Sudan University of Science and Technology
College of Graduate Studies

Offline Fraud Call Detection By Using Artificial Neural Network

الكشف عن مكالمات الاحتيال في وضع عدم الاتصال باستخدام الشبكات العصبية الاصطناعية

A Thesis Submitted in Partial fulfillment for the Requirements of the Degree of M.Sc. in Electronics Engineering (Computer and Networks Engineering)

Prepared By:

Hafsa Hassan Abdalmomen Mosa

Supervised By:

Dr. Elsadiq Saeid

June 2022

الآيـــــــة

بسم الله الرحمن الرحيم

قـــال تعالــــى

(قَالُواْ سُبْحَانَكَ لاَ عِلْمَ لَنَا إِلاَّ مَا عَلَّمْتَنَا إِنَّكَ أَنتَ الْعَلِيمُ الْحَكِيمُ)

صدق الله العظيم

سورة البقرة – الآية (32)

Dedication

Dedication to my mother…With warmth and faith...

Dedication to my father…With love and respect …

Dedication to my friends…Whom we cherish their friendship

Dedication to my special people…Who mean so much to me…

Dedication to all my teachers …In whom I believe so much …

Acknowledgement

I extend my thanks to all who stood with me to achieve this research which it comes because of grace of God and reconcile.I would like to give special thanks to my SupervisorDr. Elsadig Saeid for his great help and support. And my teachers that gave me information and all staff in Sudan University.Finally, yet importantly I dedicate this project for everyone that helped me to be at the place that I am today.

Abstract

Telecommunication Fraud can be defined as an illegal use of telecom infrastructure likemobile communications with an intention for not paying services, misuse of voice calls (or data, SMS, MMS), cheating in subscriptions and using illegally services in the networks of telecom providers.

Telecommunication fraud has continuously been causing significant financial loss to telecommunication customers in the world for several years.Traditional approaches to detect telecommunication frauds usually rely on constructing a blacklist of fraud telephone numbers. However, attackers can simply evade such detection by changing their numbers, which is very easy to achieve through VoIP (Voice over IP).

To solve this problemfeed-forward neural network is usedas a software capable of detecting fraud call for offline data, calldetailed recordswere collected, prepared to be coded and analyzed through extracting features, then rules has been built to check whether the call was normal or fraud. constituting Feed-forward neural network system was done, data has beendivided into two datasets which were Training data and Testing data, learning and validation for neural network were done, then the mean square error has been measured.This technique was designed for telecommunication fraud call detection, depending on analyzing contents of a call Instead of relying on call type and caller number then constructing a blacklist of fraud numbers.It was found that the mean square error has reduced by increasing the percentage of testing data, as well as increasing training data leads to minimizing the validation and the mean square error would be minimized, the obtained results had led to increased system accuracy to detect fraud call.

المستخلص

يتسبب الاحتيال في مجال الاتصالات السلكية واللاسلكية باستمرار في خسائر مالية كبيرة لعملاء الاتصالات في العالم منذ عدة سنوات. تعتمد الطرق التقليدية للكشف عن عمليات الاحتيال في مجال الاتصالات عادة على إنشاء قائمة سوداء بأرقام هواتف المتصلين المحتالين وفقا لرقم الهاتف ونوع المكالمة. ومع ذلك ، يمكن للمهاجمين ببساطة التهرب من هذا الكشف عن طريق تغيير أرقامهم و هو أمر ممكن جدا من خلال تقنية الصوت عبر بروتوكول الانترنت VoIPلحل هذه المشكلة وتطوير برنامج قادر على اكتشاف عمليات الاحتيال التي تتطلب بيانات غير متصلة بالإنترنت ، تم جمع سجلات تفاصيل بيانات العملاء ، وإعدادها لتحليلها من خلال استخراج الميزات ، ثم تم تشكيل نظام الشبكة العصبية ذات التغذية الامامية، وتم تقسيم البيانات إلى مجموعتين من البيانات وهما التدريب والاختبار. تم إجراء التعلم والتحقق من صحة الشبكة العصبية ، ثم تم قياس الخطأ. تم تصميم هذه التقنية للكشف عن المكالمات الاحتيالية في مجال الاتصالات السلكية واللاسلكية ، اعتمادا على تحليل محتويات المكالمة بدلا من الاعتماد على نوع المكالمة ورقم المتصل ثم إنشاء قائمةسوداء بأرقام الاحتيال. و وقد وجد أن متوسط الخطأ التربيعي قد انخفض عن طريق زيادة النسبة المئوية لبيانات الاختبار ، مما أدى إلى زيادة دقة النظام.

Table ofContents

List of Figures

List of Tables

LIST OF ABBREVIATION

AC	Accuracy
ANN	Artificial Neural Network
BP	Back Propagation
CDR	Call Data Record
CNN	Convolutional Neural Network
CSV	Comma Separated Values
FP	False Positive
FN	False Negative
FFNN	Feed-forward Neural Network
KDD	Knowledge Discovery in Databases
MCC	Mathew’s Correlation Coefficient
MNN	Modular Neural Network
MMS	Multimedia Messaging Services
NN	Neural Network
SE	Sensitivity
SMS	Short Message Services
SP	Specificity
TN	True Negative
TP	True Positive
USD	United States Dollar
VoIP	Voiceover Internet Protocol

Chapter One
Introduction

Chapter One
Introduction

1.1 Preface

Due to the increased development of new technologies recently, many fraudulent activities have been arisingincluding online banking,e-commerce credit card transactions frauds, in addition to the telecommunicationfraud, leading to multi-billion losses worldwide every year.

In fact, Fraud is very costly to all telecom carriers in term of capacity and income lost. According to the reportAccording to the data released by the Ministry of Public Securitypublished recently by Neural Technologies in 2016, the average loss of telecom industry was estimated to $249 billion dollars USD due to fraud activities. Telecom companies generate a huge amount of raw data including voice calls, SMS, recharge events, subscriptions and other services. The high volume of data collected and available in each day, which needs to be processed and manipulated, constitutes a big challenge in the telecom industry, and as a consequence, many fraudulent events can take place at any service, leading to a considerable loss of revenue to companies. Therefore, designing an accurate machinelearning model is essential to improve service usage monitoring and show significant revenue protection, in order to detect fraudulent events and activities in time[1].

In order to detect telecommunication frauds, most of the current approaches are based on labeling the caller numbers that are identified as frauds by customers. At the same time, there are also many researchers who use machine learning techniques to detect fraudulent calls. They select features based on factors such as phone numbers and call types. They use machine learning algorithms to train models, and use these models to detect fraudulent calls, which can also achieve good detection accuracy. However, as the number change software is widely used, fraudsters use software to change their phone number constantly or disguise their number as the official number of government agencies [2].

These reasons make it possible for conventional telephone number-based detection methods can be easily bypassed. However, learn of the telecommunication fraud from the reports and news on the Internet for understanding the contents of a call. Particularly, first, a collect of descriptions of telecommunication fraud from the Internet and network operators. In our study, offline data was collected from network operators and saved into CSV filesin order to be analyzed.[2]

1.2 Problem Definition

Telecommunication fraud has continuously been causing severe financial loss to telecommunication customers in the world for several years. Traditional approaches to detect telecommunication frauds usually rely on call type and phone number. However, attackers can simply evade such detection by changing their numbers, which is very easy to achieve through VoIP (Voice over IP).

1.3 Proposed Solution

In this research it intended to detect telecommunication fraud through analyzing the contents of a call by using artificial neural network. However, this is quite challenging, mainly due to the complexity of the contents of a call impedes the analysis.

1.4 Objectives

Themain goal isanalyzingoffline data for a given subscriber or customer and classified it as fraudster based on his features derived from the CDR and identify telecommunication frauds only through the contents of a call, and increasing system accuracy by reducing error value.

1.5 Methodology

Feed-forward neural network has been used for detecting fraud call depending on analyzing the content of call data record instead of relying on source caller number and has passed through different steps, call data record has been collected, then data has been analyzed, coded to be readable for matlab software and divided into sub sets such as testing dataset and training dataset, natural language has been used to extract features from the call data records, rules have been built to identify similar contents within the same call, the feed-forward neural network structure has been built which contained three layers which were input layer, hidden layer and output layer. Then back propagation algorithm has been used to apply learning to neural network, validation has been done and the error value has been measured by calculating the difference between the obtained result and the target output.

1.6 Thesis outline

The project thesis is divided into five chapters, in chapter one an introduction with project problems, proposed solution and objectives were written, while chapter two represents the background and literature review of the study,chapter three represents the methodology and requirements are listed, chapter four show the design of ANN and implementation were written, while chapter five represents a conclusion and recommendations.

Chapter Two
Background and literature Review

Chapter Two

Background andLiterature Review

2.1 Background

2.2 Fraud

Fraud can be defined as an illegal use of telecom infrastructure likemobile communications with an intention for not paying services, misuse of voice calls (or data, SMS, MMS), cheating in subscriptions and using illegally services in the networks of telecom providers [10].

2.3 Types of Frauds

There are numerous types of fraud on the e-commerce naturethat introduce telecommunication frauds, Credit card frauds, computer intrusion, Bankruptcy fraud, Theft fraud/counterfeit fraud, Application fraud and behavioral fraud[10].

- Credit Card Fraud:

Credit card fraud has been split into two types: Offline fraud and On-line fraud. Offline Fraud is because of by using a stolen physical card at any place. Online Fraud is because of via internet, phone, shopping or web[10].

- Telecommunication Fraud Call:

Telecommunication fraud involves the misuse, who has intention to harm someone by mobile phone fraud and fixed line fraud[10].

- Computer Intrusion:

The act entering without invitation by any outsider or hacker and insider who knows the system structure any Environment is defines asintrusion. That Means “Potential Possibility of Unauthorized Attempt to Access Information, Manipulate Information Purposefully”[10].

- Bankruptcy Fraud:

Bankruptcy fraud is difficult to predict the fraud, when bank send a customer an order, user will be of personal bankruptcy fraud. Also, it becomes difficult to get unwanted loans. One of the potential ways to avoid this fraud is by doing a pre-check with credit bureau. This informed about the past banking history of its customers[10].

- Theft Fraud/Counterfeit Fraud:

If the used cards are not yours then the fraud is called theft. The bank will take measures to check the thief when the owner gives some feedback and contact the bank. Likewise, remotely use of the credit card leads to counterfeit fraud. Use of your codes via various web-sites and copied card number, where no physical cards or signature are required[10].

- Application Fraud:

Application fraud is defined as when someone applies for a credit card with wrong information. Detection can be done by either when applications come from a same user with the same details, termed as duplicates and when applications come from different individuals with similar details, called as identity fraudsters [10].

2.4**Artificial Neural Network**

Artificial neural networks (ANNs), usually simply called neural networks (NNs), are computing systems vaguely inspired by the biological neural networks that constitute human brains [12].An ANN is based on a collection of connected units or nodes called artificial neurons, which loosely model the neurons in a biological brain [11]. Each connection, like the synapses in a biological brain, can transmit a signal to other neurons. An artificial neuron that receives a signal then processes it and can signal neurons connected to it. The "signal" at a connection is a real number, and the output of each neuron is computed by some non-linear function of the sum of its inputs. The connections are called edges. Neurons and edges typically have a weight that adjusts as learning proceeds[13]. The weight increases or decreases the strength of the signal at a connection. Neurons may have a threshold such that a signal is sent only if the aggregate signal crosses that threshold. Typically, neurons are aggregated into layers. Different layers may perform different transformations on their inputs. Signals travel from the first layer (the input layer), to the last layer (the output layer), possibly after traversing the layers multiple times[14].

2.4.2 Training

Neural networks learn (or are trained) by processing examples, each of which contains a known "input" and "result," forming probability-weighted associations between the two, which are stored within the data structure of the net itself. The training of a neural network from a given example is usually conducted by determining the difference between the processed output of the network (often a prediction) and a target output. This is the error. The network then adjusts its weighted associations according to a learning rule and using this error value. Successive adjustments will cause the neural network to produce output which is increasingly similar to the target output. After asufficient number of these adjustments the training can be terminated based upon certain criteria. This is known as supervised learning [15].

Such systems "learn" to perform tasks by considering examples, generally without being programmed with task-specific rules. For example, in image recognition, they might learn to identify images that contain cats by analyzing example images that have been manually labeled as "cat" or "no cat" and using the results to identify cats in other images. They do this without any prior knowledge of cats, for example, that they have fur, tails, whiskers and cat-like faces. Instead, they automatically generate identifying characteristics from the examples that they process [16].

2.5. Components of Artificial Neural Networks

2.5.1 Neurons

ANNs are composed of artificial neurons which are conceptually derived from biological neurons. Each artificial neuron has inputs and produces a single output which can be sent to multiple other neurons. The inputs can be the feature values of a sample of external data, such as images or documents, or they can be the outputs of other neurons. The outputs of the final output neurons of the neural net accomplish the task, such as recognizing an object in an image[17].

To find the output of the neuron, first we take the weighted sum of all the inputs, weighted by the weights of the connections from the inputs to the neuron. We add a bias term to this sum. This weighted sum is sometimes called the activation. This weighted sum is then passed through a (usually nonlinear) activation function to produce the output. The initial inputs are external data, such as images and documents. The ultimate outputs accomplish the task, such as recognizing an object in an image [18].

2.5.2 Connections and weights

The network consists of connections, each connection providing the output of one neuron as an input to another neuron. Each connection is assigned a weight that represents its relative importance. A given neuron can have multiple input and output connections

2.5.3 Organization

The neurons are typically organized into multiple layers, especially in deep learning. Neurons of one layer connect only to neurons of the immediately preceding and immediately following layers. The layer that receives external data is the input layer. The layer that produces the ultimate result is the output layer. In between them are zero or more hidden layers. Single layer and un-layered networks are also used. Between two layers, multiple connection patterns are possible. They can be fully connected, with every neuron in one layer connecting to every neuron in the next layer. They can be pooling, where a group of neurons in one layer connect to a single neuron in the next layer, thereby reducing the number of neurons in that layer. Neurons with only such connections form a directed acyclic graph and are known as feed-forward networks. Alternatively, networks that allow connections between neurons in the same or previous layers are known as recurrent networks [18][19].

