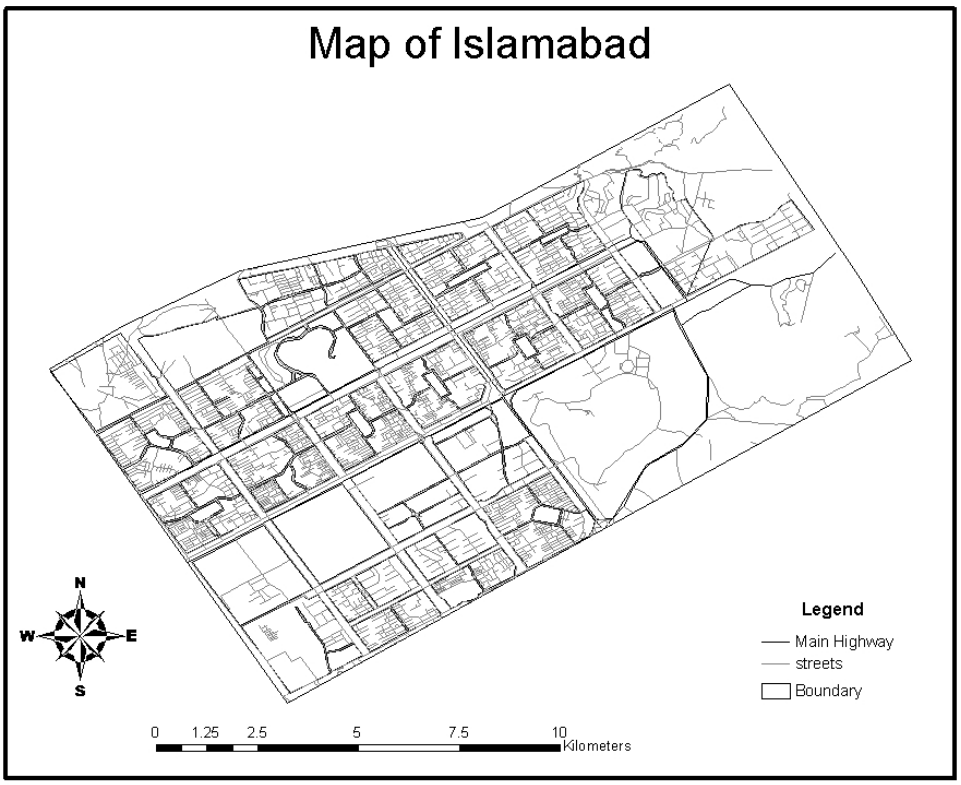

Fig. 10. Map of Islamabad, Pakistan

3.2. Data description

The National Marine Electronic Association (NMEA) has developed specifications for communication between marine equipment such as sonars, anemometers, gyrocompasses, GPS receivers and many other types of marine instruments. GPS receiver communication is defined in this standard. There are standard sentences for each device category. All

standard sentences start with two letter prefix that defines the devices they are using. For GPS receiver the prefix is GP (GPS Information, 2008; NMEA, 2008).

Each sentence begins with a '$' and ends with carriage return/line feed (CR/LF). An example of a type of GPS receiver sentence '*Recommended Minimum*' (RMC) is shown in Table (2). A GP sentence can be longer than 80 characters. The data is contained in a single line where each data item is separated by comma. A checksum is present at the end of some of the sentences followed by asterisks '*'. The checksum is a calculated value used to test the data integrity. It is useful for checking errors that may occur during transmission and reception of a sentence. It is calculated by taking the exclusive OR (XOR) of each character between, but not including, '$' and '*'.

Table (2)
The Global Positioning Recommended Minimum Sentence (GPRMC) format

GPRMC Sentence: $GPRMC,123519,A,4807.038,N,01131.000,E,022.4,084.4,230394,003.1,W,*6A			
Item #	**Data Item**	**Class Type**	**Description**
1	GPRMC	RMC	Recommended Minimum sentence C
2	123519	Time	UTC Time represented as 12:35:19 UTC
3	A	Status	Satellite Fix Status A=active or V=Void
4	4807.038	Latitude	Latitude 48 deg 07.038'
5	N	LatitudeDirection	Latitude direction N = North or S = South
6	01131.000	Longitude	Longitude 11 deg 31.000'
7	E	LongitudeDirection	Logitude direction E = East or W = West
8	022.4	Speed	Speed over the ground (knots)
9	084.4	Bearing	Track angle in degrees True
10	230394	Date	Date 23rd of March 1994
11	003.1	MagneticVariation	Magnetic Variation degrees (Easterly var. subtracts from true course)
12	W	Magnetic Direction	Magnetic direction East/West
13	*6A	CheckSum	The checksum data, always begins with *
14	NULL	True	Output class only used for ANN processing

NMEA consist of sentences where the first three words represent the data type. Each data type represents unique interpretation defined by NMEA standard. Some of the GPS receiver sentences are shown in Table (3). All of these sentences start with prefix GP.

Table (3)
Example of GPS sentences

Data Type	Brief Explanation	Data Type	Brief Explanation
GLL	Lat/Lon data	GRS	GPS Range Residuals
GSA	Overall Satellite data	GST	GPS Pseudorange Noise Statistics
GSV	Detailed Satellite data	RMA	Recommended Loran data
RMB	Recommended navigation data for gps	RMC	Recommended minimum data for gps

3.3. Data coding

The ANN requires data to be presented in a certain format supported by the learning process of the network. Therefore a certain amount of data processing is needed before training patterns are presented to the network. Data preparation and coding are mostly application dependent tasks and there are no universal guidelines available (Rafiq et al, 2001).

Data coding is necessary to convert variables to such a form that the ANN can best utilize them. The GPRMC sentence represents thirteen data items separated by comma (,) and classified into specific class based on its characteristics as shown in Table (2).

The study used supervised training. The detail about supervised training is explained in Section (2.2.1). In Table (2), classes 1 to 13 will be the input vector or independent variables and class 14 is the target vector or dependent variable. The ANN processes only use numeric data in fairly a limited range (Bishop, 1994). Classes 1, 3, 5, 7, 12 and 13 represents data in alpha numeric form. These classes should either be converted to numeric form or should not be used for ANN training. Table (4) represents a detailed description of classes that are only used for ANN utilization.

Table (4)
Variables selection for GPRMC sentence

GPRMC Sentence: $GPRMC,123519,A,4807.038,N,01131.000,E,022.4,084.4,230394,003.1,W*6A				
Item #	**Class Type**	**Actual**	**Converted**	**Remarks**

		Data	**Data**	
1	Time	123519	Null	
2	Latitude	4807.038,N	(+)48.1272	Latitude in numeric form converted to decimal degrees combined with latitude direction. If north then in plus (+) and if South then in minus(-).
3	Longitude	01131.000,E	(+)11.5166	Longitude in numeric form converted to decimal degrees combined with latitude direction. If east then in plus (+) and if west then in minus(-).
4	Speed	022.4	Null	Speed in numeric form measured in knots.
5	Bearing	084.4	Null	Bearing in numeric form between 0 and 360.
6	Date	230394	Null	
7	Magnetic Variation	003.1,W	(-)003.1	Magnetic variation is in numeric form. It is combined with magnetic direction. If east then plus (+) and if west then in minus (-).
8	CheckSum	6A	106 (Decimal value of 6A)	Check sum is in numeric form converted from hexadecimal to decimal form.
9	Output	True or False	Null	Output already in numeric form either true or false.

From Table (2), classes RMC and status are not included, as these are not important classes for the ANN training. Class *status* represents the positioning accuracy based upon the number of satellites in range. The GPS receiver is configured using the specification that if the number of satellites in range are less than 4 the status is invalid (V) and if the number of satellites in range are greater than 4 the status is valid (A). Only data with status (A) is included in the study.

Network complexity arises with the number of input variables, therefore only those input variables are selected that are most influential for the problem on hand (Bishop, 1995). Latitudes, longitudes and magnetic variation are combined with their respective directions in order to minimize the inputs. Latitude and longitude are converted to decimal degrees using algorithm shown in function ReturnDecimalLatitude() in Appendix (2). Check sum is in hexadecimal format. Check sum is converted into decimal

format using the algorithm shown in function CalculateCheckSum() in Appendix (2). An extra output class is added in order to represent ANN dependent variable, it has the value True (valid) and False (invalid). This output class is the target output which represents the integrity of the whole GPRMC sentence i.e. either True or False. An algorithm classifies each GPRMC sentence data item according to its class type and creates data based on Table (4). The source code of the algorithm is available in Appendix (2). The algorithm returns data as shown in Fig. (11).

There are more than eighty (80) commercial ANN packages that are available based on their capabilities (Suryanarayana et al., 2008). Some of the packages with their capabilities are shown in Table (5).

