A VOTING TECHNIQUE OF MULTILAYER PERCEPTRON ENSEMBLE FOR CLASSIFICATION APPLICATION

HAFIZAH BINTI TALIB

UNIVERSITI SAINS MALAYSIA

2014

A VOTING TECHNIQUE OF MULTILAYER PERCEPTRON ENSEMBLE FOR CLASSIFICATION APPLICATION

by

HAFIZAH BINTI TALIB

Thesis submitted in fulfillment of requirements

for the degree of

Master of Science

February 2014

ACKNOWLEDGEMENTS

In the name of Allah, the Most Gracious and the Most Merciful

Alhamdulillah, all praises to Allah for all the strengths and His blessing in completing this thesis. I would like to express my deep gratitude to Associate Prof. Dr Junita Mohamad Saleh, my research supervisors, for her patient guidance, enthusiastic encouragement, useful critiques of this research work and her time checking this thesis. I would also like to thank Associate Prof. Dr Zalina Abdul Aziz for her advice and assistance.

I would also like to extend my thanks to my fellow friends, Khursiah Zainal Mokhtar and Hosni Anis Kamaruddin for their ideas and help. Assistance provided by them were greatly appreciated.

To Encik Md Isa bin Ibrahim, Head of Department, National Youth Skills Institute of Bukit Mertajam, his encouragement and understanding was very much appreciated. To my colleagues, Nor Rumaizah Azmi, Norhayati Abdul Rasib and Hashim Ahmad, their continuous support gives me strength to complete this thesis.

I wish to thank my parents, Encik Talib bin Ali and Puan Rofisah Mohd Arif for their love, inspiration and encouragement throughout my study.

I would also like to acknowledge the Fundamental Research Grant Scheme No. 203/6071165 from the Malaysian Ministry of Higher Education for the financial support for this work.

TABLE OF CONTENTS

CHAPTER 4: APPLYING THE DEVELOPED MLPE TO CLASSIFICATION TASK

CHAPTER 5: CONCLUSION AND FUTURE WORK

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ANN	Artificial neural network
MLP	Multilayer perceptron
MLPE	Multilayer perceptron ensemble
PE	Processing element
MV	Majority voting
TV	Trust voting
TSV	Trust-sum voting
CCP	Correct classification percentage
LM	Levenberq Marquardt
RP	Resilient backpropagation
BR	Bayesian regularization
ECT	Electrical capacitance tomography
LSI	Landsat satellite image
GC	German credit
PID	Pima Indian diabetes
MLPE-MV	Multilayer perceptron ensemble with majority voting
MLPE-TV	Multilayer perceptron ensemble with trust voting
MLPE-TSV	Multilayer perceptron with trust-sum voting

TEKNIK UNDIAN GABUNGAN PERSEPTRON BERBILANG LAPISAN UNTUK APLIKASI PENGELASAN

ABSTRAK

Rangkaian perceptron berbilang lapisan (MLP) adalah model rangkaian neural buatan yang ringkas tetapi telah berjaya dalam pelbagai aplikasi. Namun, prestasi MLP yang tidak stabil di mana perubahan kecil dalam parameter latihan boleh menghasilkan model yang berlainan telah menjadi penghalang kepada kejituan tinggi untuk aplikasi pengelasan. Dalam kajian ini, satu sistem bersepadu gabungan MLP (MLPE) yang terdiri daripada MLPE dan algoritma undian baru telah dibina bertujuan meningkatkan kejituan pengelasan dan mengurangkan bilangan kes kelas rosak (*reject*). MLPE dihasilkan daripada MLP tunggal yang berlainan daripada segi algoritma latihan dan berat awalan. Tiga algoritma latihan yang digunakan ialah Levenberg-Marquardt (LM), Kebingkasan Perambatan Belakang (RP) dan Pengaturan Bayes (BR). Bagi memilih keluaran terakhir MLPE, teknik undian baru yang dinamakan teknik Undian Keyakinan-Jumlah (TSV) telah dicadangkan. Keberkesanan sistem bersepadu MLPE menggunakan TSV (MLPE-TSV) telah diuji ke atas empat kajian kes pengelasan iaitu Tomografi Kemuatan Elektrik (ECT), Imej Satelit Landsat (LSI), Kredit German (GC) dan kes diabetes dikalangan Pima Indian (PID). Prestasi MLPE-TSV dibandingkan dengan MLPE menggunakan teknik undian sedia ada iaitu teknik undian majoriti (MLPE-MV) dan teknik udian keyakinan (MLPE-TV). Keputusan yang diperolehi menunjukkan bahawa MLPE-TSV yang dicadangkan telah berupaya meningkatkan kejituan pengelasan berbanding dengan prestasi MLP tunggal, MLPE-MV dan MLPE-TV. MLPE-TSV juga telah berjaya mengurangkan bilangan kes kelas rosak.