2.5.4 Hyper-parameter

A hyper parameter is a constant parameter whose value is set before the learning process begins. The values of parameters are derived via learning. Examples of hyper parameters include learning rate, the number of hidden layers and batch size. The values of some hyper parameters can be dependent on those of other hyper parameters. For example, the size of some layers can depend on the overall number of layers [18][19].

2.6 Types of Artificial Neural Network

2.6.1 Feed-forward Neural Network

Feed-forward Neural Network are the mostly encountered type of artificial neural networks and applied to many diverfields. ANNs, are inspired from their biological counterparts, the biological brain and the nervous system. Biological brain is entirely different than the conventional digital computer in terms of its structure and the way it processes information. The most important distinctive feature of a biological brain is its ability to learn and adapt while a conventional computer does not have such abilities. There are two categories of network architectures depending on the type of the connection between the neurons, “feed-forward neural network” and the “recurrent neural network”. If there is no “feedback” from the output of the neurons towards the inputs throughout the network, then the network is referred to as “feed-forward neural network”. Otherwise, if there exist such a feedback, i.e., a synaptic connection from the out towards the input, then the network called a “recurrent neural network”. Usually, neural networks are arranged in the form of a “layer” s feed-forward neural networks fall into two categories depending on the number of the layers, either “single layer” or “multi-layer”.

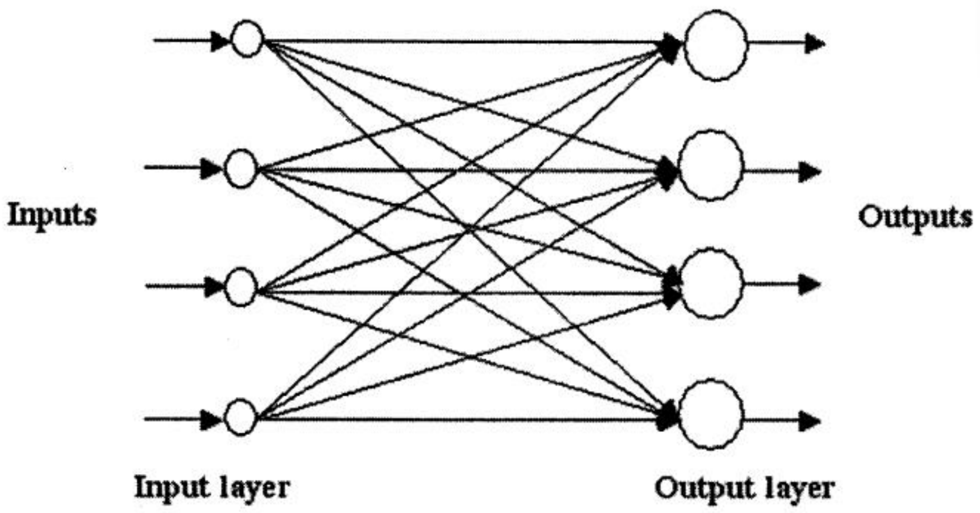
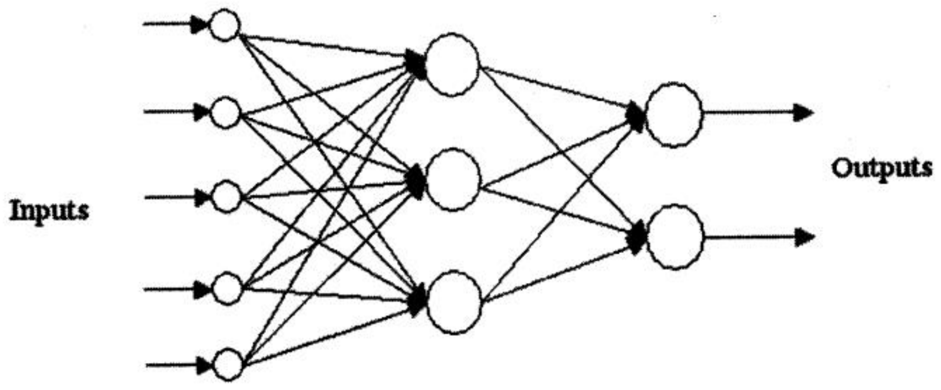


Figure 2.1 Single Layer Feed-forward Neural Network

In Figure 2.2, a single layer feed-forward neural network is shown, including the input layer, there are two layers in this structure. Input signals are passed on to the output signals.



Input Layer **Hidden Layer** **Output Layer**

Figure 2.2 Multi-layer Feed-forward Neural Network

In Figure 2.3 a Multi-layer Feed-forward Neural Network with one "hidden layer" is depicted. As opposed to a single-layer network, there is (at least) one layer of "hidden neurons" between the input and the output layers. In both Figure 2.2 and Figure 2.3, network is "fully connected" because every neuron in each layer is connected to every other neuron in the next forward layer. If someof the synaptic connections were missing, the network would be called as "partially connected"[20].

2.6.2 Modular Neural Network

A Modular Neural Network (MNN) is a Neural Network (NN) that consists of several modules, each module carrying out one sub-task of the NN's global task, and all modules functionally integrated. A module can be a sub-structure or a learning sub-procedure of the whole network. The network's global task can be any neural network application, e.g., mapping, function approximation, clustering or associative memory application.MNN is a rapidly growing field in NNs research. Researchers from several backgrounds and objectives are contributing to its growth. For example, motivated by the "non-neuromorphic" nature of the current artificial NN generation, some researchers with a biology-background are suggesting modular structures. Their goal is either to model the biological NN itself, i.e., a reverse engineering study, or to try to build artificial NNs which achieve the high capabilities of the biological system. Motivated by the psychology of learning in the human system, some other researchers modularize the NN's learning in an attempt to achieve clearer representation of information and less amount of internal interference. Another group of researchers develop modular NNs to fulfill the constraints put by the current hardwareimplementation technology. Nevertheless, most of the work in the MNN field aims to enhance the computational capabilities of thenonmodular alternatives, e.g., enhancing the networks' generalization, scalability, representation, and learning speed. Figure 1 shows the growth of the MNN field in a profile very much similar to the growth in the NN field. Notice that 44% of the MNNs research is done in the last two years. This illustrates the recent high interest in the field. Biologists have studied modularization of the natural brain long time ago. However, the first two attempts to build artificial modular neural networks, we are aware of, were in November 1987. E. Micheli-Tzanakou2 outlined, briefly, a model he built of the vertebrate retina using an artificial MNN. He designed a collection of modules connected in series and parallel and used them to study the effects of lateral connectivity. In the same month, the Third Conference of ArtificialIntelligence for Space Applications, Huntsville, Al, USA, published an abstract written by E. Fiesler and A. Choudry,3 in which they suggested NNs as "a possible architecture" for building multimodular space systems.[21]

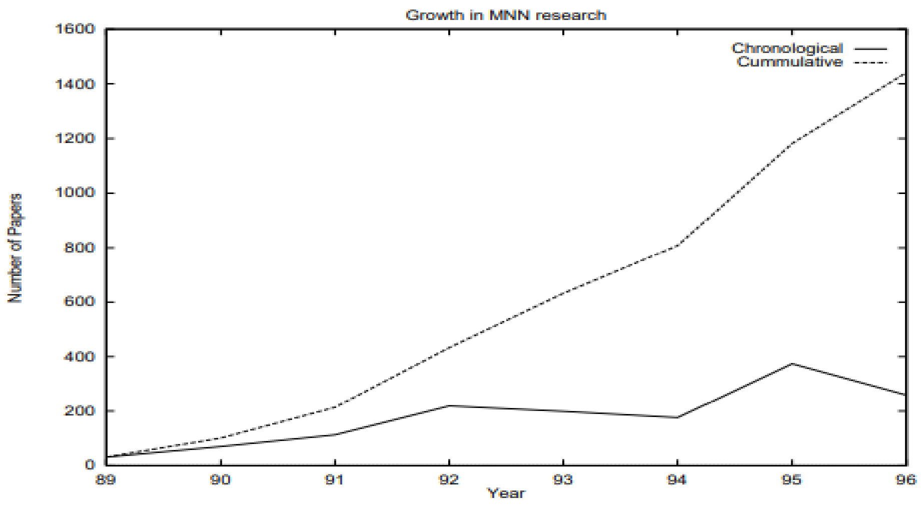


Figure 2.3 Growth in Modular Neural Network

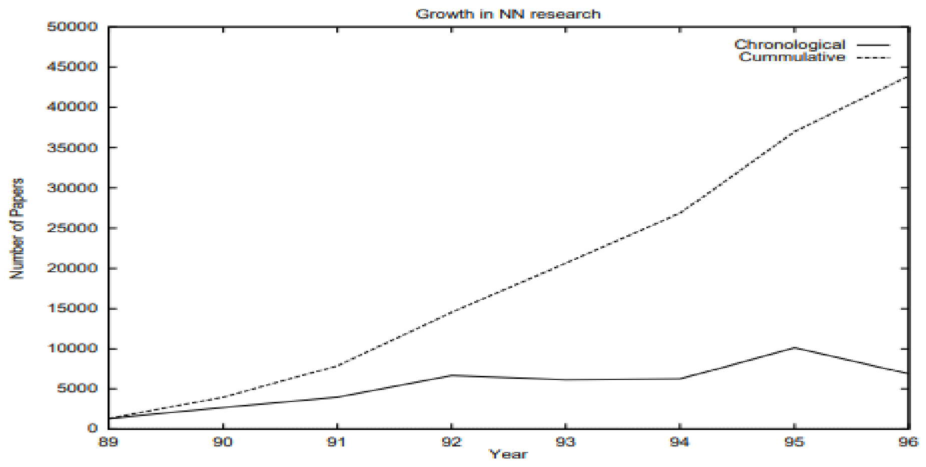


Figure 2.4 Growth in Neural Network

2.6.3 Multi-Layer Perceptron

Feed Forward Neural Network contains neurons and edges that form a network. The neurons are set of nodes and are of three types: input, hidden and output. Each node is a unit of processing. The edges are the links between two nodes and they have associated weights. In Multi-layer perceptron the network consists of multiple layers of computational units, usually connected in a feed-forward way. Each neuron in one layer has direct connections to the neurons of the subsequent layer although not to other nodes in the same layer. There might be more than one hidden layer. A neuron has a number of inputs and one output. It combines all the input values (Combination), does certain calculations, and then triggers an output value (activation). There are different ways to combine inputs. One of the most popular methods is the weighted sum, meaning that the sum of each input value is multiplied by its associated weight. Therefore, for a given node g we have:

$$Net_g = \sum w_{ij}x_{ij_1} = w_{0j}x_{0j} + w_{1j}x_{1j} + \cdots . w_{ij}x_{ij} \qquad (1)$$

wherexij represents the ith input to node j, wij represents the weight associated with the ith input to node j and there are Iinputs to node j. The value obtained from the combination function is passed to non-linear activation function as input.

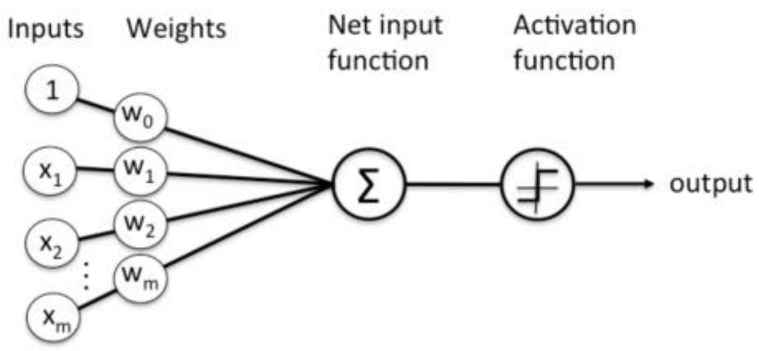


Figure 2.5 Multi-layer Perceptron

2.6.4 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are analogous to traditional ANNs in that they are comprised of neurons that self-optimize through learning. Each neuron will still receive an input and perform an operation (such as a scalar product followed by a non-linear function) - the basis of countless ANNs. From the input raw image vectors to the final output of the class score, the entire of the network will still express a single perceptive score function (the weight). The last layer will contain loss functions associated with the classes, and all of the regular tips and tricks developed for traditional ANNs still apply. The only notable difference between CNNs and traditional ANNs is that CNNs are primarily used in the field of pattern recognition within images. This allows us to encode image-specific features into the architecture. The basic functionality of the example CNN above can be broken down into four key areas.

1. As found in other forms of ANN, the input layer will hold the pixel values of the image.

2. The convolutional layer will determine the output of neurons of which are connected to local regions of the input through the calculation of the scalar product between their weights and the region connected to the input volume. The rectified linear unit aims to apply an element activation function such as sigmoid to the output of the activation produced by the previous layer.

3. The pooling layer will then simply perform downsampling along the spatial dimensionality of the given input, further reducing the number of parameters within that activation. 4. The fully-connected layers will then perform the same duties found in standard ANNs and attempt to produce class scores from the activations, to be used for classification. It is also suggested that ReLu may be used between these layers, as to improve performance[23].

There for, a set of desired outputs must be available for training. For the reason, back-propagation is a supervised learning rule[21].

2.6.5 Deep Neural Network

Due to practical limitation of single-layer network on the linear separable problem, deep neural network (DNN) was introduced to solve an arbitrary classification problem. It contains one or more hidden layers whose computational nodes are called hidden nodes. The depth of the model refers to the number of hidden layers. Figure 2.6shows the topology of deep neural network with two hidden layers and an output layer. The input information enters the first hidden layer and the outputs of this layer are transferred as inputs to the second hidden layer and so on. Each layer receives outputs from previous layer as inputs, thus the input signal propagates forward on layer-by-layer base until the output layer. An error signal is produced at neurons in the output layer which is propagated backward through the network. Deep neural network can be used for pattern recognition and classification. The input layer consists of the components of a feature vector to be classified. Each hidden layer of neurons processes on a different set of features that are output of the previous layer. The more hidden layers the network has, the more complicated features that can be detected by neurons since they collect and combine information generated by previous layer. The nonlinearity from training data gives the network greater computational power and can implement more complex functions than network without hidden neurons[23].