Table (5)
Features of ANN packages (Suryanarayana et al., 2008)

Features of NN packages			
Package	**Type of network**	**Algorithm**	**Salient features**
Direct executing simulation in real time (DESIRE)/NEUNET		Fuzzy logic	Dynamic-system simulation (6000 differential equations, 20,000 difference equations, 13 integration rules) for control, aerospace, chemical engineering, physiological modeling, ecology
Matlab: Neural Network Toolbox	Perceptron RBF Hopfield LVQ Competitive NN Kohonen Elman Hebb	BP Levenberg–Marquard	MATLAB Signal processing Nonlinear control Financial modeling

NeuralWorks	ART 1 Kohonen Modular NN General Regression Fuzzy ART Probabilistic NN LVQ Boltzmann BSB SPR	BP	NeuralWorks Explorer Developers package NeuralWorks Professional
NeuroShell	Ward nets Recurrent Kohonen Probabilistic NN General regression	BP	DuPont in making safety glass Texaco in process control in oil refineries
STATISTICA: Neural Networks	Kohonen PNN GRNN	BP Levenberg–Marquardt conjugate gradient Quick propagation Delta–Bar–Delta Linear SVD *k-Means* *k-Nearest neighbour isotropic deviation*	Genetic input selection, automatic network designer
Trajan	Kohonen	Levenburg–Marquardt Conjugate gradient BP Quick propagation Delta–Bar–Delta SVD *k-Means, k-nearest* Weighed weight regularisation	PCA Neuro-genetic input selection, automatic network design
SNNS	RBF MLP Competitive NN Associative memory	BP Quick propagation Resilient Prop Backpercolation Counter propagation	http://www-ra.informatik.uni-tuebingen.de/snns/

	Jordan Elman Kohonen SOM Elman ART1, ART2 ART MAP Time Delay NN	Dynamic LVQ Dynamic decay adjustment for RBF Simulated annealing Scaled conjugate gradient TACOMA (Task decomposition Correlation Measures and local Attention)	

The empirical study is done using ANN software Trajan 6.0 Demonstrator. The Trajan is a powerful ANN software simulation package. Table (5) shows the Trajan support for a wide range of ANN and the training algorithms. It has implemented many new features, and some of them include graphical tools, statistical tools, pre and post processing tools and labeling (Trajan information, 2009). Data pre and post processing in Trajan supports variety of standard ASCII formats (tab/comma/space separated), STATISTICA and Fast binary data file format (Trajan information, 2009). Due to wide range of support of Trajan for the ANN, it has been used in many researches (Maddena et al., 2000; Hilder et al., 2000; Ková et al., 2002; Spanilá et al., 2004; Hameda et al., 2004, Kellya et al., 2004; Pazourek et al., 2005). Trajan has easy to use Graphical User Interface (GUI). It is also one of the reasons to use Trajan in this study.

Data is prepared according to Trajan specifications. In Trajan, column names cannot exceed more than 8 characters; therefore column names for each class type are reduced to 8 characters as shown in Fig. (11).

	TIME	LATITUDE	LONGITUD	SPEED	BEARING	DATE	MAGVAR	CHECKSUM	OUTPUT	ERRORTYP
1	0	33.72	73.07	34	49	170209	12	5	FALSE	1
2	93404	33.68	73.01	172	247	200209	61	13	TRUE	0
3	190435	0.00	73.05	180	260	170209	65	118	FALSE	2
4	92904	33.68	73.01	172	247	200209	61	119	TRUE	0
5	54949	33.72	0.00	69	99	180209	24	13	FALSE	4
6	92404	33.68	73.01	172	247	200209	61	1	TRUE	0
7	174319	33.72	73.08	0	299	180209	74	1	FALSE	8
8	91904	33.67	73.01	172	247	200209	61	119	TRUE	0
9	33642	33.67	73.08	96	0	190209	34	1	FALSE	16
10	91404	33.68	73.01	172	247	200209	61	1	TRUE	0
11	164816	33.66	73.00	234	338	0	84	6	FALSE	32
12	90904	33.68	73.01	172	247	200209	61	114	TRUE	0
13	42432	33.69	73.01	82	118	200209	0	5	FALSE	64

Fig. 11. Example of GPRMC sentences used for ANN processing

Errors are divided into eight (8) categories based upon the GPRMC sentence data items. See Table (6). Each error type is represented by unique binary value. Error type equal to zero (0) means GPRMC sentence contains no error. From (Fig. 11), column ERRORTYP (error type) shows different kinds of errors present in the GPRMC sentence. This column is not utilized for the ANN processing.

Table (6)
Error types in GPRMC sentences

Error Type	Value
UTC Time Error	1
Latitude Error	2
Longitude Error	4
Speed Error	8
Bearing Error	16
UTC Date Error	32
Magnetic Variation Error	64
CheckSum Error	128

Final data is shuffled using the algorithm presented in Appendix (3). Data shuffling changes the order of presentation of the cases within each epoch. This may add noise to the training process, which means the error may inverse. However, network performance improves. An example of shuffled GPRMC sentences according to ANN and Trajan requirements is shown in Fig. (11).

4. Building ANN

In this study, feed forward multi-layer ANN with one input, one hidden layer and one output layer is used as shown in (Fig. 4). The number of nodes in the input and the output was determined by the structure of GPRMC sentence. The number of neurons in the input layer and the output layer depends upon the number of input and the output variables. Neurons in the hidden layer are determined by trial and error (Kanungo et al., 2006). The difference between the ANN lies in the activation function. This study uses a linear function discussed in Section (2.3.1) at the input layer and a binary sigmoid or logistic function discussed in Section (2.3.3) in the hidden and the output layer.

The samples were divided automatically by Trajan into three subsets in ratio 2:1:1 i.e. training set (50%), selection set (25%) and testing set (25%). The training set is used to train the network, while the selection set is used to check the progress of the network and to define the epoch at which training should be stopped. Training is stopped when the error in the selection set reaches a minimum value. At this point the network has achieved the best generalization. If network training is not stopped, the network will over-train and the performance of the network will deteriorate, despite the error of the training set still decreasing (Luk et al., 2000). The testing set is used at the end to check the selection set is not artificial.

The network is trained with backpropagation algorithm with sum of square error (SSE) as the error function. The details of the algorithms are:

- Threshold: Accept (0.5), Reject (0.5)
- Learning rate : 0.01
- Momentum: 0.3
- Maximum number of epochs: 200

4.1. Designing ANN in Trajan

The ANN models are difficult to optimize and design. There are two ways with which Trajan helps to design the ANN processes.

- Intelligent problem solver (IPS).
- Custom network designer (CND).

IPS is an intelligent tool that helps to create and test ANN for data analysis and prediction problems (Trajan, 1998). It takes as an input the type of problem, the variables to use, the amount of time to be spent designing the network, and the number of networks to be retained (Trajan, 1998). It automatically designs a number of networks to solve the problem. It is a useful tool for novices in the ANN.
On the other hand CND is used to create individual custom neural networks. It gives much more independence to specify the network type, basic architecture and the inputs and the outputs variables (Trajan, 1998). It is rather complex and needs deeper understanding of the ANN design (Trajan, 1998). This study uses the CND tool to design the ANN.

Fig. (12) shows the flow diagram of the different steps involved in the design of ANN for processing the GPRMC sentence. Initially, the GPRMC sentences are imported to Trajan. See Section (4.1.1) for more details. Trajan automatically divides the GPRMC sentences into three subsets i.e. training, selection and testing set. See Section (4.1.3) for more details. Training of the ANN is performed using the training and the selection set. See Section (4.1.4) for more details. If the ANN trained network shows a good performance, then it is tested using the testing set. If the ANN tested network shows a good result, then the network is identified as suitable for GPRMC classification. See Section (4.1.5) for more details. The ANN training process works in a cycle. If the ANN fails to produce better results during training or testing, then the ANN training process is repeated again.