A VOTING TECHNIQUE OF MULTILAYER PERCEPTRON ENSEMBLE FOR CLASSIFICATION APPLICATION

ABSTRACT

MLP is a model of artificial neural network, which is simple yet successfully applied in various applications. The instability of MLP performance where small changes in training parameter could produce different models that inhibiting attainment of high accuracy in classification applications. In this research, an integrated system of Multi-Layer Perceptron Ensemble (MLPE) consisting of an MLPE and a new voting algorithm has been developed to increase classification accuracy and reduce the number of reject class cases. MLPE is produced from singular MLPs that are diverse in term of training algorithm and their initial weights. Three training algorithms used are Levenberg-Marquardt (LM), Resilient Backpropagation (RP) and Bayesian Regularization (BR). In order to choose the final output of MLPE, a new voting algorithm named Trust-Sum Voting (TSV) is proposed. The effectiveness of MLPE with TSV (MLPE-TSV) has been tested on four classification case studies which are Electrical Capacitance Tomography (ECT), Landsat Satellite Image (LSI), German Credit (GC) and Pima Indian Diabetes (PID). The performance of MLPE-TSV has been compared with the performance of MLPE which employs existing voting algorithms which are Majority Voting (MLPE-MV) and Trust Voting (MLPE-TV). The obtained results have shown that the proposed MLPE-TSV is capable of increasing the accuracy of classification as compared to singular MLPs, MLPE-MV and MLPE-TV. MLPE-TSV has also managed to reduce the number of cases in reject class.

CHAPTER 1

INTRODUCTION

1.1 Classification Using Multilayer Perceptron (MLP)

Classification is a process of assigning an object to one of several pre-specified categories or classes (Windeatt, 2008). Classification task aims to make the learning process and pattern recognition become explicit, where it can partially or entirely be automated by computers. Automating the classification to obtain optimal performance has been investigated in various disciplines such as medicine (Calcagno *et al.*, 2010), industrial sectors (Balabin *et al.*, 2010) and finance (Marinaki *et al.*, 2010). Although there are many types of classifiers such as Bayes and k_n-nearest neighbour, Multi-Layer Perceptron (MLP) has become one of the most widely used classifiers (Windeatt, 2008; Chabaa *et al.*, 2010; Balabin *et al.*, 2010; Marinaki *et al.*, 2010; Calcagno *et al.*, 2010). MLP becomes one of the popular classifiers because it has the ability to handle the non-linearity nature of most classification problems and learn complex tasks. Hence, MLP has been applied in this research work.

1.2 Multilayer Perceptron (MLP)

MLP is a type of Artificial Neural Network (ANN). ANN is inspired by the function of the human brain. The brain is the central element of our nervous system. It is joined by receptors that carry sensory information to it and deliver action commands to effectors. The brain itself has a network of about 10^{11} neurons that are interconnected through sub-networks called nuclei (Haykin, 2008). The sensory system and its sub-networks in the brain are exceptionally good at decomposing

complex sensory information into those fundamental components that are the crucial element of sense.

Amongst all ANNs, MLP is the most widely used, accounting for more than 70% of ANN applications (Tsai, 2009). It has been proven in the literature that MLP is able to solve a variety of problems, including classification (Ouelli *et al.*, 2012; Barnaghi *et al.*, 2012; Rowan *et al.*, 2007; Balabin and Safieva, 2008; Yan *et al.*, 2004; Xia and Yang, 2000; Ren *et al.*, 2000), system modelling and prediction (Niroobaksh *et al.*, 2012; Yarlagadda and Khong, 2001; Maqsood *et al.*, 2004) and function approximation (Lee *et al.*, 2004).