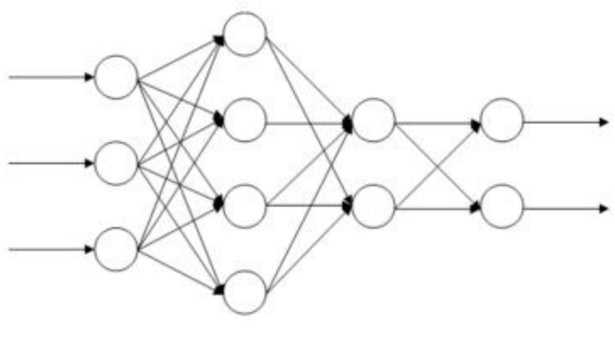


Figure2.6 Structure of Deep Neural Network with Two Hidden Layers

2.7 Learning Methods

The two key learning paradigms in image processing tasks are supervised and unsupervised learning. Supervised learning is learning through pre-labeled inputs, which act as targets. For each training example there will be a set of input values (vectors) and one or more associated designated output values. The goal of this form of training is to reduce the model's overall classification error, through correct calculation of the output value of training example by training.

Unsupervised learning differs in that the training set does not include any labels. Success is usually determined by whether the network is able to reduce or increase an associated cost function. However, it is important to note that most image-focused pattern-recognition tasks usually depend on classification using supervised learning.

2.8 Back-Propagation Algorithm

Among many other learning algorithms, "back-propagation algorithm" is the most popular and the mostly used one for the training of feed forward neural networks. It is in essence, a mean of updating networks synaptic weight by back-propagating a gradient vector in which each element is defined as the derivatives of an error measure with respect to a parameter. Error signals are usually defined as the difference of the actual network outputs and the desired outputs[23]. The propagation function computes the input to a neuron from the outputs of its predecessor neurons and their connections as a weighted sum. A bias term can be added to the result of the propagation [19].

2.9 Sigmoid Function

One of the most common activation functions used by Neural Network is the sigmoid function. This is a nonlinear functions and result in nonlinear behavior. Sigmoid function is used in this study. Following is the definition of sigmoid function:

$$\text{Sigmoid} = \frac{1}{1 + e^{-x}} \tag{2}$$

Where x is the input value and eis the base of natural logarithms, equal to about 2.718281828. The output value from this activation function is then passed along the connection to the connected nodes in the next layer. Back-propagation algorithm is a commonly used supervised algorithm to train feed-forward networks. The whole purpose of neural network training is to minimize the training errors[22].

2.10 Literature Review

2.10.1 Fraud Detection

Fraud detection has always been the subject of some surveys and commentary articles because of the severe damage to the society. Delamaire et al. (2009) proposed different types of credit card frauds, such as bankruptcy fraud, theft fraud/counterfeit fraud, application fraud and behavioral fraud, discussing the feasibility of various techniques to combat this type of fraud, such as decision tree, genetic algorithms, clustering techniques and neural networks. Rebahi et al. (2011) proposed the VoIP fraud and the fraud detection systems to it checking their availability in VoIP environments in various fields [2]. These detection systems are classified as two categories: rule-based supervised and unsupervised methods [3]. LookmanSithic and Balasubramanian (2013) investigated the categories of fraud in medical field and vehicle insurance systems [4]. Various types of data mining techniques were used to detect fraud in these areas according to the results. The financial fraud detection has become the most popular topic in the area of fraud detection (Abdallah et al. 2016) which usually leads to high economic losses [3][4].

A purpose to the detection of telecommunication fraud is that they could be warned by the notification from the application on Android platform when users receive fraudulent calls [5]. The whole process is divided into three parts: the first is the collection and pre-processor of telecommunication fraud data. The second part is extracting features and building rules of detection. The last part is the implementation of telecommunication fraud alert applications [5]. The overview of our approach is shown in Fig. 2.7

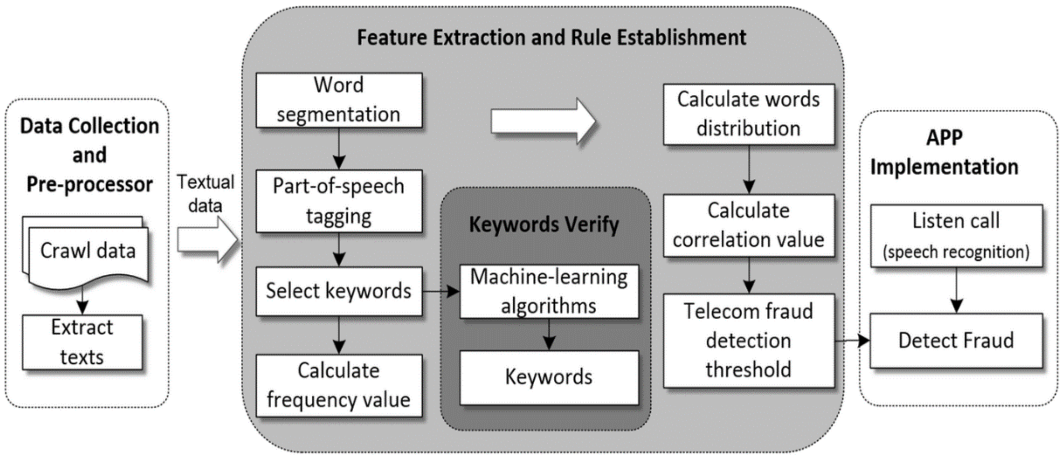

Data Collection and Pre-processor
Crawl data
Extract texts
Textual data
Feature Extraction and Rule Establishment
Word segmentation
Part-of-speech tagging
Select keywords
Calculate frequency value
Keywords Verify
Machine-learning algorithms
Keywords
Calculate words distribution
Calculate correlation value
Telecom fraud detection threshold
APP Implementation
Listen call (speech recognition)
Detect Fraud

Figure 2.7: Feature Extraction

The first step is the collection of telecommunications fraud data. In order to analyze the characteristics and modes of telecommunication fraud, the first thing to do is collecting textual data. Data collection is mainly to collect telecommunication fraud-related calls. The target data includes the case of fraudulent calls, the description language of telecommunication fraud, and the news on the media. In the data collection process, web crawler technology is used to collect data, and search engines (such as Baidu, etc.) are helped to collect textual data on telecommunication fraud on the Internet [6].

The second step is feature extraction and rule-building. After the data collected in the first step, it is important to extract the features and build rules for detecting telecommunication fraud. This research uses natural language processing technology to extract features which are keywords from fraud call. And we use machine learning algorithms to prove the appropriateness of textual data we collected and the validity of keywords we extract [7]. Then, according to the features which are extracted from the call, this research builds the detection rules of telecommunication fraud [8].

The last part is the implementation of telecommunication fraud detection. A telecommunication fraud alert application on the Android platformdeveloped. In detail, the application first starts to monitor the incoming call when a call coming to the users' phone. Then the application uses speech recognition technology to convert the caller's voice into text. After that, the application uses the detection rules that built in the previous step to determine if it is a fraudulent call or not. If the application predicates that it is a fraudulent call, a warning information will pop up on the smartphone's screen to prompt the user to pay attention to this call.[9]

Fawcett and Provost (1997) present fraud rule generation from each cloned phone account's labelled data and rule selection to cover most accounts. Each selected fraud rule is applied in the form of monitors (number and duration of calls) to the daily legitimate usage of each account to find anomalies. The selected monitors' output and labels on an account's previous daily behavior are used as training data for a simple Linear Threshold Unit. An alarm will be raised on that account if the suspicion score on the next evaluation day exceeds its threshold. In terms of cost savings and accuracy, this method performed better than other methods such as expert systems, classifiers trained without account context, high usage, collision detection, velocity checking, and dialed digit analysis on detecting telecommunications superimposed fraud. [23]

2.11 Data Analysis Techniques for Fraud Detection

Fraud that involves cell phones, insurance claims, tax return claims, credit card transactions, government procurement etc. represent significant problems for governments and businesses and specialized analysis techniques for discovering fraud using them are required. These methods exist in the areas of Knowledge Discovery in Databases (KDD), Data Mining, Machine Learning and Statistics. They offer applicable and successful solutions in different areas of electronic fraudcrimes[9][11].

The classification module was designed with a hierarchical tree structure, including three layers and six nodes, as shown in Figure 2.8

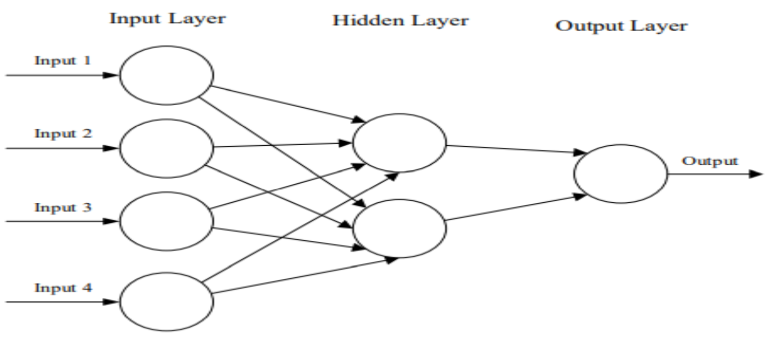


Figure 2.8 Fee-forward Neural Network

The first layer consists of the root node, which discriminates between fraudulent and normal subscribers, but assigns the insolvent subscribers to any of the two groups. The second layer has two nodes. Node N/I discriminate between normal and insolvent cases. Node F/I discriminate between fraudulent and insolvent cases. The third layer has two nodes that discriminate among subscription fraudulent, otherwise fraudulent and insolvent cases. Node I/O distinguishes between insolvent and otherwise fraudulent. Node S/O discriminates between subscription fraudulent and otherwise fraudulent.as well as to design fuzzy rules to discriminate among the categories.the proposed system is predictive and operates at the application time. Demographics and commercial antecedents, as well as other characteristics associated to the application for a new phone line, were used as predictors. The predictive module was able to identify 3.5% of the subscribers containing 56.2% of the true fraudsters. A manual analysis of errors showed that most of the FP cases corresponded to the insolvent category. One third of these corresponded to customers that never paid the bills but had a typical residential average expenditure. This pattern corresponds to the category of fraud for personal usage, and couldbe considered as kind of subscription fraud.

[24].the contact phone number which is required by the telecom in the application process, allowed the detection of sequences of cases of subscription fraud. Fig2.3 shows a sequence of seven fraud cases committed within a period of three months which are related to each other through the contact phone number used for ordering new lines. Typically a fraudster ordered a new phone line and committed fraud a few days after the installation. In the meantime, previous to the

blocking, the fraudster ordered another line using the first line installed as the contact phone number.

Fraud cases would generally be detected online triggered by traffic measures by the commercial fraud detection system, and confirmed later on as such during the billing process. In order to generate a database of known fraudulent/legitimate cases, it was necessary to formalize the definition of subscribers' categories. Consequently, the following four categories of subscribers were defined:

- Subscription fraudulent. Most of the users in this category do not pay their bills at all, but if they do, the debt/payment ratio is very high. The line is typically blocked due to suspicious behavior in long distance calls within 6 months after the installation date.
- Otherwise, fraudulent. Subscribers for more than a year who present a sudden changein their calling behavior, generating an abnormal rise in their newer billing accounts. Insolvent. Subscribers with a total debt of less than 10 times their monthly payments, having two or more unpaid bills. This category includes new customers that have never paid their bills but whose monthly expenditures are similarto average residential lines.
  - Normal Customers with their bills up to date or at most asingle unpaid bill for less than 30 days after due date.

The proposed system for preventing subscription fraud consists of two modules: a classification module and a prediction module. The classification module separates subscribers according to their historical behavior into one out of four of the defined categories: subscription fraudulent, otherwise fraudulent, insolvent or normal. This module uses as inputs the information available about bills, payments, phone line blockings. The main purpose of the classification module is to generate a database of known fraudulent/legitimate cases. The prediction module allows the identification of potential fraudulent subscribers at the time of application. This module uses as inputs the information available about new subscribers, such as demographics and commercial antecedents, as well as application and account information [25].

CHAPTER THREE
Methodology

Chapter Three

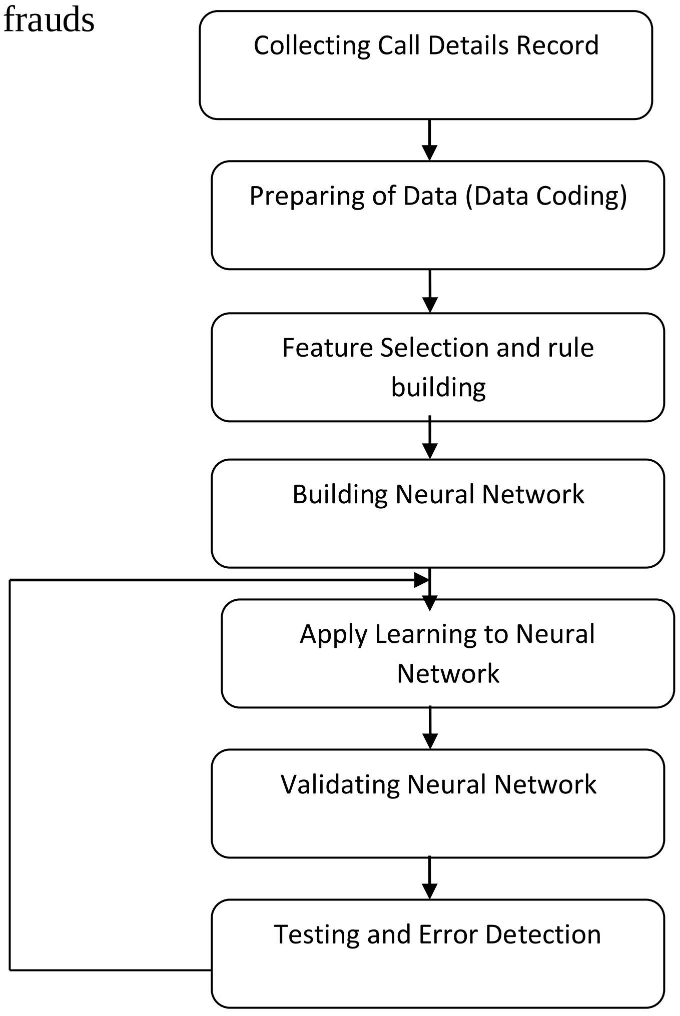
Methodology

3.1 Introduction

This chapter represents the methodology of the project including the creation of artificial neural network, loading and preparing data.