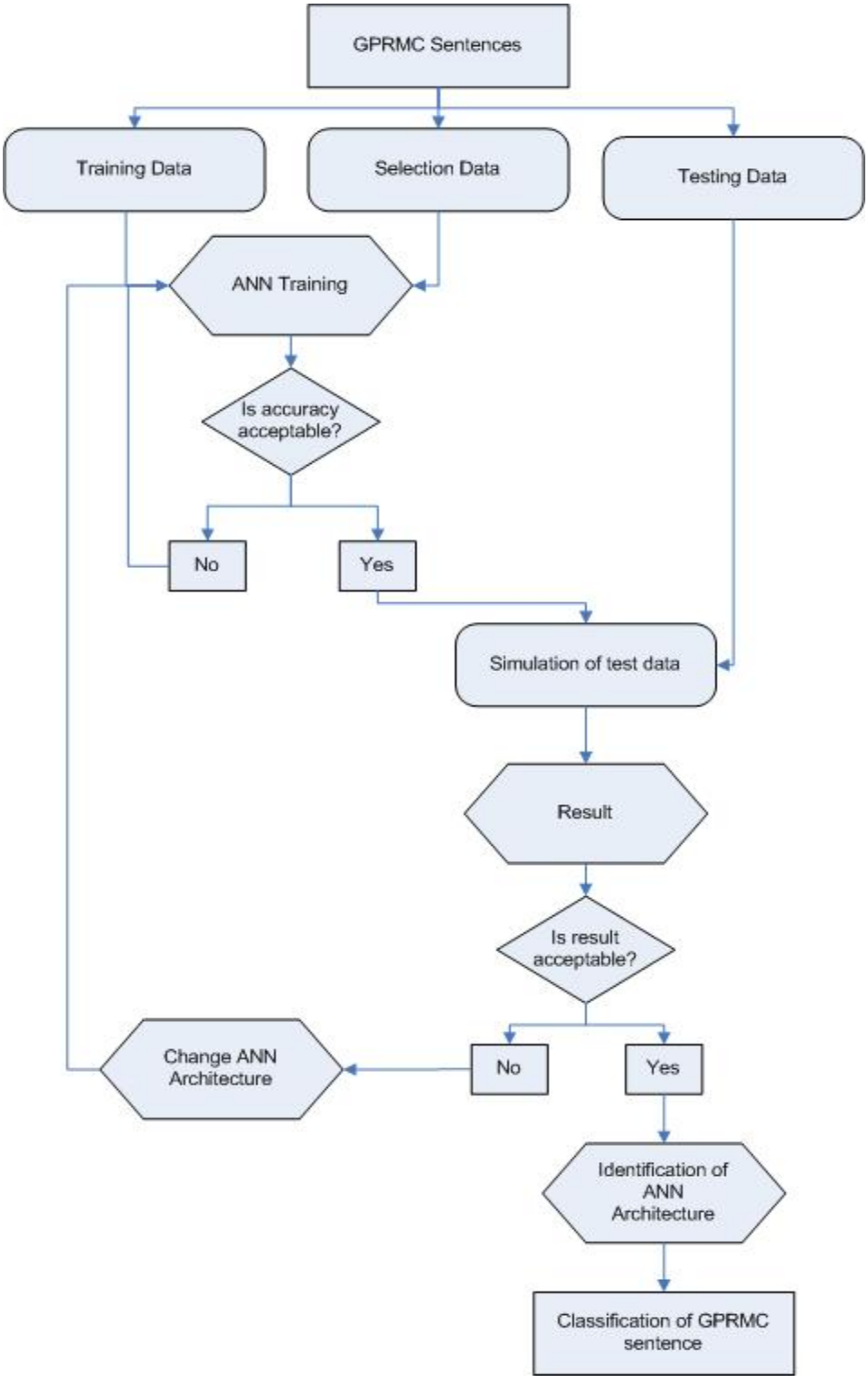

Fig. 12. Flow diagram showing different steps of ANN black box procedure for the classification of GPRMC Sentence

4.1.1. Importing training data sets

Data preparation is the important step for ANN utilization. Detailed description about ANN data sets and preparation of data is explained in Section (3). Data is created in the form of Comma Separated Values (CSV) according to Trajan data import criteria. Data is imported in the Trajan as shown in Fig. (11).

4.1.2. Input and output variable selection

The GPRMC sentence, after pre processing consists of total 8 input (independent) variables and 1 output (dependent) variable as shown in Fig. (13) and (14).

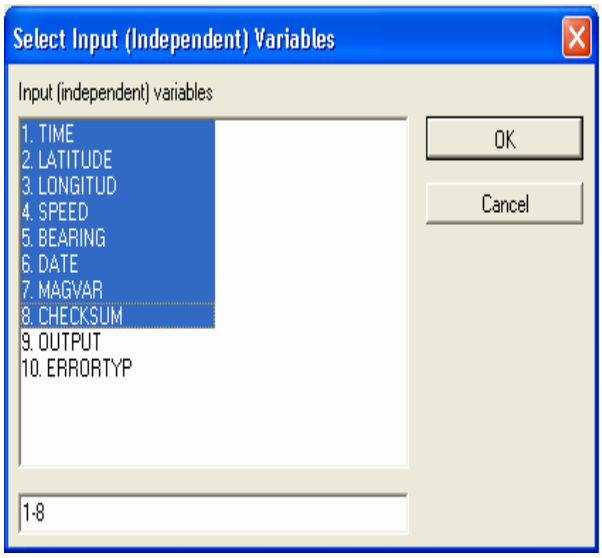

Fig. 13. Selection of input (independent) variable(s)

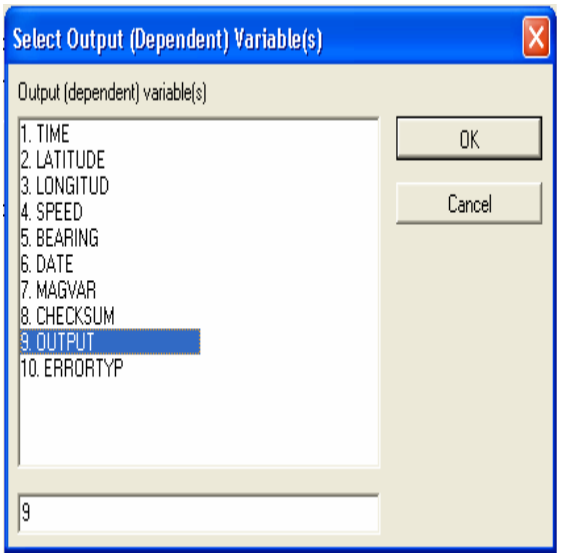

Fig. 14. Selection of output (dependent) variable(s)

4.1.3. Data sampling

A total of 1248 GPRMC sentences were used and divided into three subsets in ratio 2:1:1 i.e. training set (624), selection set (312) and test set (312). See Fig. (15).

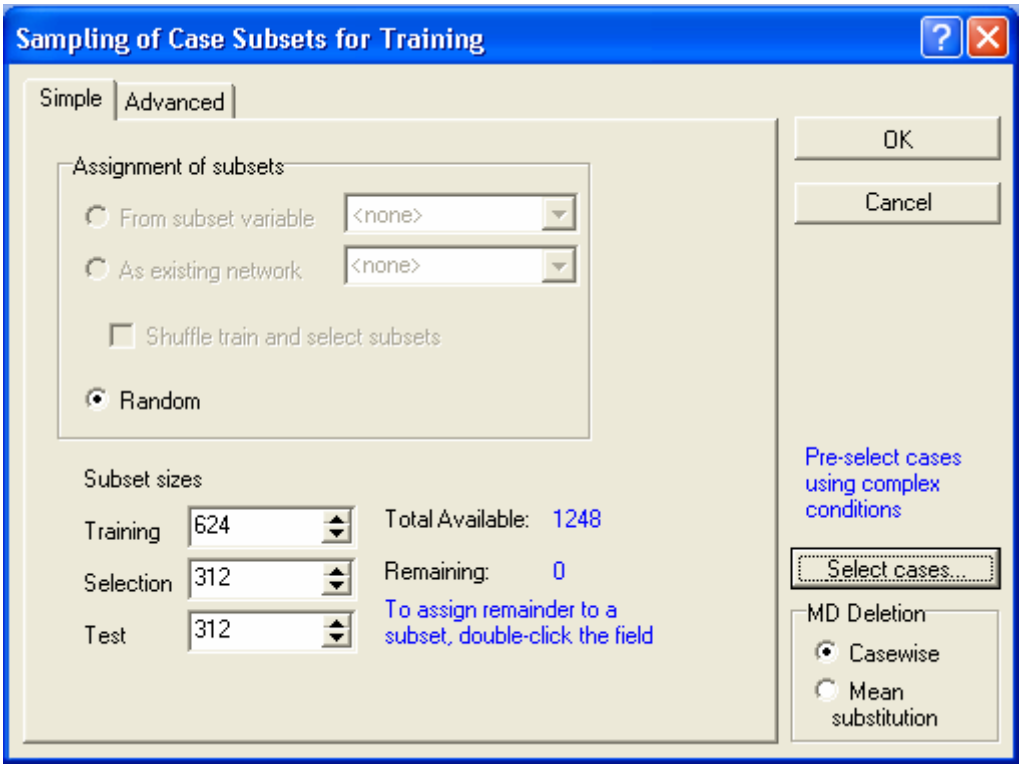

Fig. 15. Data sampling for ANN

4.1.4. Training

Fig. (16) shows the progress of the ANN training with a graph plotting network performance by monitoring the sum of square error (SSE) on each epoch for selection and training data. Progress bar will stop when maximum epochs reached to 200 and ends with total estimated time.

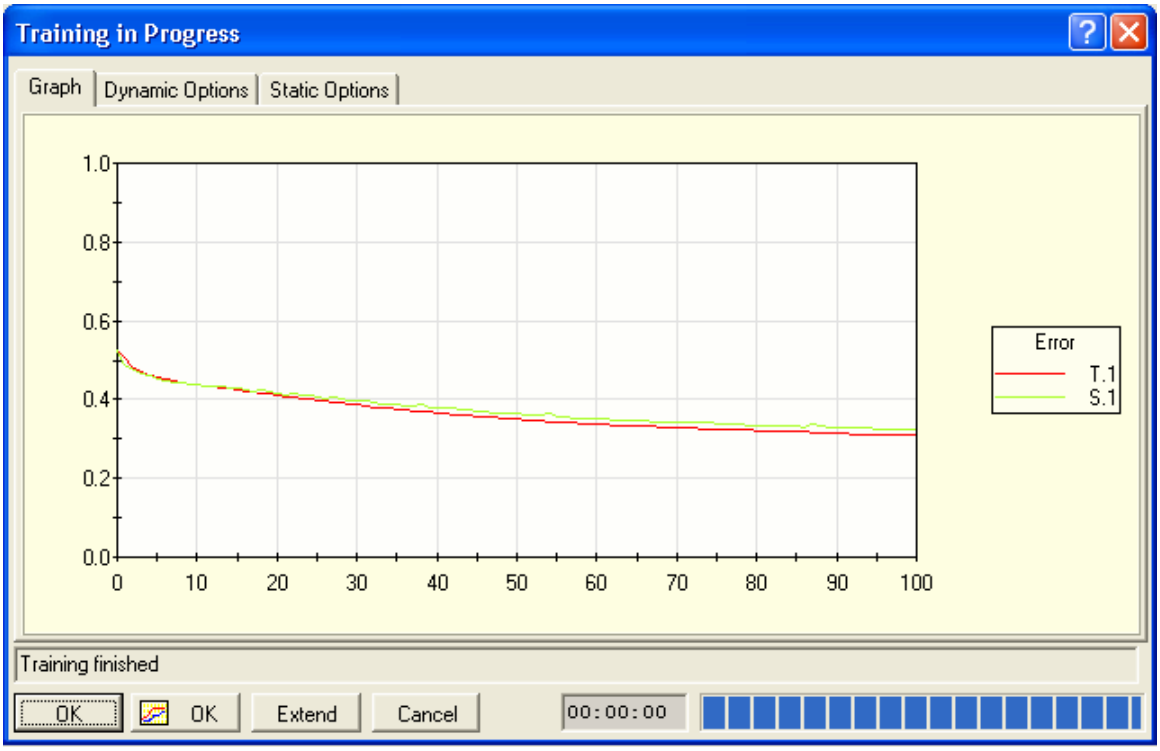

Fig. 16. ANN training progress

4.1.5. Results

Fig. (17) shows the result dialog of ANN with backpropagation algorithm. Index is a unique identifier assigned when the model is created and preserved throughout its lifetime. Profile shows a summary of the model's structure in the form I:N-N-N:O, where I is the number of input variable, O the number of output variables, and N the number of units in each layer. Perf (Train, Select, Test) shows the performance (correct classification) of a network on the subsets used during training. Error (Train, Select, Test) shows the error (SSE) of the network on the subsets used during training. Training contains a concise description of the training algorithms used to optimize the network. The code BP98b indicates that the Back Propagation algorithm is used, the network is found on the 98th epoch and “b” represents the best network with the lowest selection error.

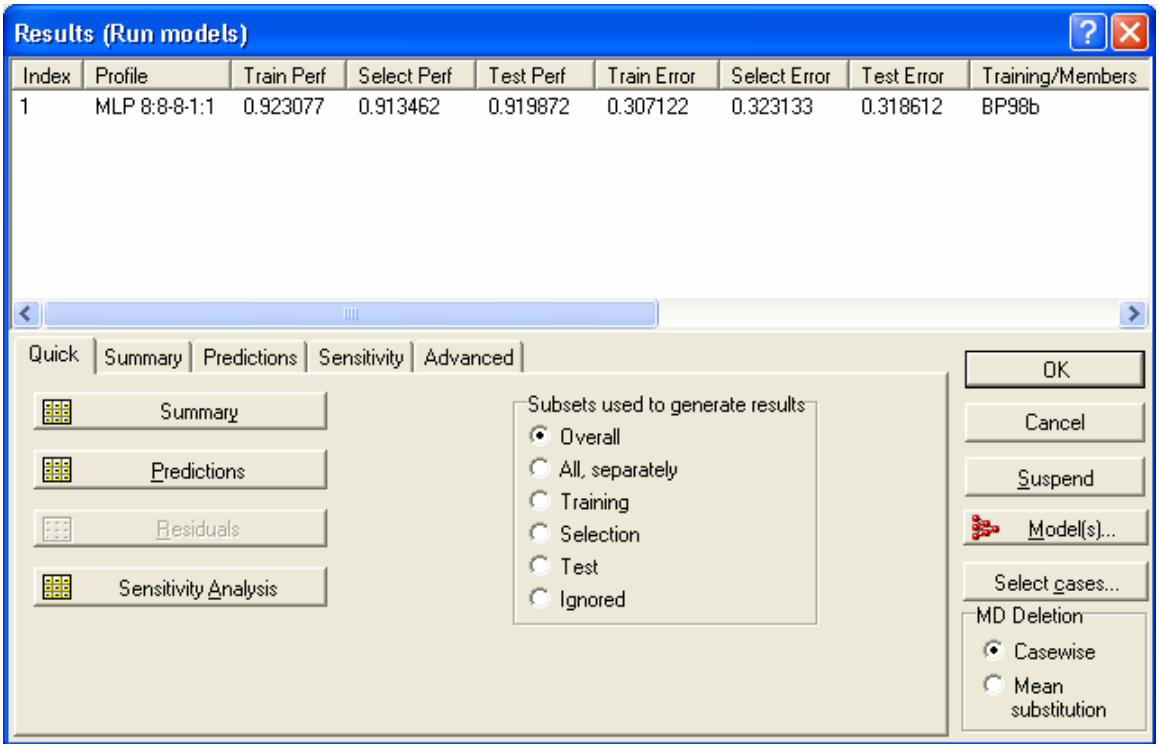

Fig. 17. Results of ANN training

5. Results

The performance of the ANN is evaluated by the classification accuracy and the sum of square error of the selection data. In total 40 ANN architectures were created by varying the number of neurons in hidden layers. Table (7) shows the results from 15 different ANN architectures. The best obtained ANN architecture was 8:8-12-1:1, showing an input layer having 8 neurons, preprocessed into 8 neurons, a hidden layer with 12 neurons and finally output layer with 1 neuron. The ANN architecture of 8:8-12-1:1 network is shown in (Fig. 18). The network shows good performance with the selection classification accuracy of 96.79 % and the selection sum of square error 0.2022.

Table (7)
Training, testing and selection data accuracies of ANN black box procedure (bold indicates the best acceptable architecture in this study)

Test #	ANN Architecture	Classification %			Sum of Square Error		
		Training	Testing	Selection	Training	Testing	Selection
1	8:8-8-1:1	94.23	92.62	90.38	0.2402	0.2694	0.3029
2	8:8-9-1:1	93.26	92.30	92.94	0.2594	0.2638	0.2498
3	8:8-10-1:1	93.91	92.94	95.19	0.2565	0.2802	0.2347
4	8:8-11-1:1	93.58	90.70	91.02	0.2584	0.2943	0.3069
5	**8:8-12-1:1**	**95.83**	**94.87**	**96.79**	**0.2084**	**0.2362**	**0.2022**
6	8:8-13-1:1	93.91	91.02	93.58	0.2335	0.2689	0.2214
7	8:8-14-1:1	95.19	96.15	95.83	0.2083	0.2046	0.2145
8	8:8-15-1:1	93.42	93.26	93.26	0.2518	0.2620	0.2593
9	8:8-16-1:1	97.91	96.79	96.79	0.1931	0.2195	0.2299
10	8:8-17-1:1	95.67	93.58	93.26	0.2108	0.2506	0.2626
11	8:8-18-1:1	91.98	92.62	92.62	0.2798	0.2543	0.2578
12	8:8-19-1:1	95.03	91.66	92.94	0.2149	0.2682	0.2569
13	8:8-20-1:1	96.63	96.47	95.19	0.1991	0.2230	0.2291
14	8:8-21-1:1	92.14	94.23	92.94	0.2534	0.2273	0.2589
15	8:8-22-1:1	94.23	92.62	90.38	0.2412	0.2621	0.2917

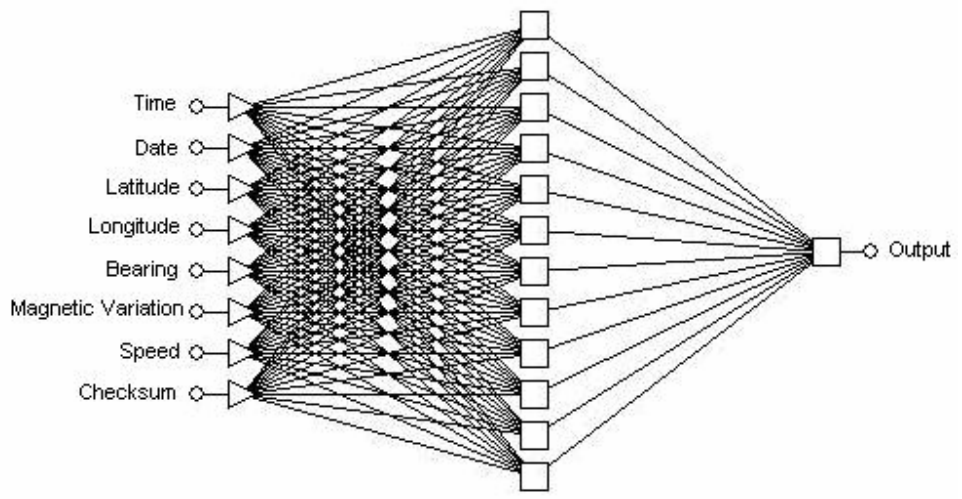

Fig. 18. The ANN architecture of 8:8-12-1:1 network.

5.1. Classification accuracy

Table (8) shows the overall classification accuracy of the network 8:8-12-1:1. There are a total of 1248 cases presented to the network with 624 cases true and false each as shown in Table (8). The network 8:8-12-1:1 has correctly classified 95.513% of the true output and 96.154% of the false output cases; and it has incorrectly classified 4.487% of the true cases and 3.846% of false cases. Unknown means the network was unable to reach any answer. The network 8:8-12-1:1 reported 0% of unknown cases for both true and false output.