3.2 Methodology Framework

The framework illustrates the steps designing and building the code to detect frauds



Collecting Call Details Record
Preparing of Data (Data Coding)
Feature Selection and rule building
Building Neural Network
Apply Learning to Neural Network
Validating Neural Network
Testing and Error Detection

Figure 3.1: Framework of Methodology

3.3 Neural Network -Detecting with prediction Call Data Record Fraud Data

A neural network is essentially a highly variable function for mapping almost any kind of linear and nonlinear data. It can be used to recognize and analyze trends, recognize images, data relationships, and prediction of intrusions. It is one of the largest developments in artificial intelligence.

A simple neural network design was created based on three-layers neural network in MATLAB, and using it to detect and predict trends in CDR data in order to predict fraud. Sample dataset that used is real CDR records, in which contains some important parameters, such as

- Call duration
- Repeating
- Call source
- Same number to multi-destinationnumber with a time duration almost fixed
- day time call was placed
- is the number reachable
- during part of the day
- number is Listed in international spam numbers

3.4 Preparing of Data (Data Coding)

Table 3.1 Call Duration

Option	Value
Less than minute	-1
Between 1 to 9	0
More than 9 minutes	1

Table 3.2 Call Repeating

Option	Value
More than 18 time or equal or greater than 36(yes)	0
More than 18 time or equal or greater than 36(No)	1

Table 3.3 Call Source

Option	Value
Call source is the same (Yes)	0
Call source is the same (No)	1

Table 3.4 Single Number to Multi Number International

Option	Value
Single number to multi number international (Yes)	0
Single number to multi number international (No)	1

Table 3.5 Number Reachability

Option	Value
Reachable number (Yes)	0
Reachable number (No)	1

Table 3.6 Time During the Day

Option	Value
Morning	-1
Mid-day	0
End of the Day	1

3.5 Implementation of MATLAB Code

Step 1: Importing Data into MATLAB

Call duration: 1) less than min, 2) between 1 to 3, 3) between 3 to 9, 4) fall. (-1, -0.33, 0.33, 1)
Call repeating. 18 time or equal or greater than 36 (0, 1)

Call source is the same: 1) yes, 2) no. (0, 1)

Single number to multi number international 1) yes, 2) no. (0, 1)

Reachable number 1) yes, 2) no. (0, 1)

Time during the day 1) morning, 2) mid-day, 3) end of the day. (-1, 0, 1)

The data is already in a computer-readable format, and looks like:

```
-0.33,0.69,0,1,1,0,0.8,0,0.88,N
-0.33,0.94,1,0,1,0,0.8,1,0.31,O

-0.33,0.5,1,0,0,0,1,-1,0.5,N

-0.33,0.75,0,1,1,0,1,-1,0.38,N

-0.33,0.67,1,1,0,0,0.8,-1,0.5,O

-0.33,0.67,1,0,1,0,0.8,0,0.5,N

-0.33,0.67,0,0,0,-1,0.8,-1,0.44,N
```

In order to make reading it for MATLAB easier, it's required to need to modify the document a bit. The instructions list the output as either a "N" for normal, or an "O" for Fraud.So, a change of the two values to a 0 and a 1, respectively. Now it should look like:

```
-0.33,0.69,0,1,1,0,0.8,0,0.88,0
-0.33,0.94,1,0,1,0,0.8,1,0.31,1

-0.33,0.5,1,0,0,0,1,-1,0.5,0

-0.33,0.75,0,1,1,0,1,-1,0.38,0

-0.33,0.67,1,1,0,0,0.8,-1,0.5,1

-0.33,0.67,1,0,1,0,0.8,0,0.5,0

-0.33,0.67,0,0,0,-1,0.8,-1,0.44,0
```

Now data is ready to be imported MATLAB. A creation of MATLAB script and imported the data with the following code:

```
% input data
    filename = 'CDR_Diagnosis.txt';
delimiterIn = ',';
Data = importdata(filename,delimiterIn);
```

Step 2: Neural Network Structure

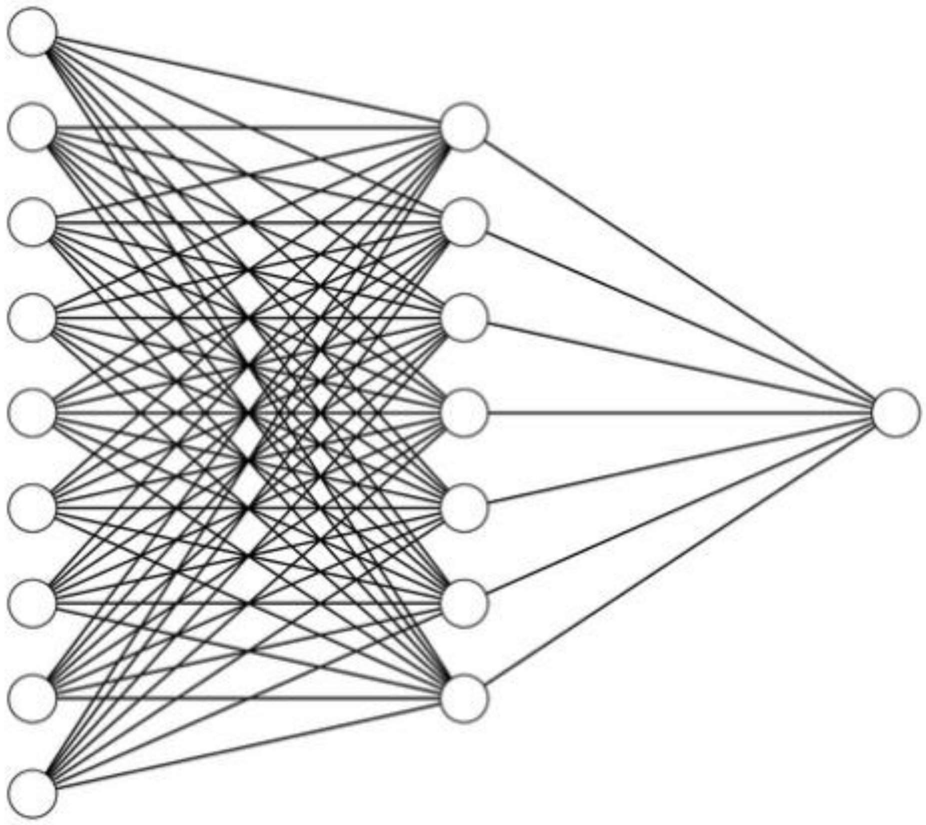


Figure 3.2 Neural Network Structure

Starting on the left, represent the input nodes (circles). Into these nodes the feeding subject data such as the

- Call duration
- Repeating
- Call source

These nodes are connected to the ones to the right by synapses (lines). These synapses can be reprogrammed (by changing their value) to change the behavior of the function (neural network). Modifying these synapses is the function of train the neural network.

Next layer is what is called the hidden layer. It adds depth to the processing and a sort of "second layer of abstraction" to processing data. Some neural networks do not have hidden layers, but for a neural network to be able to graph non-linear data relationships, it is a necessity.The hidden layer nodes sum all the numbers fed to it by the synapses and sends it through a non-linear mapping function.

In this project sigmoid function was used, this function takes any real number and maps it to a number between 0 and 1.The choosing to use 7 neurons for our hidden layer because its medium between the input layer (9 neurons) and the output layer (1 neuron).

Next is another layer of synapses, which connects to our last node, our output neuron. This output neuron can have a value of 0 to 1, and will be used as an output for predicting whether or not our subject is likely fraud. Just like the hidden layer, it maps the sum of the synapses through the sigmoid function.

Above is a basic neural network, but they can become very complex in high level applications. For instance, google's image classification algorithm. Their neural network is what is called a "deep neural network" because it has many hidden layers, and therefore many layers of abstraction necessary for classifying an image. (Think one layer for edge detection, one layer for shape detection, one layer for depth, etc.)

Step 3: Creating the Neural Network Structure in MATLAB

To create the neural network structure in MATLAB, first create two separate sets of data from the original. This step is not necessary to make a functional neural network, but is necessary for testing its accuracy on real world data. Creation of set aside two sets, in which the training set has 90% of the data, and the testing set contains 10%. In doing so, also a creation of two other matrices for each set, one for our input data, and our output data.

Listing 3.1. Neural network planning

```
% create training and testing matrices

   [entries, attributes] = size(Data);

entries_breakpoint = round(entries*.90); %set breakpoint for training and testing data at 90% of
dataset

inputlayersize=9;

outputlayersize=attributes-inputlayersize;

trainingdata = Data(1:entries_breakpoint,:); %truncate first 90% entries for training data

trainingdata_inputs = trainingdata(:,1:inputlayersize); %90%x9 matrix input training data

trainingdata_outputs = trainingdata(:,inputlayersize+1:end); %90:1 matrix output training data

testingdata = Data(entries_breakpoint:end,:); %truncate last 10 entries for testing data

testingdata_inputs= testingdata(:,1:inputlayersize); %10:9 matrix input testing data

testingdata_outputs= testingdata(:,inputlayersize+1:end); %10:1 matrix output testing data
```

in this project the storing of data is saved as matrices as MATLAB’s built-in matrix multiplication functions significantly speed up processing.

Next initializing the two sets of synapses as a matrix of random numbers (for now)

```
%initialize random synapse weights with a mean of 0
hiddenlayersize=7;

    syn0 = 2*rand(inputlayersize,hiddenlayersize) - 1; %random matrix, inputlayersize X
hiddenlayersize

    syn1 = 2*rand(hiddenlayersize,outputlayersize) - 1; %random matrix, hiddenlayersize X
outputlayersize
```

As a preliminary step, feeding data through the network (with random synapse values) and check for accuracy. It is training the network which is the hard part.

Listing 3.2 Feeding data through network

```
%feedforward training data
layer0=trainingdata_inputs;

layer1=(1)./(1+exp(-1.*(layer0*syn0))); %multiply inputs by weights and apply sigmoid activation
function

layer2=(1)./(1+exp(-1.*(layer1*syn1))); %multiply hidden layer by 2nd set of weights and apply
sigmoid activation function

%check for accuracy

err = immse(layer2, trainingdata_outputs);

fprintf("Untrained: Mean Squared Error with Trainingdata: %f\n", err)

%feedforward testing data

layer0=testingdata_inputs;

layer1=(1)./(1+exp(-1.*(layer0*syn0))); %multiply inputs by weights and apply sigmoid activation
functoin

layer2=(1)./(1+exp(-1.*(layer1*syn1))); %multiply hidden layer by 2nd set of weights and apply
sigmoid activation function

%check for accuracy
```

```
err = immse(layer2, testingdata_outputs);

fprintf("Untrained: Mean Squared Error with Testingdata: %f\n", err)
```

these values will be used to compare the accuracy values later to view how effective the neural network training has been.

Step 4: Training the Network.

Training the network requires feeding data through the network, measuring error, and adjusting the synapses in a way that will decrease the error the fastest. Rinse and Repeat.

First, a loop will be created which will repeat a set number of times, constantly re-training the network. Repeat until it either reaches a specific threshold of error or times out. Using a very large value for the for loop for ease of debugging.

```
foriter=[1:1000000]
end
```

Now we may begin writing the training code which will reside within the loop. The first step is to first feed data through the network. We can do this with the same way we did it before.

```
layer0=trainingdata_inputs;
        layer1=(1)./(1+exp(-1.*(layer0*syn0))); %multiply inputs by weights and apply sigmoid
activation functoin

        layer2=(1)./(1+exp(-1.*(layer1*syn1))); %multiply hidden layer by 2nd set of weights and
apply sigmoid activation function
```

The training algorithm works using an already established method called backpropagation. The output of the untrained network is measured against what the output should be. This is called our cost function. Our specific cost function is very simple:

```
%cost function (how much did we miss)
        layer2_error=layer2-trainingdata_outputs;
```

Next, we must do some mathematical to find out which weights will reduce the error the fastest. To do this, calculus will be used, measuring the rate at which the cost function changes with respect to the rate at which each synapse changes. By modify each value of each synapse depending upon how fast it reduces the error (cost function). The synapses with the biggest impact on the error get modified the most, and the synapses with the least impact on the error get modified the least. This process works through all layers of the neural network. In our case, the contribution of error of the first set of synapses to the second set of synapses is calculated. This method of using calculus to determine which weights (synapses) need to be modified the most is called gradient descent.

Listing 3.3 Cost Function

```
%which direction is the target value
        layer2_delta = layer2_error.*(exp(layer2)./(exp(layer2)+1).^2);

    %how much did each l1 value contribute to l2 error

        layer1_error = layer2_delta*syn1.';

    %which direction is target l1

layer1_delta = layer1_error.*(exp(layer1)./(exp(layer1)+1).^2);
```

Next, synapse values are modified using out error that was calculated from above. The variable "alpha" is set to 0.001 in our case because it sets a good rate for training this specific neural network. This value is soft-coded into the program to make debugging easier and is a default value.

Listing 3.4 Adjust Synapses

```
%adjust values
errorval = mean(abs(layer2_error));

        syn1 = syn1 - alpha.*(layer1.'*layer2_delta);

        syn0 = syn0 - alpha.*(layer0.'*layer1_delta);
```

Rinse and Repeat.

setting a diagnosis/debugging code snippet that outputs the current error value to the console with the following code, included within the for loop. Variable "errorval" references the variable created in the above code snippet.

```
%print out debug data
        if iter==1 || mod(iter,100000) == 0

fprintf("\titer=%.0f, Error: %f\n", iter, errorval)

          %syn0

          %syn1

end
```

Step 5: Testing the Trained Output Data.

After training (after the for loop), accuracy of the neural network can be tested with the real-world data that set aside from before.

Listing 3.5 Testing the Trained Output Data

```
%feedforward training data
layer0=trainingdata_inputs;

layer1=(1)./(1+exp(-1.*(layer0*syn0))); %multiply inputs by weights and apply sigmoid activation
function

layer2=(1)./(1+exp(-1.*(layer1*syn1))); %multiply hidden layer by 2nd set of weights and apply
sigmoid activation function

%check for accuracy

err = immse(layer2, trainingdata_outputs);

fprintf("Trained: Mean Squared Error with Trainingdata: %f\n", err)

%feedforward testing data

layer0=testingdata_inputs;

layer1=(1)./(1+exp(-1.*(layer0*syn0))); %multiply inputs by weights and apply sigmoid activation
functoin

layer2=(1)./(1+exp(-1.*(layer1*syn1))); %multiply hidden layer by 2nd set of weights and apply
sigmoid activation function

%check for accuracy

err = immse(layer2, testingdata_outputs);

fprintf("Trained: Mean Squared Error with Testingdata: %f\n", err)
```

the code the for loop used to set up preliminary benchmarks on the accuracy of the neural network.

Running the code, the following information on the console can be gathered.

Listing 3.6Accuracy of the Neural Network

```
Untrained: Mean Squared Error with Trainingdata: 0.267557

Untrained: Mean Squared Error with Testingdata: 0.273381
```

```
Training with alpha: 0.001000

        iter=1, Error: 0.515190

        iter=100000, Error: 0.167916

        iter=200000, Error: 0.130336

        iter=300000, Error: 0.098990

        iter=400000, Error: 0.079489

        iter=500000, Error: 0.068687

        iter=600000, Error: 0.057204

Stopping at: 0.050000 error

Value Below Tolerance found: 0.050000

Trained: Mean Squared Error with Trainingdata: 0.016290

Trained: Mean Squared Error with Testingdata: 0.219258
```

The fully trained neural network significantly minimized the error when fed with the training data. The testing data error was improved, but definitely did not have as drastic an effect as the training data. This is because the neural network was specifically trained to imitate the training data, and did only as good a job as it could've in predicting the testing data.

Final Step:

The included Matlab file is fully functional out of the box, and is included with the CDR data. Using a non-random seed for debugging purposes, as it makes it easier to predict values with which to train the neural network:

```
%set non-random seed
rng('default');

rng(1);
```

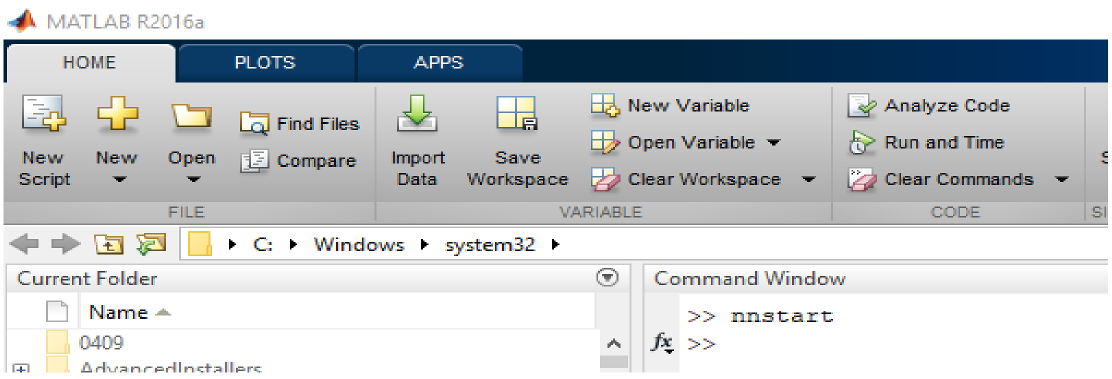
Another thing is the error-tolerance value, which I added as a measure to prevent overtraining of the neural network. (So, the for loop would stop if the error fell below a certain value)

```
error_tolerance = 0.05;

iferrorval<error_tolerance

fprintf("Stopping at: %f error\n", errorval)

break

end
```

3.6 Steps Implementing Artificial Neural Network

3.6.1 Starting with Neural Network

first open the MATLAB software and wait till it is ready, in the command windows write NNSATRT to start GUI of ANN in MATLAB.

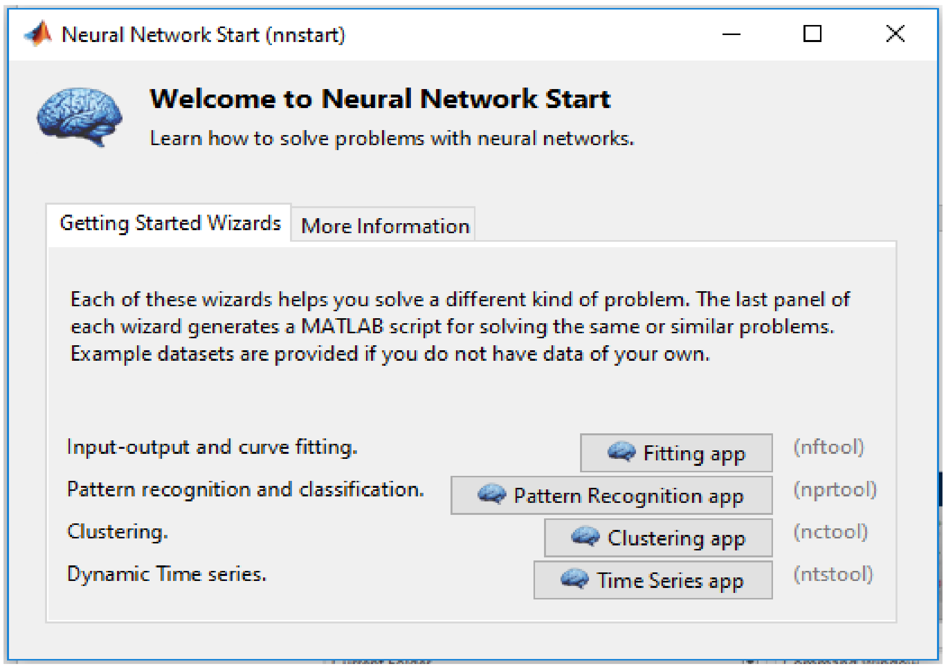


```
>> nnstart
>>
```

Figure 3.3: Starting Artificial Neural Network in MATLAB

3.6.2 Artificial Neural Network Wizard

The following figure is the automatic creation wizard of ANN, >> press Fitting App


Neural Network Start (nnstart)

Welcome to Neural Network Start

Learn how to solve problems with neural networks.

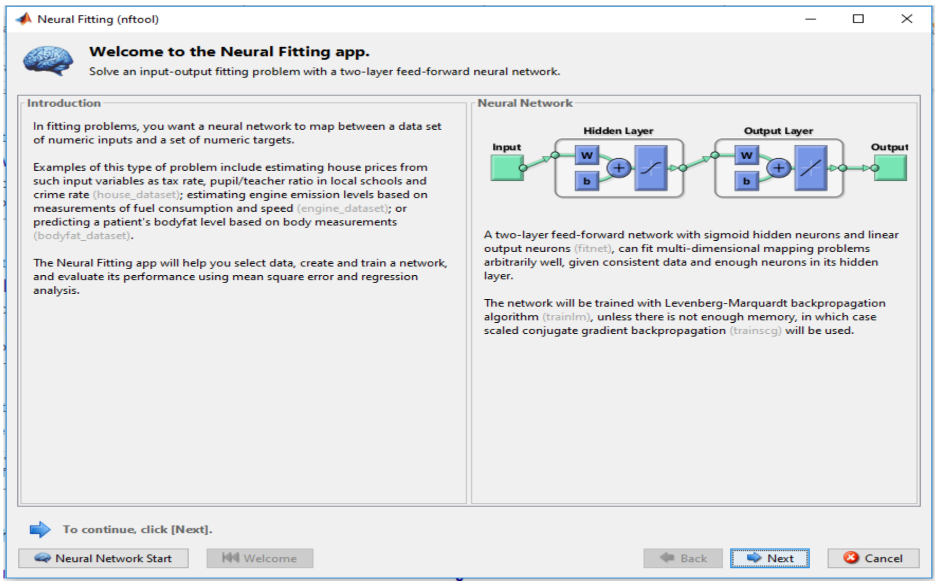
Each of these wizards helps you solve a different kind of problem. The last panel of each wizard generates a MATLAB script for solving the same or similar problems. Example datasets are provided if you do not have data of your own.

Input-output and curve fitting.	Fitting app	(nftool)
Pattern recognition and classification.	Pattern Recognition app	(nprtool)
Clustering.	Clustering app	(nctool)
Dynamic Time series.	Time Series app	(ntstool)

Figure 3.4: Artificial Neural Network Wizards

3.6.3 Fitting App Form

The following form is information about neural network press next


Neural Fitting (nftool)

Welcome to the Neural Fitting app.

Solve an input-output fitting problem with a two-layer feed-forward neural network.

Introduction

In fitting problems, you want a neural network to map between a data set of numeric inputs and a set of numeric targets.

Examples of this type of problem include estimating house prices from such input variables as tax rate, pupil/teacher ratio in local schools and crime rate (house_dataset); estimating engine emission levels based on measurements of fuel consumption and speed (engine_dataset); or predicting a patient's bodyfat level based on body measurements (bodyfat_dataset).

The Neural Fitting app will help you select data, create and train a network, and evaluate its performance using mean square error and regression analysis.

Neural Network

A two-layer feed-forward network with sigmoid hidden neurons and linear output neurons (fitnet), can fit multi-dimensional mapping problems arbitrarily well, given consistent data and enough neurons in its hidden layer.

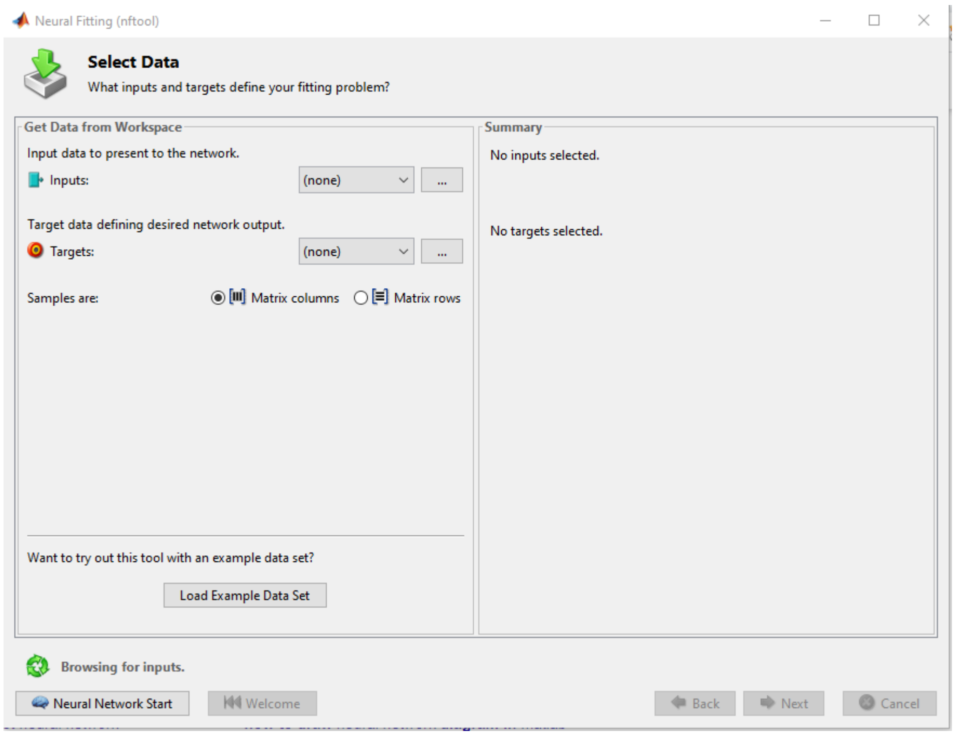
The network will be trained with Levenberg-Marquardt backpropagation algorithm (trainlm), unless there is not enough memory, in which case scaled conjugate gradient backpropagation (trainscg) will be used.

To continue, click [Next].

Figure 3.5: Neural Fitting

3.6.4 Selecting Data

In this step user must enter the dataset by browsing for data set, to load comma separated file that is ready prepared for neural network.


Neural Fitting (nftool)

Select Data

What inputs and targets define your fitting problem?

Get Data from Workspace

Input data to present to the network.

Inputs: (none)

Target data defining desired network output.

Targets: (none)

Samples are: [✓ Matrix columns] Matrix rows

Want to try out this tool with an example data set?

Load Example Data Set

Summary

No inputs selected.

No targets selected.