Table (8)
Classification accuracy of 8:8-12-1:1 network

	Output - True	**Output - False**
Total	624	624
Correct	596	600
Wrong	28	24
Unknown	0	0
Correct (%)	95.513	96.154
Wrong (%)	4.487	3.846
Unknown (%)	0	0

5.2. Sensitivity analysis

Sensitivity analysis measures the importance of one or more variables based on the ANN model. The data set is submitted to the network repeatedly, with each variable in turn

treated as missing, and the resulting network error is recorded (Hunter et al., 2000). If the error increases it means the variable is important for the ANN and if it decreases then it means the input is unimportant for the ANN.

Trajan has defined a missing value substitution procedure for every model. The substitution procedure allows predictions to be made in the absence of values for one or more inputs. To define the sensitivity of a particular variable, first run the network on a set of test cases, and accumulate the network error. Then run the network again using the same cases, but this time replacing the observed values with the value estimated by the missing value procedure, and again accumulate the network error (Trajan, 1998).

The basic measure of sensitivity is the ratio of the error with missing value substitution to the original error. The more sensitive the network is to a particular input, the greater the deterioration is expected, and therefore the greater the ratio. If the ratio is one or lower, then making the variable "unavailable" either has no effect on the performance of the network, or actually enhances it (Trajan, 1998). Table (9) shows sensitivity analysis of the 8:8-12-1:1 network, ranked in order. It can be seen that all the input variables have passed the sensitivity analysis with a ratio greater than 1.

Table (9)
Sensitivity analysis of 8:8-12-1:1 network

Ranking	Variable	Ratio
1	Checksum	2.56548
2	Speed	1.96498
3	Time	1.87891
4	Date	1.48226
5	Latitude	1.47060
6	Magnitude variation	1.31819
7	Bearing	1.25569
8	Longitude	1.21437

5.3. Training graph

The training graph demonstrates the concept of the early stopping technique that provides an efficient way to constrain the training of an ANN. This technique monitors the progress of the network training using the selection set. Fig. (19) shows a graphical plot

of the 8:8-12-1:1 network between SSE (Y-axis) and the number of epochs (X-axis) using the training set and the selection set. Training is stopped when the error in the selection set reaches a minimum value. At this point the network has achieved the best generalization. If training is not stopped, the network will be over-trained and the performance of the network will deteriorate despite the error of the training data still decreasing.

The network 8:8-12-1:1 has been tested using different maximum number of epochs like 100, 200, 300, 500 and 1000. The network 8:8-12-1:1 showed good generalization and obtained minimum SSE with the maximum number of epochs 200. See Fig. (19).

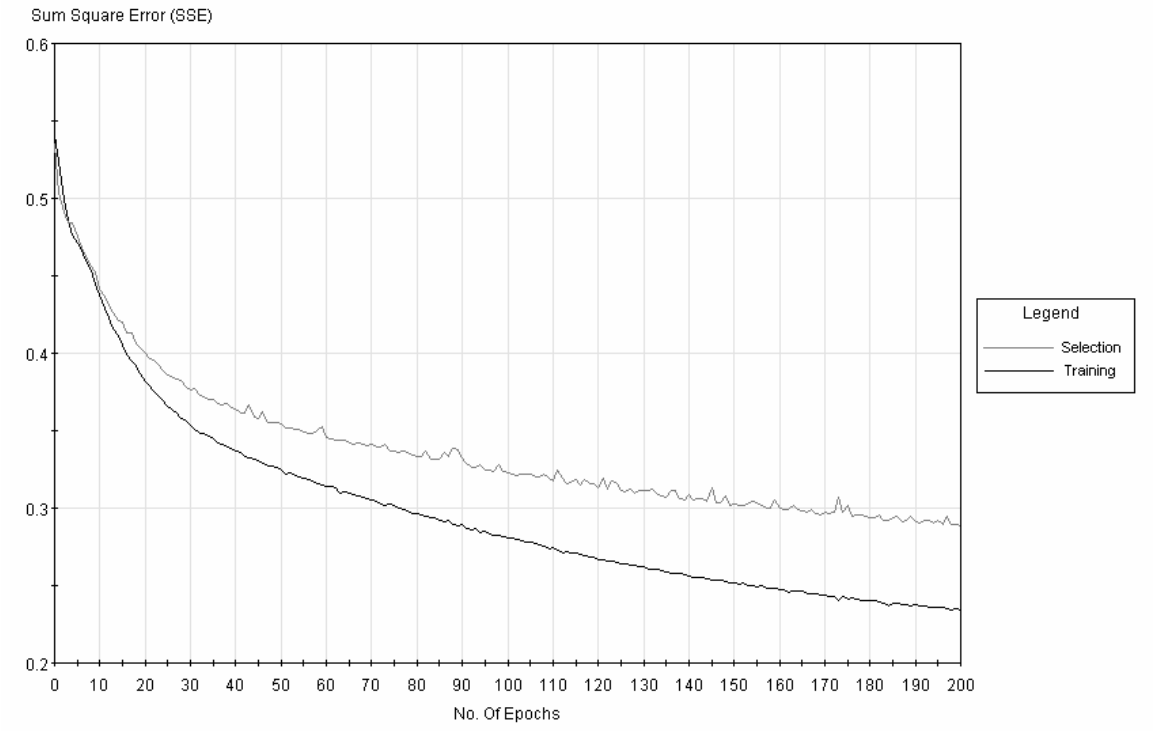

Fig. 19. Training graph of 8:8-12-1:1 network

6. Discussion

This study checks the integrity of GPRMC sentences using the ANN. The ANN modeling is performed using GPRMC sentences. A total of 1248 GPRMC sentences were prepared for the ANN utilization. The GPRMC sentences were divided into three subsets in ratio 2:1:1 i.e. training set (624), selection set (312) and test set (312). In total 40 ANN architectures were created and the best ANN architecture is selected based on the classification and the SSE.

Table (7) shows the result from 15 different ANN architectures based on the classification and the SSE. Both the classification and the SSE are used to show the network overall performance. A good network has high classification percentage with a least number of SSE. In table (7), classification and the SSE are represented on the three subsets i.e. training set, selection set and test set. The performance of the network is evaluated using only the selection set. Training and the testing cannot be used for checking network performance. As in the case of training set, there are chances that may be one of the network may have over-learned and have good training performance. While the testing set is used at the end of design process to check that the selection set is not artificial.

The best obtained 8:8-12-1:1 ANN architecture shows a good performance in comparison to other ANN architectures. It shows the selection classification 96.79 % and the selection SSE 0.2022. The results show that the ANN classification technique has achieved a good generalization with a least number of SSE. Table (8) shows in detail the overall classification accuracy of network 8:8-12-1:1.

The ANN is famous for solving non linear problems. Several studies have been made to measure the accuracy of ANN classification technique for solving non linear problems like Pasquier and Hamodrakas (1999) achieved the classification accuracy of 100% by classifying transmembrane proteins using the hierarchical ANN, Zhichen, et al. (2009) achieved the ANN classification accuracy of 78% by classifying the weed species based

on color leaf texture feature , Wei, et al. (2009) performed the comparison study using two classifiers, the Maximum Likelihood Classifier (MLC) and the ANN to identify selected internal wood, and found that the ANN produced high classification accuracy (97.6%) compared with the MLC (80.9%). Our study is also an attempt to solve the non linear problem using the ANN classification technique. The result from our study shows that the ANN can be used for checking the integrity of GPRMC sentences and has successfully achieved the classification of 96.79%.

Backpropagation is the best known training algorithm for ANN. It is easy to understand and the most commonly used for ANN modeling. However, there are some numerous drawbacks for using backpropagation. It is very slow to properly converge on an error minimum, it can be trapped in a local minimum and it is difficult to find appropriate parameters for network training (Mas & Flores, 2008).

It is difficult to find appropriate ANN models. The difficulty lies in the determination of appropriate characteristics for training data, architecture of the network and avoiding over learning. There are no standard rules for the design of ANN models and majority of researchers use trial and error approaches to find best network

Parameter selection is the most important part of ANN training and learning. There are no standard guidelines for parameter selection (Mas & Flores, 2008). The GPRMC sentence is composed of thirteen data items separated by comma (,). Each data item represent unique information. It is difficult to neglect any data item for ANN modeling. Therefore, different data items are combined in a way to have minimum parameters for ANN processing. Numerous ANN architectures are tested to find the best network with minimum parameters.

This study performs ANN training of GPRMC sentences between latitude 33.7182 N and 33.7179 N, and longitude 73.0713 E and 73.0713 E. If the GPRMC sentences are provided outside extent, then it may not provide acceptable data classification. One of the advantages of the ANN network is that it is constantly learning and, with the passage of

time, it may produce an acceptable classification of patterns outside the latitude and longitude extents as mentioned above.

In training samples, null values are replaced by zero as there was no alternative way of assigning null values to GPRMC sentences data item. If the data items are left blank then the Trajan automatically assigns values using methods like mean substitution or various types of interpolation and extrapolation. Therefore, all the data items have a lower bound starting from zero. Despite the null value problem, the study produces better generalization.

7. Conclusions

An ANN is a powerful tool for learning and generalizing. This study used ANN classification technique to check the integrity of GPRMC sentences. The GPRMC sentences are erroneous and contain errors like missing values, incorrect position, incorrect directions and incomplete sentences. The ANN is trained on all possible errors in the GPRMC sentences to achieve better classification. Different ANN architectures are designed and tested for GPRMC sentences classification. The best obtained 8:8-12-1:1 ANN architecture shows a good performance with the selection classification of 96.79 % and the selection sum of square error 0.2022. Hence, the result shows that the ANN classification technique can be used successfully for checking the integrity of GPRMC sentences.