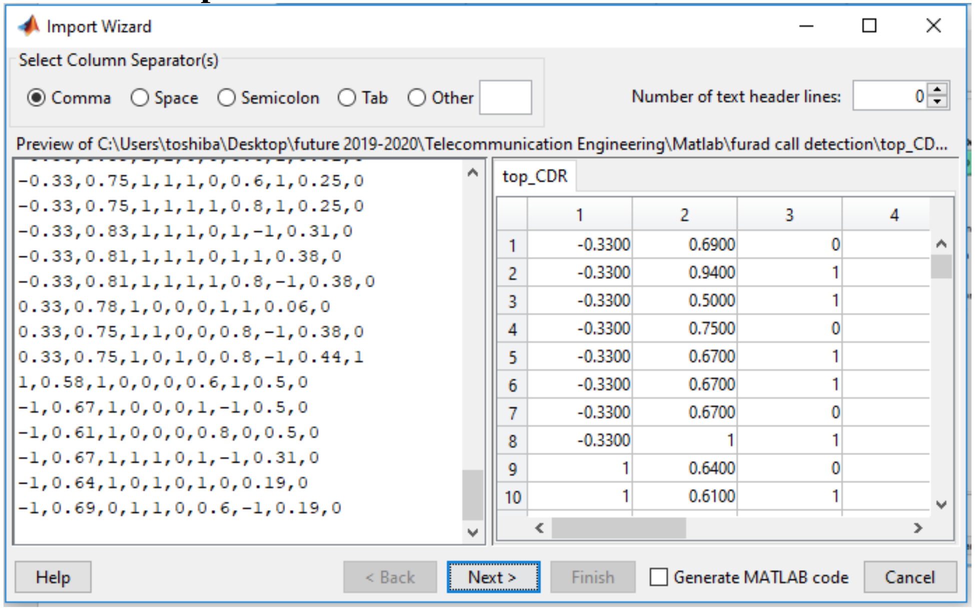
Browsing for inputs.

Neural Network Start | Welcome | Back | Next | Cancel

Figure 3.6Selecting Data Inputs and Targets-

3.6.5 Loading Comma File

In this step the comma file is loaded into matlab workspace in order to use it as input to the ANN


Import Wizard

Select Column Separator(s)

[✓ Comma] Space Semicolon Tab Other

Number of text header lines: 0

Preview of C:\Users\toshiba\Desktop\future 2019-2020\Telecommunication Engineering\Matlab\furad call detection\top_CD...

```
-0.33,0.75,1,1,1,0,0.6,1,0.25,0
-0.33,0.75,1,1,1,1,0.8,1,0.25,0
-0.33,0.83,1,1,1,0,1,-1,0.31,0
-0.33,0.81,1,1,1,0,1,1,0.38,0
-0.33,0.81,1,1,1,1,0.8,-1,0.38,0
0.33,0.78,1,0,0,0,1,1,0.06,0
0.33,0.75,1,1,0,0,0.8,-1,0.38,0
0.33,0.75,1,0,1,0,0.8,-1,0.44,1
1,0.58,1,0,0,0,0.6,1,0.5,0
-1,0.67,1,0,0,0,1,-1,0.5,0
-1,0.61,1,0,0,0,0.8,0,0.5,0
-1,0.67,1,1,1,0,1,-1,0.31,0
-1,0.64,1,0,1,0,1,0,0.19,0
-1,0.69,0,1,1,0,0.6,-1,0.19,0
```

top_CDR

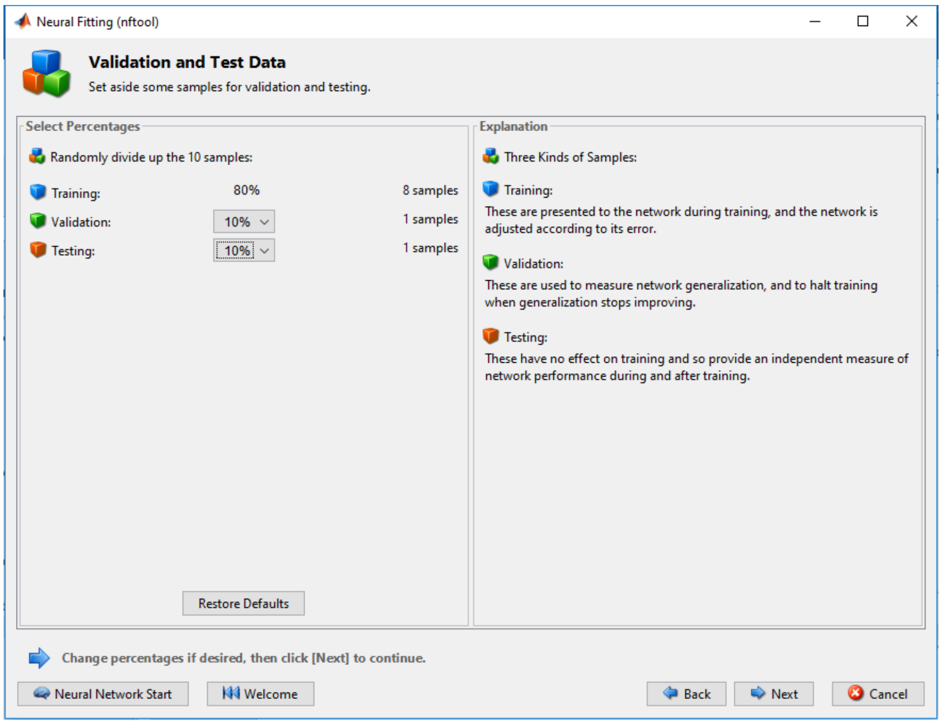
	1	2	3	4
1	-0.3300	0.6900	0	
2	-0.3300	0.9400	1	
3	-0.3300	0.5000	1	
4	-0.3300	0.7500	0	
5	-0.3300	0.6700	1	
6	-0.3300	0.6700	1	
7	-0.3300	0.6700	0	
8	-0.3300	1	1	
9	1	0.6400	0	
10	1	0.6100	1	

Help | < Back | Next > | Finish | Generate MATLAB code | Cancel

Figure 3.7: Importing Data to MATLABWorkspace

Selecting Validation and Percentage from the Records

In this step the user must select the percentage of validation records and testing records



Neural Fitting (nftool)

Validation and Test Data

Set aside some samples for validation and testing.

Select Percentages

Randomly divide up the 10 samples:

Training:	80%	8 samples
Validation:	10%	1 samples
Testing:	10%	1 samples

Restore Defaults

Explanation

Three Kinds of Samples:

Training:
These are presented to the network during training, and the network is adjusted according to its error.

Validation:
These are used to measure network generalization, and to halt training when generalization stops improving.

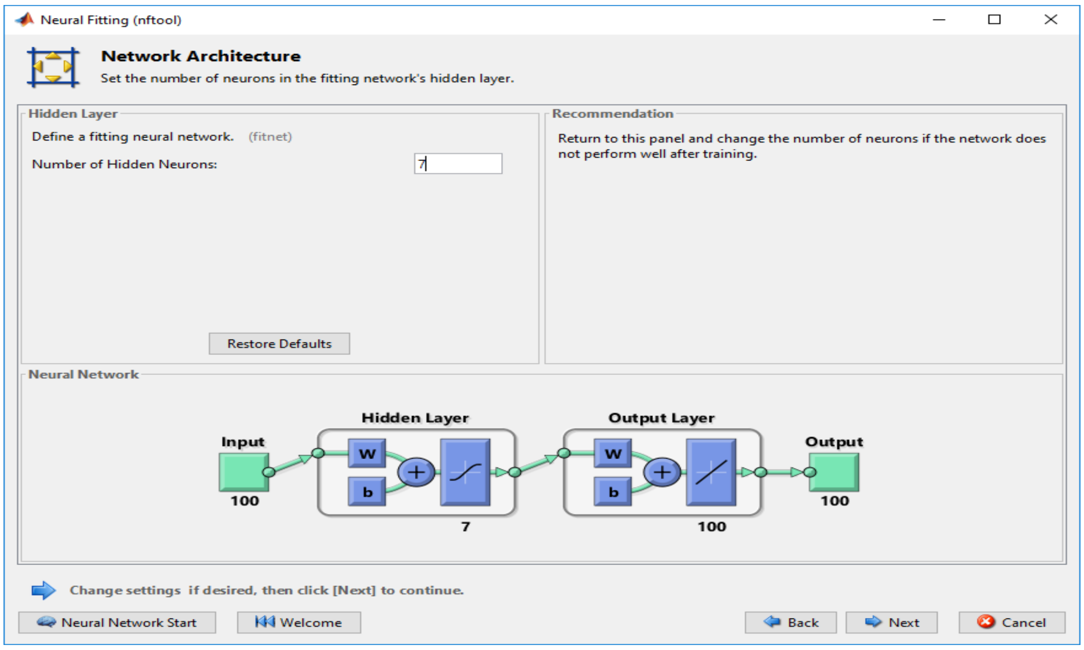
Testing:
These have no effect on training and so provide an independent measure of network performance during and after training.

Change percentages if desired, then click [Next] to continue.

Neural Network Start | Welcome | Back | Next | Cancel

Figure 3.8: Selecting Validation and Testing Samples

3.4.5 Setting the Hidden Layers



Neural Fitting (nftool)

Network Architecture

Set the number of neurons in the fitting network's hidden layer.

Hidden Layer

Define a fitting neural network. (fitnet)

Number of Hidden Neurons: 7

Restore Defaults

Recommendation

Return to this panel and change the number of neurons if the network does not perform well after training.

Neural Network

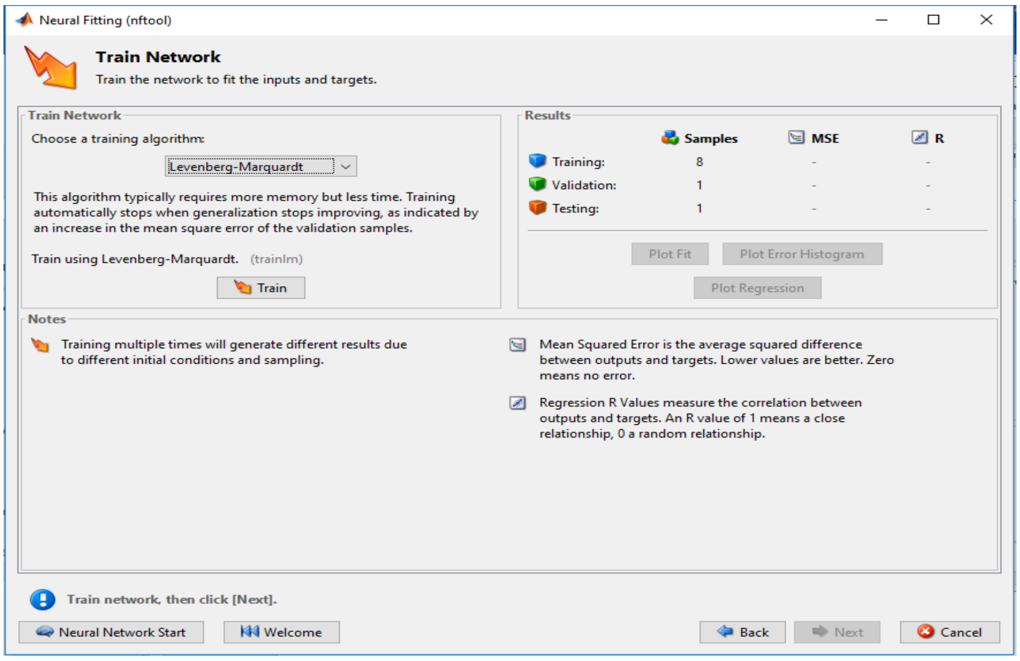
Change settings if desired, then click [Next] to continue.

Neural Network Start | Welcome | Back | Next | Cancel

Figure 3.9: Set Number of Hidden Layers

3.5 Network Training

In this step a network training is done so the network can validate the performance



Train Network
Train the network to fit the inputs and targets.

Train Network
Choose a training algorithm: Levenberg-Marquardt

This algorithm typically requires more memory but less time. Training automatically stops when generalization stops improving, as indicated by an increase in the mean square error of the validation samples.

Train using Levenberg-Marquardt. (trainlm)

Results

	Samples	MSE	R
Training:	8	-	-
Validation:	1	-	-
Testing:	1	-	-

Notes

Training multiple times will generate different results due to different initial conditions and sampling.

Mean Squared Error is the average squared difference between outputs and targets. Lower values are better. Zero means no error.

Regression R Values measure the correlation between outputs and targets. An R value of 1 means a close relationship, 0 a random relationship.

Train network, then click [Next].

Figure 3.10: Training the Neural Network

3.7 Measures for Performance:

A number of different measures are commonly used to evaluate the performance of the proposed method. These measures including classification, sensitivity, specificity, Mathew’s correlation coefficient (MCC) calculated from confusion matrix.

True Positive (TP) – counts of all samples which are correctly called by the algorithm as normal call.

False Positive (FP) –counts of all samples which are incorrectly called by the algorithm as fraud call while they are normal.

True Negative (TN) – counts of all samples which are correctly called by the algorithm as fraud call.

False Negative (FN) – count of all samples which are incorrectly called by the algorithm as normal call while they are fraud call.

The performance of the classification algorithms was evaluated by computing the percentages of Sensitivity (SE), Specificity (SP), Accuracy (AC) and Mathews Correlation Coefficient (MCC), The respective definitions are as follows:

SE= TP/ (TP+FN)*100 (1)

SP= TN/ (TN+TP)*100 (2)

AC= (TP+TN)/ (TN+TP+FN+FP)*100 (3)

MCC= (TP×TN-FP×FN)/ √ (TP+FP) (TP+FN) (TN+FP) (TN+FN) (4)

Artificial Neural Network is an artificial representation of the human brain that tries to simulate its learning process. To train a network and measure how well it performs, an objective function must be defined. A commonly used performance criterion function is the sum of squares error function.

$$E = \frac{1}{2} \sum_{p=1}^{P} \sum_{i=1}^{N} \left(t_{pi} - y_{pi} \right)^2 \qquad (5)$$

Where, p represents the patterns in the training set, yp is the output vector (based on the hidden layer output), tp is the training target. The above equation represents the output nodes, tpi and ypi are, respectively, the target and actual network output for the ith output unit on the pth pattern. The network learns the problem at hand by adjusting weights. The process of adjusting the weights to make the Neural Network learn the relationship between the inputs and the targets is known as learning or training. There are several methods of finding the weights of which the gradient descent method is most common[19].