The sensitivity analysis shows the importance of input variables for ANN modeling. The results from sensitivity analysis show that the selected eight (8) input variables are required for the ANN modeling and will not affect the network performance for the GPRMC sentences classification. The training graph is used to check the over learning of the ANN. The results from the training graph show that the ANN did not over learn.

8. Future work

This study produced results based on ANN Backpropagation learning with a learning rate and a momentum as its parameters. Further research can be done using alternate learning algorithms like conjugate gradient, quasi Newton and Levenberg-Marquardt.

Research used only one GPS sentence RMC for checking data integrity. The ANN can be designed to check data integrity of multiple GPS sentences like GLL (latitude and longitude data), GRS (range residuals), GSA (overall satellite data), GST (pseudorange noise statistics), GSV (detailed satellite data) etc.

The GPRMC training samples used in this research contain only one type of error in each sentence. Training can be made more complex by adding multiple errors in each sentence like time, date and speed errors etc. This will ultimately improve the performance of the ANN for data classification, but at the same time more training samples are needed for better classification.

References

Baum, E.B., & Haussler, D. (1989). What size net gives valid generalization? Neural computation. (Vol 1), 151-160

Besaw, L.E., Rizzo, D.M., Kline, M., Underwood, K.L., Doris, J.J., Morrissey, L.A., Pelletier, K. (2009). Stream classification using hierarchical artificial neural networks: A fluvial hazard management tool. Journal of Hydrology 373 (2009) 34–43

Bishop, C.M. (1994). Neural network for pattern recognition, Oxford, UK: Oxford University Press

Chiu, C.C., Liau, B.Y., Yeh, S.J., Hsu, C.L. (2008). Artificial Neural Network Classification of arterial pulse waveforms in cardiovascular diseases. Biomed 2008, Proceeding 21, pages: 129-132

Dey, R., Bajpai, V., Gandhi, G., Dey, B., (2008). Application of Artificial Neural Network (ANN) technique for diagnosing diabetes mellitus. IEEE Region 10 Colloquium and third internation conference on industrial and information systems. Vol 1 & 2, Pages 197-200

Djarfour, N., Aïfa, T., Baddari K., Mihoubi, A., Ferahtia, J. (2008). Application of feedback connection artificial neural network to seismic data filtering, Internal Geophysics 2008.

Erol, R., Oğulata, S.N., Alparslan Z.N. (2008). A radial basis function neural network (RBFNN) approach for structural classification of thyroid diseases. J Med Syst (2008) 32:215–220

Fausett, L. (1994). Fundamentals of neural network architecture, algorithms, and applications, UK: Prentice-Hall

García, N., Gámez, M., Alfaro, E. (2008). *ANN + GIS: An automated system for property valuation*, Neurocomputing, 71, (pp. 733-742)

GPS information. (2008). [On-line]. Retrieved October 27, 2008 13:00 at URL: http://www.gpsinformation.org/dale/nmea.htm.

Hameda, A.B., Elosta, S., Havel, J. (2004). Optimization of the capillary zone electrophoresis method for Huperzine A determination using experimental design and artificial neural networks. Journal of Chromatography A, Volume 1084, Issues 1-2, 19 August 2005, Pages 7-12

Haykin, S. (1994). Neural Networks A Comprehensive Foundatio, Macmillan College Publishing Company, USA

Heaviside Step Function (2009). [On-line]. Retrieved August 16, 2009 10:00 at URL: http://en.wikipedia.org/wiki/Heaviside_step_function

Hilder, E.F., Klampfl, C. W., Haddad, P. R. (2000). Pressurized-flow anion-exchange capillary electrochromatography using a polymeric ion-exchange stationary phase. Journal of Chromatography A, Volume 890, Issue 2, 25 August 2000, Pages 337-345

Hunter, A., Kennedy, L., Henry, J. and Ferguson, R.I. (2000) . Application of Neural Networks and Sensitivity Analysis to improved prediction of Trauma Survival, Computer Methods and Algorithms in Biomedicine 62, pp. 11-19.

Jenkins, W.M. (1997). An introduction to neural computing for the structural engineering. The Struct Engng, 75, (pp. 38-41)

Kanungo, D.P., Sarkar, S., Arora, M.K., Gupta, R.P. (2006). A comparative study of conventional, ANN black box, fuzzy and combined neural and fuzzy weighting

procedures for landslides susceptibility zonation in Darjeeling Himalayas, Engineering Geology, 85, (pp. 347- 366)

Kantardzic, M., Djulbegovic, B., Hamdan, H. (2002). A data-mining approach to improving Polycythemia Vera diagnosis. Computers and industrial engineering Volume (43), Issue (4), Pages: 765-773, Published: SEP 2002

Kellya, T., Doble, P., Dawson, M. (2004). Chiral analysis of methadone and its major metabolites (EDDP and EMDP) by liquid chromatography–mass spectrometry. Journal of Chromatography B, Volume 814, Issue 2, 25 January 2005, Pages 315-323

Ková, P.P., Bocaz, G., Li, H., Havel, J. (2002). Evaluation of calibration data in capillary electrophoresis using artificial neural networks to increase precision of analysis. Journal of Chromatography A, Volume 979, Issues 1-2, 6 December 2002, Pages 59-67

Lee, S., Ryu, J.H., Won, J.S., Park, H.J. (2004). Determination and application of the weights for landslide susceptibility mapping using an artificial neural network, Engineering Geology, 71, (pp. 289-302)

Luk, K.C., Ball, J.E., Sharma, A. (2000). *A study of optimal lag and spatial inputs to artificial neural network for rainfall forecasting*, Journal of Hydrology, 227, (pp. 56-65)

Maddena, J.E., Avdalovicb, N., Haddad, P.R., Havel, J. (2000). Prediction of retention times for anions in linear gradient elution ion chromatography with hydroxide eluents using artificial neural networks. Journal of Chromatography A, Volume 910, Issue 1, 23 February 2001, Pages 173-179

Marsh, I., & Brown, C. (2008). Neural network classfication of multibeam backscatter and bathymetry data from stanton bank. Applied Acoustics 70 (2009) 1269–1276

Mas, J.F and Flores, J.J. (2008). *The application of artificial neural networks to the analysis of remotely sensed data*, International journal of Remote Sensing, 29(3), (pp. 617-663)

Mudroch, M., Pechac, P., Grábner, M., Kvicera, V. (2009). Classification and Prediction of Lower Troposphere Layers Influence on RF Propagation Using Artificial Neural Networks. ICONIP 2008, Part I, LNCS 5506, pp. 893–900, 2009.

Müller, R. D. & Eagles, S. (2007). Mapping Seabed Geology by Ground-Truthed Textural Image/Neural Network Classification of Acoustic Backscatter Mosaics. Math Geol (2007) 39: 575–592

NMEA. (2008). Retrieved October 27, 2008 13:00 at URL http://en.wikipedia.org/wiki/NMEA

Pavel, M., Fannin. R.J., Nelson, J.D. (2007). *Replication of a terrain stability mapping using an artificial neural network*, Geomorphology

Palade, V., Howlett, R.J., Jain, L.C. (2003). A Study of Aquatic Toxicity Using Artificial Neural Networks. KES 2003, LNAI 2774, pp. 911-918, 2003.

Pasquier,C., Hamodrakas, S.J. (1999). An hierarchical artificial neural network system for the classification of transmembrane proteins. Protein Engineering vol.12 no.8 pp.631-634

Pazourek, J., Gajdošová, D., Spanilá, M., Farková, M., Novotná, K., Havel, J. (2005). Analysis of polyphenols in wines: Correlation between total polyphenolic content and antioxidant potential from photometric measurements: Prediction of cultivars and vintage from capillary zone electrophoresis fingerprints using artificial neural network. Journal of Chromatography A, Volume 1081, Issue 1, 15 July 2005, Pages 48-54

Rafiq, M.Y., Bugmann, G., Easterbrook, D.J. (2001), Neural network design for engineering applications, Computers and Structures, 79, (pp.1541-1552)

Spanilá, M., Pazourek,J., Farková, M., Havel, J. (2004). Optimization of solid-phase extraction using artificial neural networks in combination with experimental design for determination of resveratrol by capillary zone electrophoresis in wines. Journal of Chromatography A, Volume 1084, Issues 1-2, 19 August 2005, Pages 180-185

Trajan information. (2009). [On-line]. Retrieved August 18, 2009 13:00 at URL: http://www.cis.hut.fi/projects/somtoolbox/links/trajan.shtml

Trajan Demonstrator (Version 6.0) [Artificial Neural Network Software]. (1998). The Old Rectory, Low Toynton, Horncastle, Lincs., LN9 6JU, UK.