Chapter Four
Result and Discussion

Chapter Four
Results and Discussion

In this chapter the results and discussion were included along with the analysis to the accuracy of detection of the program.

Three basic criteria are considered to evaluate the results in form of Performance, Mean Square Error (MSE) and Regression which are Testing, Training and Validation.

4.1 Artificial Neural Network Basic Configuration

Table4.1 Parameter Setting Table

Parameters	Options

Input Layer Size	9 Neurons
Hidden Layer Size	7 Neurons
Output Layer Size	1 Neurons
Activation Function	Back propagation Function
Neural Network Type	Feed-forward Neural Network

4.2Running the Neural Network Training

After setting the network training screen view the performance, time and regression, it was found that Mean Square Error and regression have been reduced, according to increasing the testing and validation samples.

Table 4.2 case 1: Training, Testing and Validation with associated Mean Square Error and Regression Value

Feature	(10 Samples)	Percentage	Mean Square Error	Regression
Training	8	90%	4.01716e-1	4.05901e-1
Validation	1	5%	5.33483e-1	1.80909e-1
Testing	1	5%	2.82459e-0	1.39926e-2

Table 4.3 case 2: Training, Testing and Validation with associated Mean Square Error and Regression Value

Feature	(10 Samples)	Percentage	Mean Square Error	Regression
Training	6	70%	8.25951e-1	1.93857e-1
Validation	2	15%	1.09469e-0	5.95325e-2
Testing	2	15%	6.42803e-1	-1.57506e-1

Table 4.4 case 3: Training, Testing and Validation with associated Mean Square Error and Regression Value

Feature	(10 Samples)	Percentage	Mean Square Error	Regression
Training	4	50	1.70276e-1	5.92323e-1
Validation	3	25%	1.2647e-0	1.48749e-1
Testing	3	25%	1.66778e-0	1.00170e-1

Tables 4.2, 4.3 and 4.4 illustrate how the mean square error and regression changed and inversely proportion with the training samples, And direct proportion with Testing and Validation.

4.3Best Validation of Training

It was found that the error reduced at the 3 epochs

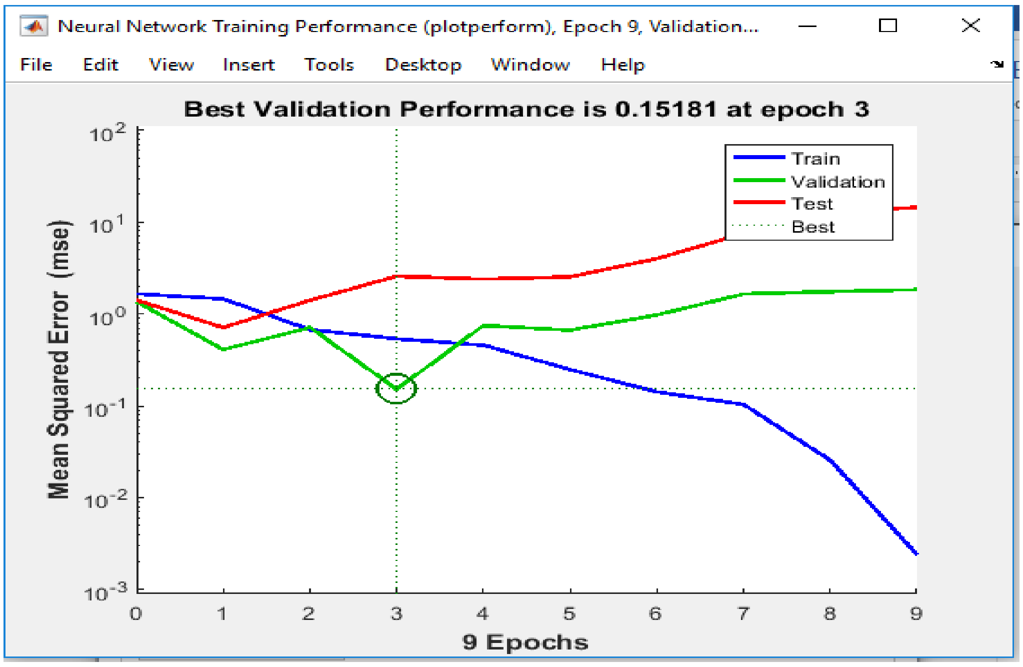


Figure 4.1: Detecting Mean Square Error

Figure 4.2 illustrates that the best validation Performance foundat the epoch 3.

It was found that the mean square error is minimized by maximizing the percentage of testing data, as well as increasing training data leads to minimizing the validation and the mean square error would be minimized.

4.4 Validation

It was found that the epoch at 3 is the best validation area with a lower error

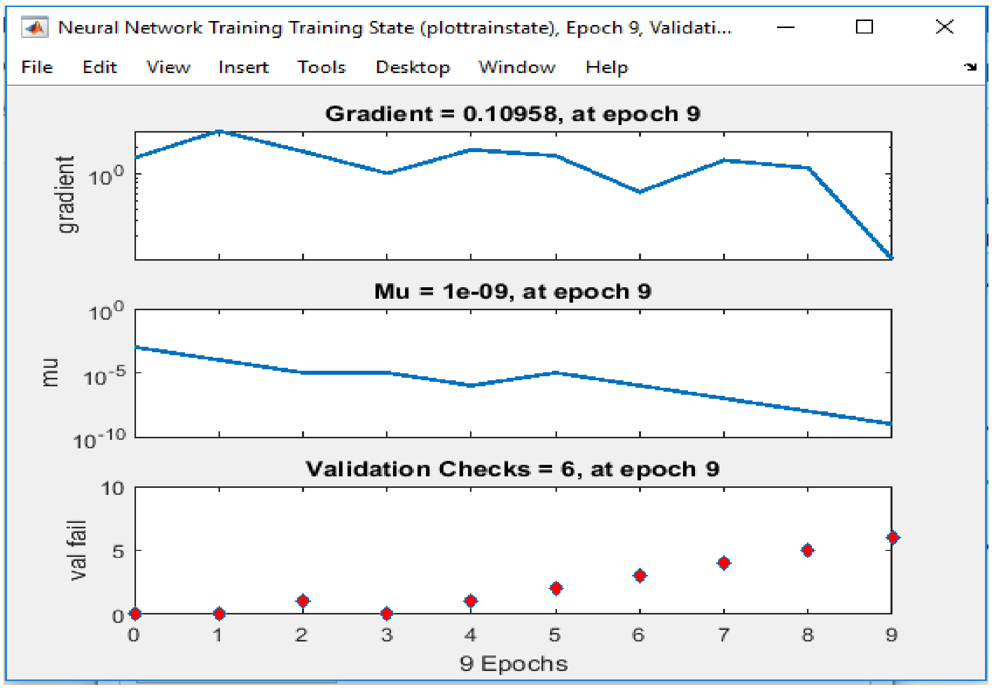


Figure 4.2: Results of 9 Epochs

Figure 4.2 illustrate how the learning rate (mu)gradient changes during time and how it affects the validation value. And it was found that the best validation was at instant 9 which learning rate took its lowest value.

4.5 Histogram of Error

The following graph represents the error histogram of the neural network and also it is increases after 3 epoch.

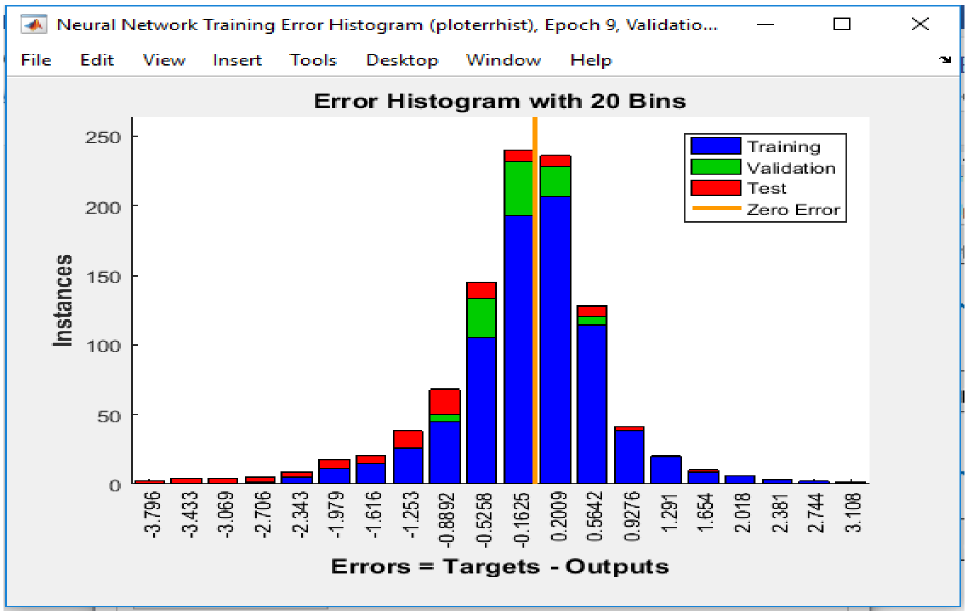


Figure 4.3: Error Histogram

Figure 4.3 shows the correlation betweenthe target output and the calculated output for training data and testing data. And it is found that zero error has been obtained when target and result output are becoming so close to each other, which means the regression value would become very high.

4.6 Iterations and Error Reduction

It is found that after each iteration the error are reduced and the best is 0.10282

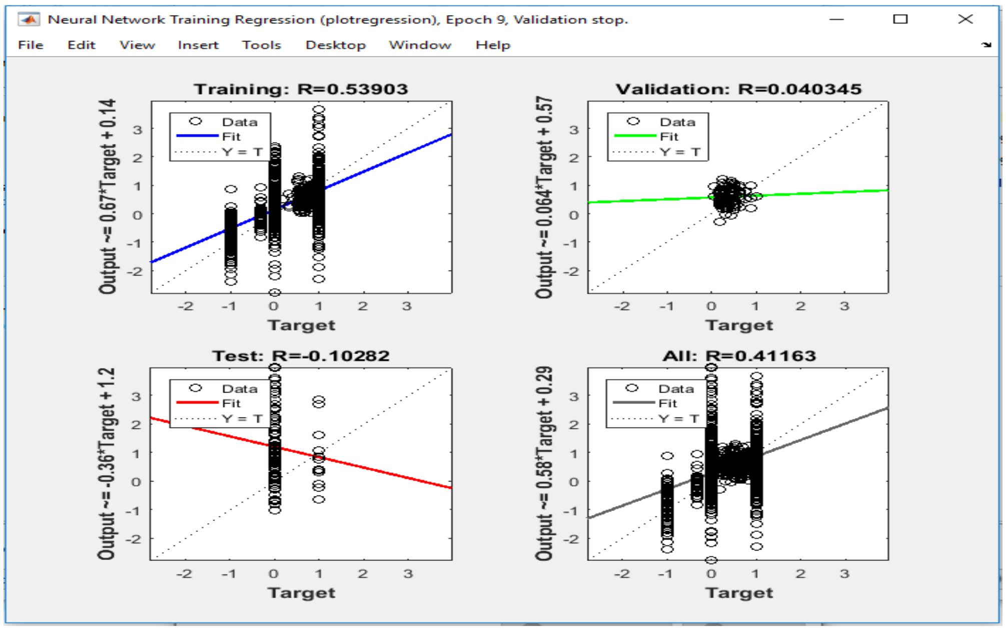


Figure 4.4: Detection of Error While Iterations

Figure 4.4 illustrates the regression value calculated for training data, testing data, validation data and all data, it is noted that whenever the difference between the result output and target output is maximized in term of error value that means the relation ' regression 'between them is so far, in contrast whenever the difference between the result output and target output is minimizedthe relation between the target and result is so close which refer to the regression.

4.7 Performance Matrices

Table 4.5: System Validation Test

Measure	**Value**	**Derivations**
Sensitivity	0.9435	TPR = TP / (TP + FN)
Specificity	0.7000	SPC = TN / (FP + TN)
Accuracy	0.9333	ACC = (TP + TN) / (P + N)
Matthews Correlation Coefficient	0.4652	TP*TN - FP*FN / sqrt((TP+FP)*(TP+FN) *(TN+FP) *(TN+FN))

this system capable of determining the true positive value, the true negative value, the false positive value, and the false negative value.

The true positive value which refers to the number of normal call and classified correctly as normal call by neural network, the true negative which refers to the number of normal call and classified in correctly as fraud call by neural network, the true negative which refers to the number of fraud call and classified correctly as fraud call by neural network, false negative which refers to the number of fraud call and classified in correctly as normal call by neural network.

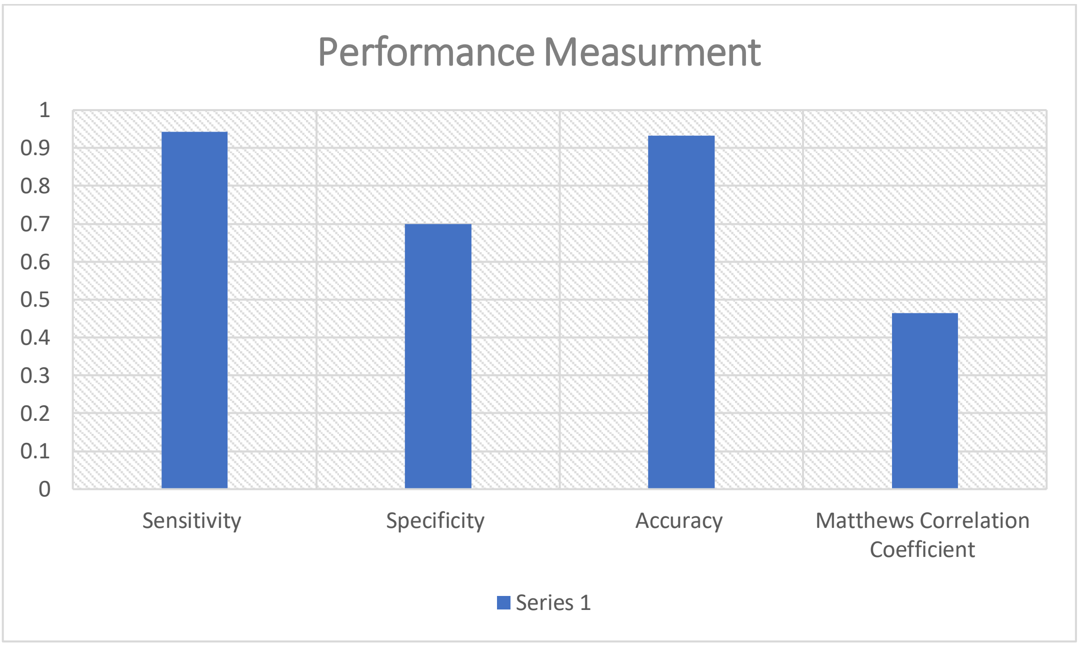


Figure 4.5: Performance Measurement

Figure 4.5 shows the main factors that play more important role to enhance neural network performance as follow:

Sensitivity it measures how the neural network system is able to identify the new input data according to the defined features for fraud call.