Wei, Q., Chui, Y.H., Leblon, B., Zhang, S.Y. (2009). Identification of selected internal wood characteristics in computed tomography images of black spruce: a comparison study.Journal of wood science. Vol(55:3) Pages: 175-180

Zhichen, L., Qiu, A., Changying, J. (2009). Classification of weed species using artificial neural networks based on color leaf texture feature. Computer and Computer Technologies In Agriculture II, Vol (2) Pages: 1217-1225

Appendix 1

```
/// <summary>
/// This algorithm is developed using Microsoft Visual Studio.Net 2003
/// using C# language.
/// The function CheckDataExtents() will check whether the provided
/// data latitude and longitude exists between the specified overall
/// data extents. If the data doesn't exists between the specified
/// coordinates then the function  will return output false otherwise
/// it will return true
/// </summary>
/// <param name="dataLatitude">Latitude of the data under investigation
</param>
/// <param name="dataLongitude">Longitude of the data under
investigation
</param>
/// <param name="mapTopLatitude">Top latitude of the overall data
</param>
/// <param name="mapBottomLatitude">Bottom latitude of the overall data
</param>
/// <param name="mapLeftLongitude">Left longitude of the overall data
</param>
/// <param name="mapRightLongitude">Right longitude of the overall data
</param>
/// <returns>Returns true if data is within map specified extents
otherwise false</returns>

public bool CheckDataExtents(double dataLatitude, double dataLongitude,
            double mapTopLatitude, double mapBottomLatitude,
            double mapLeftLongitude, double mapRightLongitude)
{
if( (dataLatitude >= mapTopLatitude &&
      dataLatitude <= mapBottomLatitude) &&
      (dataLatitude >= mapLeftLongitude
      && dataLatitude <= mapRightLongitude))
      return true;
else
      return false;
}
```

Appendix 2

```
// GPRMC Errors Types enumeration
public enum RMCErrorType
{
      RMCNull = 0,
      RMCTime = 1,
      RMCLatitude = 2,
      RMCLongitude = 4,
      RMCSpeed = 8 ,
      RMCBearing = 16,
      RMCDate = 32 ,
      RMCMagneticVariation = 64,
      RMCCheckSum = 128
};

/// <summary>
/// This function will process GPRMC sentence and
/// convert accroding to NN and Trajan specification
/// </summary>
/// <param name="sGPRMC">GPRMC sentence</param>
/// <returns>Returns GPRMC sentence in NN form</returns>
public string ProcessGPRMCSentenceForANN(string sGPRMC)
{

// ANN trajan format for GPRMC sentence
// Time,Latitude,Longitude,Speed,Bearing,Date,
// MagneticVariation,CheckSum,Ouput,ErrorType

// GPRMC Sentence Formation when splitted with criteria comma (,)
// [0]=GPRMC,[1]=Time,[2]=Status,[3]=Latitude,[4]=LatDirection,
// [5]=Longitude,[6]=LongDirection,[7]=Speed,[8]=Bearing,[9]=Date,
// [10]=MagneticVariation,[11]=MagneticDirection,[12]=CheckSum

// Split GPRMC sentence
string[] saSentence = sGPRMC.Split(",".ToCharArray);

// Create GPRMC sentence in ANN format
string sGPRMCANN = string.Format("{0},{1},{2},{3},{4},{5},{6},{7},{8}",
saSentence[1],
ReturnDecimalLatitude(saSentence[3],saSentence[4]).ToString(),
ReturnDecimalLongitude(saSentence[5],saSentence[6]).ToString(),
saSentence[7],
saSentence[8],
saSentence[9],
ReturnMagneticVariation(saSentence[10],saSentence[11]).ToString(),
GetGPRMCSentenceOuputAndError(sGPRMC));

// Return GPRMC sentence in ANN format
return sGPRMCANN;
}
```

```
/// <summary>
/// Function checks all the GPRMCS sentence and outputs
/// as TRUE or FALSE and with GPRMC error enumeration
/// </summary>
/// <param name="sGPRMC">GPRMC sentence</param>
/// <returns>Returns ouput either TRUE/FALE with error</returns>
public string GetGPRMCSentenceOuputAndError(string sGPRMC)
{
// GPRMC Sentence Formation when splitted with criteria comma (,)
// [0]=GPRMC,[1]=Time,[2]=Status,[3]=Latitude,[4]=LatDirection,
// [5]=Longitude,[6]=LongDirection,[7]=Speed,[8]=Bearing,[9]=Date,
// [10]=MagneticVariation,[11]=MagneticDirection,[12]=CheckSum

// Split GPRMC sentence
string[] saSentence = sGPRMC.Split(",".ToCharArray());

// Check Time
if (saSentence[1] == "")
return "FALSE,1";

// Check Latitude
if (saSentence[3] == "")
return "FALSE,2";

// Check Latitude direction
if (saSentence[4] == "")
return "FALSE,2";

// Check Longitude
if (saSentence[5] == "")
return "FALSE,4";

// Check Longitude direction
if (saSentence[6] == "")
return "FALSE,4";

// Check Speed
if (saSentence[7] == "")
return "FALSE,8";

// Check Bearing
if (saSentence[8] == "")
return "FALSE,16";

// Check Date
if (saSentence[9] == "")
return "FALSE,32";

// Check Magnetic variation
if (saSentence[10] == "")
return "FALSE,64";

// Check Magnetic direction
if (saSentence[11] == "")
return "FALSE,64";
```

```
// Check the checksum
if (saSentence[12] == "")
return "FALSE,128";

return "TRUE,0";

}

/// <summary>
/// Function checks magnetic variation and its direction and return
/// magnetic variation in (+/- Magnitude Variation)
/// </summary>
/// <param name="magnetic">Magnetic Variation</param>
/// <returns>returns (+/-) Magnitude Variation</returns>
public double ReturnMagneticVariation(string magVar, string magDir)
{
// Try block to catch any irrelevant error
try
{
// Get magnetic variation
string sMagVar = magVar;
string sMagDir = magDir;

// if empty
if(magVar == string.Empty || magDir == string.Empty)return 0;

// Check length
if(magVar.Length <=0) return 0;
if(magDir.Length > 1) return 0;

double dMagVariation = double.Parse(magVar) * ((magDir == "W")?-1:1);
return dMagVariation;

}

// Catch block works with Try block. If any error arrives, it catches
and Return error response according to user specification
catch{return 0;}

}

/// <summary>
/// Function calculate checksum of GPRMC sentence and returns error in
decimal
/// </summary>
/// <param name="sentence">Accept sentence between string $ and
*</param>
/// <returns>Returns decimal heck sum in two character format</returns>
public string CalculateCheckSum(string sentence)
{
// Try block to catch any irrelevant error
try
{
// Loop through all chars to get a checksum
int Checksum=0;

foreach(char Character in sentence)
```

```
{
switch(Character)
{
case '$':
// Ignore the dollar sign
break;
case '*':
// Stop processing before the asterisk
continue;
default:
// Is this the first value for the checksum?
if (Checksum == 0)
{
// Yes. Set the checksum to the value
Checksum = Convert.ToByte(Character);
}
else
{
// No. XOR the checksum with this character's value
Checksum = Checksum ^ Convert.ToByte(Character);
}
break;
}
}
// Return the checksum formatted as a two-character hexadecimal
return Checksum.ToString("00");
}
// Catch block works with Try block. If any error arrives, it catches
// and Return error response according to user specification
catch{return string.Empty;}
}

/// <summary>
/// Function returns decimal latiude with direction as (+/-)
/// </summary>
/// <param name="lat">Latitude </param>
/// <param name="direction">Direction </param>
/// <returns>returns decimal latiude with direction as (+/-) </returns>
public double ReturnDecimalLatitude (string lat,string direction)
{
// Try block to catch any irrelevant error
try
{
// local variable assignment
string sLatitude = lat;
string sLatDirection = direction;
// if empty
if (lat == string.Empty || direction == string.Empty)return 0;
// Check length
if(lat.Length < 4)return 0;
// Check direction
if(direction.Length  > 1)return 0;
int iDirection = 1;
if(!(direction == "N" || direction == "S"))
return 0;
// Get direction
iDirection = (direction=="S")?-1:1;
```

```
// GRPRMC format for latitude is DDMM.SS
// Where DD represents degress, MM reprsents minutes and SS seconds
//int iDegrees, iMinutes, iSeconds = 0 ;
double dDegrees, dMinutes, dSeconds = 0 ;
double dDecimalDegrees = 0.0;
dDegrees = Convert.ToDouble(lat.Substring(0,2));
dMinutes = Convert.ToDouble(lat.Substring(2,2));
if(lat.IndexOf(".") != -1)
dSeconds = Convert.ToDouble(lat.Split(".".ToCharArray())[1]);
dDecimalDegrees = ((dSeconds / 3600) + (dMinutes / 60) + (dDegrees))*
Direction;
return Math.Round(dDecimalDegrees, 2);
}

// Catch block works with Try block. If any error arrives, it catches
and Return error response according to user specification
catch{return 0;}
}

/// <summary>
/// Function returns decimal latiude with direction as (+/-)
/// </summary>
/// <param name="longitude">Longitude </param>
/// <param name="direction">Direction </param>
/// <returns>returns decimal latiude with direction as (+/-) </returns>
public double ReturnDecimalLongitude (string longitude,string
direction)
{
// Try block to catch any irrelevant error
try
{
// local variable assignment
string sLongitude = longitude;
string sLongDirection  = direction;
// if empty
if (longitude == string.Empty || direction == string.Empty)return 0;
// Check length
if(longitude.Length < 4)return 0;
// Check direction
if(direction.Length  > 1)return 0;
int iDirection = 1;
if(!(direction == "W" || direction == "E"))
return 0;
// Get direction
iDirection = (direction=="W")?-1:1;
// GRPRMC format for latitude is DDMM.SS
// Where DD represents degress, MM reprsents minutes and SS seconds
double dDegrees, dMinutes, dSeconds = 0 ;
double dDecimalDegrees = 0.0;
dDegrees = Convert.ToDouble(longitude.Substring(0,3));
dMinutes = Convert.ToDouble(longitude.Substring(3,2));
if(longitude.IndexOf(".") != -1)
dSeconds = Convert.ToDouble(longitude.Split(".".ToCharArray())[1]);
dDecimalDegrees = (dSeconds / 3600 + dMinutes / 60 + dDegrees) *
iDirection;
return Math.Round(dDecimalDegrees,2);
}
```

```
// Catch block works with Try block. If any error arrives, it catches
and Return error response according to user specification
catch{return 0;}
}
```