Specificity it measures how the neural network system is able to determine whether the input data record was normal or fraud call record.

Accuracy it measures the account of predictions where the predict value is equal to the true value.

Neural network system is able to determine whether the input data record was normal or fraud call record.

The Matthews Correlation Coefficient (MCC) is one of the popular measurements for classification accuracy. It has been generally regarded as a balanced measure which can be used even if the classes are of very different sizes. MCC deteriorates seriously when the dataset in classification are imbalanced. Finally,the obtained results were led to increased system accuracy to detect fraud call.

Chapter Five

Conclusion and Recommendations

Chapter Five

Conclusion and Recommendations

5.1 Conclusion

In this project, an analyzing offline call data record has been done in order to classify and detect telecommunication fraud call, by analyzing the content of the call data record Instead of relying onthe source caller number and call type, and then constructing a blacklist of fraud numbers. The project solves this problem, first a detection of telecommunication frauds through analyzing and extracting features from the contents of a call. Particularly, descriptions and features ofcall data record had been collected, rules had been built and applied on testing data then training data then error value has been calculated through obtaining the difference between the target output and the result output. In this project feed-forward artificial neural network had been used to analyze data and to high-quality descriptions had selected from the data collected previously to construct datasets. Then the performance metrics had been measured such as False Positive, true positive, false negative and true negative. Finally, sensitivity, specificity and accuracy had beendetermined.

It was found that the mean square error and regression changed and inversely proportion with the training samples, And direct proportion with Testing and Validation, the mean square error is minimized by maximizing the percentage of testing data, as well as increasing training data leads to minimizing the validation and the mean square error would be minimized, the learning rate (mu) gradient changes during time and how it affects the validation value.

And it was found that the best validation was at instant 9 which learning rate took its lowest value, shows the correlation between the target output and the calculated output for training data and testing data. And it is found that zero error has been obtained when target and result output are becoming so close to each other, which means the regression value would become very high.

It was found that the correlation between the target output and the calculated output for training data and testing data. And it is found that zero error has been obtained when target and result output are becoming so close to each other, which means the regression value would become very high.

Figure 4.3 shows the correlation between the target output and the calculated output for training data and testing data. And it is found that zero error has been obtained when target and result output are becoming so close to each other, which means the regression value would become very high.

5.2 Recommendations

After the research was completed, it requires more development such as applying the automatic updates to the system for increasing the system accuracy. And more use of classification techniques and detection methods such as support vector machine, fuzzy logic and genetic algorithms and to develop a tool to secure the remote connection.

References

1- Chouiekh,Alae, and EL HassaneIbn EL Haj, (2018)Convnetsfor fraud detection analysis. Procedia Computer Science 127: 133-138.

2- Zhao, Qianqian, et al, (2018)Detecting telecommunication fraud by understanding the contents of a call. Cybersecurity 1.1 : 1-12.

3- REBAHI, Yacine, et al, (2011) A survey on fraud and service misuse in voice over IP (VoIP) networks. Information Security Technical Report, 16.1: 12-19.

4- Azad, Muhammad Ajmal, and Ricardo Morla, (2013)Caller-rep: Detecting unwanted calls with caller social strength. Computers & Security 39: 219-236.

5- Chen, Kuang-Hua, and Hsin-Hsi Chen, (2001) Cross-language Chinese text retrieval in NTCIR workshop: towards cross-language multilingual text retrieval.ACM SIGIR Forum. Vol. 35. No. 2. New York, NY, USA: ACM,.

6- Delamaire, Linda, Hussein Abdou, and John Pointon, (2009)Credit card fraud and detection techniques: a review.Banks and Bank systems 4.2 57-68.

7- Geng, Yingchao, (2017) Research on How to Deal with the Dilemma of Global Cooperative Governance of Cross-Border Telecom Network Fraud in China. Chinese Studies 6.04 249.

8- Gao, Jianfeng, Mu Li, and Chang-Ning Huang, (2003) Improved source-channel models for Chinese word segmentation.

9- Jackson, Peter, and Isabelle Moulinier, 2002Natural language processing for online applications. Philadelphia: John Benjamins.

10- Jiang, Nan, et al, (2012) Isolating and analyzing fraud activities in a large cellular network via voice call graph analysis. Proceedings of the 10th international conference on Mobile systems, applications, and services.

11- Rana, Priya J., and JwalantBaria, (2015) A survey on fraud detection techniques in ecommerce. International Journal of Computer Applications 113.14.

12- Jiang, Nan, et al. (2013) Greystar: Fast and Accurate Detection of {SMS} Spam Numbers in Large Cellular Networks Using Gray Phone Space.22nd USENIX Security Symposium (USENIX Security 13).

13- Kolan, Prakash, Ram Dantu, and Joao W. Cangussu. (2008) Nuisance level of a voice call. ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)5.1 : 1-22.

14- Fan, Yuchen, et al. (2014) TTS synthesis with bidirectional LSTM based recurrent neural networks.Fifteenth annual conference of the international speech communication association.

15- Zen, Heiga, and HaşimSak. (2015) Unidirectional long short-term memory recurrent neural network with recurrent output layer for low-latency speech synthesis. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE.

16- Fan, Bo, et al. (2015) Photo-real talking head with deep bidirectional LSTM. 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE.

17- Silver, David, et al, (2017)Mastering chess and shogi by self-play with a general reinforcement learning algorithm.arXiv preprint arXiv:1712.01815.

18- Goodfellow, Ian, et al. (2014) Generative adversarial nets. Advances in neural information processing systems 27.

19- Sun, Jian. SOSPCNN: Structurally Optimized Stochastic Pooling Convolutional."

20- Sazli, M. H. (2006). A brief review of feed-forward neural networks. *Communications Faculty of Sciences University of Ankara Series A2-A3 Physical Sciences and Engineering*, *50*(01).

21- Auda, Gasser, and Mohamed Kamel. "Modular neural networks: a survey." *International journal of neural systems* 9.02 (1999): 129-151.

22- Elmi, A. H., Ibrahim, S., &Sallehuddin, R. (2013). Detecting sim box fraud using neural network. In *IT Convergence and Security 2012* (pp. 575-582). Springer, Dordrecht.

23- O'Shea, K., & Nash, R. (2015). An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*.

24- Phua, Clifton, et al. (2010) A comprehensive survey of data mining-based fraud detection research. *arXiv preprint arXiv:1009.6119*.

25- Estévez, Pablo A., Claudio M. Held, and Claudio A. Perez. "Subscription fraud prevention in telecommunications using fuzzy rules and neural networks, (2006) *Expert Systems with Applications* 31.2 :337-344.

Appendix

```
%{

USING CDR DATASET

Call duration: 1) less than min, 2) between 1 to 3, 3) between 3 to 9, 4) fall. (-1, -0.33, 0.33, 1)

Call repeating. 18 time or equal or greater than 36 (0, 1)

Call source is the same:  1) yes, 2) no. (0, 1)

Single number to multi number international 1) yes, 2) no. (0, 1)

Reachable number 1) yes, 2) no. (0, 1)

Time during the day 1) morning, 2) med-day, 3) end of the day. (-1, 0, 1)

Output: Diagnosis normal (N-->0), Fraud (O-->1)

%}

clc

clear

%set non-random seed
```

```
rng('default');

rng(1);

% input data

filename = 'data.txt';

delimiterIn = ',';

   Data = importdata(filename,delimiterIn);

% create training and testing matrices

   [entries, attributes] = size(Data);

entries_breakpoint = round(entries*.90); %set breakpoint for training and testing
data at 90% of dataset

inputlayersize=9;

outputlayersize=attributes-inputlayersize;

trainingdata = Data(1:entries_breakpoint,:); %truncate first 90% entries for
training data

trainingdata_inputs = trainingdata(:,1:inputlayersize); %90%x9 matrix input
training data

trainingdata_outputs = trainingdata(:,inputlayersize+1:end); %90:1 matrix output
training data

testingdata = Data(entries_breakpoint:end,:); %truncate last 10 entries for testing
data

testingdata_inputs= testingdata(:,1:inputlayersize); %10:9 matrix input testing
data

testingdata_outputs= testingdata(:,inputlayersize+1:end); %10:1 matrix output
testing data

error_tolerance = 0.05;

hiddenlayersize=7;
```

```
%initialize random synapse weights with a mean of 0
   syn0 = 2*rand(inputlayersize,hiddenlayersize) - 1; %random matrix,
inputlayersize X hiddenlayersize
   syn1 = 2*rand(hiddenlayersize,outputlayersize) - 1; %random matrix,
hiddenlayersize X outputlayersize
%feedforward training data
layer0=trainingdata_inputs;
layer1=(1)./(1+exp(-1.*(layer0*syn0))); %multiply inputs by weights and apply
sigmoid activation functoin
layer2=(1)./(1+exp(-1.*(layer1*syn1))); %multiply hidden layer by 2nd set of
weights and apply sigmoid activation function
%check for accuracy
err = immse(layer2, trainingdata_outputs);
fprintf('Untrained: Mean Squared Error with Trainingdata: %f\n', err)
%feedforward testing data
layer0=testingdata_inputs;
layer1=(1)./(1+exp(-1.*(layer0*syn0))); %multiply inputs by weights and apply
sigmoid activation functoin
layer2=(1)./(1+exp(-1.*(layer1*syn1))); %multiply hidden layer by 2nd set of
weights and apply sigmoid activation function
%check for accuracy
err = immse(layer2, testingdata_outputs);
fprintf('Untrained: Mean Squared Error with Testingdata: %f\n', err)
%best alpha for fertilitydata = 0.001
for alpha=[0.001]
fprintf('Training with alpha: %f\n', alpha)
```

```
foriter=1:1000000

    %feed-forward

        layer0=trainingdata_inputs;

        layer1=(1)./(1+exp(-1.*(layer0*syn0))); %multiply inputs by weights and
apply sigmoid activation functoin

        layer2=(1)./(1+exp(-1.*(layer1*syn1))); %multiply hidden layer by 2nd set
of weights and apply sigmoid activation function


    %cost function (how much did we miss)

        layer2_error=layer2-trainingdata_outputs;


    %which direction is the target value

        layer2_delta = layer2_error.*(exp(layer2)./(exp(layer2)+1).^2);

    %how much did each l1 value contribute to l2 error

        layer1_error = layer2_delta*syn1.';

    %which direction is target l1

        layer1_delta = layer1_error.*(exp(layer1)./(exp(layer1)+1).^2);

%adjust values

errorval = mean(abs(layer2_error));

        syn1 = syn1 - alpha.*(layer1.'*layer2_delta);

        syn0 = syn0 - alpha.*(layer0.'*layer1_delta);


iferrorval<error_tolerance

fprintf('Stopping at: %f error\n', errorval)

break
```

```
end

        %print out debug data

ifiter==1 || mod(iter,100000) == 0

fprintf('\titer=%.0f, Error: %f\n', iter, errorval)

                %syn0

                %syn1

end

end

iferrorval>error_tolerance

fprintf('Value Below Tolerance not found, please adjust alpha\n\n')

else

fprintf('Value Below Tolerance found: %f\n\n', errorval)

end

end

%feedforward training data

layer0=trainingdata_inputs;

layer1=(1)./(1+exp(-1.*(layer0*syn0))); %multiply inputs by weights and apply
sigmoid activation functoin

layer2=(1)./(1+exp(-1.*(layer1*syn1))); %multiply hidden layer by 2nd set of
weights and apply sigmoid activation function

%check for accuracy

err1 = immse(layer2, trainingdata_outputs);

fprintf('Trained: Mean Squared Error with Trainingdata: %f\n', err1)

%feedforward testing data

layer0=testingdata_inputs;
```

```
layer1=(1)./(1+exp(-1.*(layer0*syn0))); %multiply inputs by weights and apply
sigmoid activation functoin

layer2=(1)./(1+exp(-1.*(layer1*syn1))); %multiply hidden layer by 2nd set of
weights and apply sigmoid activation function
```

%check for accuracy

```
err2 = immse(layer2, testingdata_outputs);

fprintf('Trained: Mean Squared Error with Testingdata: %f\n', err2)

%[layer2.'; testingdata_outputs.'].'

gold_data=randint(400,400,2,200)

test_data=randint(400,400,2,1.5)


% sigmoid_std=(1)./(1+exp(-1*input));

% sigmoid_deriv=(exp(input)./(exp(input)+1).^2);

% sigmoid_deriv=(sigmoid_std)(1-sigmoid_std)

TP=0;FP=0;TN=0;FN=0;

fori=1:400;

for j=1:400;

if(gold_data(i,j)==1 &test_data(i,j)==1);

        TP=TP+1;

elseif(gold_data(i,j)==0 &test_data(i,j)==1);

        FP=FP+1;

elseif(gold_data(i,j)==0 &test_data(i,j)==0);

        TN=TN+1;

else

        FN=FN+1;
```

```
end

end

end
```