Appendix 3

```
// Array list that contains record of each GPRMC sentence
private ArrayList m_alTime = new ArrayList();
private ArrayList m_alLatitude = new ArrayList();
private ArrayList m_alLongitude = new ArrayList();
private ArrayList m_alSpeed = new ArrayList();
private ArrayList m_alBearing = new ArrayList();
private ArrayList m_alDate = new ArrayList();
private ArrayList m_alMagneticVariation = new ArrayList();
private ArrayList m_alCheckSum = new ArrayList();
private ArrayList m_alCorrect = new ArrayList();

/// <summary>
/// This function will shuffle ANN records and create
/// comma seperated file (CSV) of shuffled records
/// </summary>
private void ShuffleRecords()
{
// Create collection of each GPRMC error
AddRecords();
// Create unique file name
string strFileName = "ANNShuffledTraingDataLog " +
DateTime.Now.ToString("dd-MM-yyyy-hh-mm-ss");
strFileName = Application.StartupPath + "\\" + strFileName+".txt";
// Write File header
string sFileHeader =
"Time,Latitude,Longitud,Speed,Bearing,Date,MagVar,CheckSum,
Output,ErrorTyp";
WriteToFile(strFileName,sFileHeader);
int iRandom = 0;
while(m_alBearing.Count !=0 ||
m_alCheckSum.Count !=0 ||
m_alCorrect.Count != 0 ||
m_alDate.Count != 0 ||
m_alLatitude.Count != 0 ||
m_alLongitude.Count != 0 ||
m_alMagneticVariation.Count != 0 ||
m_alSpeed.Count != 0 ||
m_alTime.Count != 0 )
{
/*Time,Latitude,Longitud,Speed,Bearing,Date,MagVar,CheckSum,Output,
ErrorTyp*/
if(m_alTime.Count > 0)
{
iRandom = new Random().Next(0,m_alTime.Count-1);
WriteToFile(strFileName,m_alTime[iRandom].ToString());
m_alTime.RemoveAt(iRandom);
}
if(m_alCorrect.Count > 0)
{
iRandom = new Random().Next(0,m_alCorrect.Count-1);
WriteToFile(strFileName,m_alCorrect[iRandom].ToString());
m_alCorrect.RemoveAt(iRandom);
```

```
}
if(m_alLatitude.Count>0)
{
iRandom = new Random().Next(0,m_alLatitude.Count-1);
WriteToFile(strFileName,m_alLatitude[iRandom].ToString());
m_alLatitude.RemoveAt(iRandom);
}
if(m_alCorrect.Count > 0)
{
iRandom = new Random().Next(0,m_alCorrect.Count-1);
WriteToFile(strFileName,m_alCorrect[iRandom].ToString());
m_alCorrect.RemoveAt(iRandom);
}
if(m_alLongitude.Count > 0)
{
iRandom = new Random().Next(0,m_alLongitude.Count-1);
WriteToFile(strFileName,m_alLongitude[iRandom].ToString());
m_alLongitude.RemoveAt(iRandom);
}
if(m_alCorrect.Count > 0)
{
iRandom = new Random().Next(0,m_alCorrect.Count-1);
WriteToFile(strFileName,m_alCorrect[iRandom].ToString());
m_alCorrect.RemoveAt(iRandom);
}
if(m_alSpeed.Count > 0)
{
iRandom = new Random().Next(0,m_alSpeed.Count-1);
WriteToFile(strFileName,m_alSpeed[iRandom].ToString());
m_alSpeed.RemoveAt(iRandom);
}
if(m_alCorrect.Count > 0)
{
iRandom = new Random().Next(0,m_alCorrect.Count-1);
WriteToFile(strFileName,m_alCorrect[iRandom].ToString());
m_alCorrect.RemoveAt(iRandom);
}
if(m_alBearing.Count > 0)
{
iRandom = new Random().Next(0,m_alBearing.Count-1);
WriteToFile(strFileName,m_alBearing[iRandom].ToString());
m_alBearing.RemoveAt(iRandom);
}
if(m_alCorrect.Count > 0)
{
iRandom = new Random().Next(0,m_alCorrect.Count-1);
WriteToFile(strFileName,m_alCorrect[iRandom].ToString());
m_alCorrect.RemoveAt(iRandom);
}
if(m_alDate.Count > 0)
{
iRandom = new Random().Next(0,m_alDate.Count-1);
WriteToFile(strFileName,m_alDate[iRandom].ToString());
m_alDate.RemoveAt(iRandom);
}
if(m_alCorrect.Count > 0)
{
```

```
iRandom = new Random().Next(0,m_alCorrect.Count-1);
WriteToFile(strFileName,m_alCorrect[iRandom].ToString());
m_alCorrect.RemoveAt(iRandom);
}
if(m_alMagneticVariation.Count > 0)
{
iRandom = new Random().Next(0,m_alMagneticVariation.Count-1);
WriteToFile(strFileName,m_alMagneticVariation[iRandom].ToString());
m_alMagneticVariation.RemoveAt(iRandom);
}
if(m_alCorrect.Count > 0)
{
iRandom = new Random().Next(0,m_alCorrect.Count-1);
WriteToFile(strFileName,m_alCorrect[iRandom].ToString());
m_alCorrect.RemoveAt(iRandom);
}
if(m_alCheckSum.Count > 0)
{
iRandom = new Random().Next(0,m_alCheckSum.Count-1);
WriteToFile(strFileName,m_alCheckSum[iRandom].ToString());
m_alCheckSum.RemoveAt(iRandom);
}
if(m_alCorrect.Count > 0)
{
iRandom = new Random().Next(0,m_alCorrect.Count-1);
WriteToFile(strFileName,m_alCorrect[iRandom].ToString());
m_alCorrect.RemoveAt(iRandom);
}
}
}

/// <summary>
/// This function will populate above collection of arrays for
/// each GPRMC errors
/// </summary>
private void AddRecords()
{
// File open streams
FileStream ofsFile = new FileStream(txtFile.Text,
FileMode.Open,
FileAccess.Read,
FileShare.Read );
// File reading streams
StreamReader ofrFile = new StreamReader(ofsFile);
string sSentence = ofrFile.ReadLine();
while(sSentence != null)
{
if(IsANNTrainingData(sSentence ) == true)
{
string[] saSentence = sSentence.Split(",".ToCharArray());
RMCErrorType rmcError = (RMCErrorType)( int.Parse(saSentence[9]));
switch(rmcError)
{
case RMCErrorType.RMCTime:
m_alTime.Add(sSentence);
break;
case RMCErrorType.RMCLatitude :
```

```
m_alLatitude.Add(sSentence);
break;
case RMCErrorType.RMCLongitude:
m_alLongitude.Add(sSentence);
break;
case RMCErrorType.RMCSpeed:
m_alSpeed.Add(sSentence);
break;
case RMCErrorType.RMCBearing:
m_alBearing.Add(sSentence);
break;
case RMCErrorType.RMCMagneticVariation:
m_alMagneticVariation.Add(sSentence);
break;
case RMCErrorType.RMCDate:
m_alDate.Add(sSentence);
break;
case RMCErrorType.RMCCheckSum:
m_alCheckSum.Add(sSentence);
break;
case RMCErrorType.RMCNull:
m_alCorrect.Add(sSentence);
break;
}
}
// Read line
sSentence = ofrFile.ReadLine();
}
ofrFile.Close();
ofsFile.Close();
}

/// <summary>
/// This function will check whether the data is ANN training data or
not
/// </summary>
/// <param name="sentence">ANN training data</param>
/// <returns>Returns true if the data is ANN supported</returns>
public static bool IsANNTrainingData(string sentence)
{
try
{
if(sentence.Trim() == string.Empty)
return false;
string[] saSentence = sentence.Split(",".ToCharArray());
if(saSentence.Length != 10)
return false;
if(saSentence[0].Trim() != "" )
{
if (double.IsNaN(double.Parse(saSentence[0])) == true)
{
return false;
}
}
return true;
}
catch {return false;}
```

```
}

/// <summary>
/// This function will write the ANN training sentence in a file
/// </summary>
/// <param name="sFilePath">File path with its name</param>
/// <param name="sSentence">GPRMC ANN formated trainning sentence
</param>
/// <returns>Returns true if successfully written</returns>
public static bool WriteToFile(string sFilePath, string sSentence)
{
if(sSentence.Trim() == string.Empty)return false;
// Create filestream for writer
FileStream ofsClientFile = new FileStream(sFilePath,
FileMode.Append,
FileAccess.Write,
FileShare.ReadWrite);
// Create Stream Writer
StreamWriter oswClientFile = new StreamWriter(ofsClientFile);
oswClientFile.WriteLine(sSentence);
// Flush write
oswClientFile.Flush();
// Close
oswClientFile.Close();
return true;
}
```