A basic MLP is a simple perceptron consisting of input, hidden and output layers with weight and connections laid between them as shown in Figure 1.1.

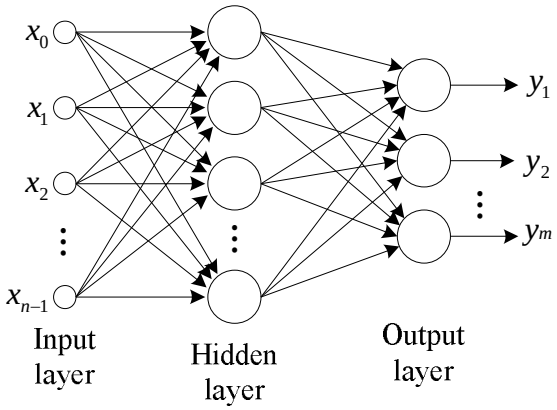

Figure 1.1 MLP architecture

The MLP neural network will process the data that have been presented to the input layer by multiplying at the weight layer (Noriega, 2005). The outcome of this multiplication will be processed by neurons in the output layer using a particular function that verifies whether or not the output nodes fire. The process of finding the correct values for the weight is called learning rule. Finding the correct values of

weight can be done using a learning paradigm called supervised learning which sometimes is referred to as training. The term ‘supervise’ refers to the fact that the input data are constituted with correct output, which acts like a ‘teacher’. Beginning with random weights, input data present the network and the initial guess on what output should be, is made. During the training phase, the error between the output of the network and the actual output value is measured and the weights are changed in order to minimize the error. Then, the MLP neural network is tested with new data which have never been observed by the performance and to determine whether it can work well when new data are presented.

Although MLP has been proven to be a good classifier, its learning algorithm has complex error surface that can get trapped in local minima (Windeatt, 2008). The problem of local minima is caused by a gradient descent algorithm which is used to train the MLP neural network to find globally optimal solution (Ng *et al.*, 2012). The training is believed to have reached a local minimum when there is no obvious change in the error function through large number of epochs, because of the change of weight becomes negligible. Hence, the performance of MLP will remain unchanged due to the insignificant change of weight. There are different ways of trying to overcome this local minima problem (Haykin, 2008) and MLP ensemble (MLPE) is one of the methods (Valdovinos and Sanchez, 2006; Adhikari and Agrawal, 2012).

Another problem when employing MLP is the selection of the best-performed MLP system. During the training phase, several MLP neural networks might give the same highest percentage of accuracy. Theoretically, in such a case, the MLP neural network with the least number of hidden PE is chosen to represent the problem due to the fact that it has simpler network architecture. However, a problem is

encountered when not all of the best MLPs (with the same highest percentage of accuracy) show the same performance towards a new set of data because of the different ways in their generalization during learning and different architectural structure (i.e. different number of hidden neurons). As an example, two MLP neural networks, say MLPa and MLPb give 90% of accuracy when presented with 100 patterns. The first pattern is correctly classified by the MLPa, yet is incorrectly classified by MLPb. The second pattern is incorrectly classified by MLPa but is correctly classified by MLPb and so on. At the end of the training process, both of the MLP neural networks give 90% of accuracy as they produce the same number of correctly classified patterns although they disagree with some of the presented patterns. Hence, logically, utilization of MLPa and MLPb in a MLPE, gives a better chance to improve the overall classification accuracy.

The structure of an integrated MLPE is as shown in Figure 1.2. The integrated MLPE system consists of an assembly of several MLP neural networks and a voting system. The MLPs in MLPE are first independently trained with the same set of data. Then, the MLPs with highest accuracy are chosen to be part of a MLP ensemble. In general, the MLPs cannot be combined in parameter (i.e weight) space as their integration involves 'stacked' MLP neural networks (see Figure 1.2). Hence, the ensemble needs a voting system to select the final output. The voting system is responsible to determine the best output solution for the data presented. The selection is done based on a specific voting algorithm.

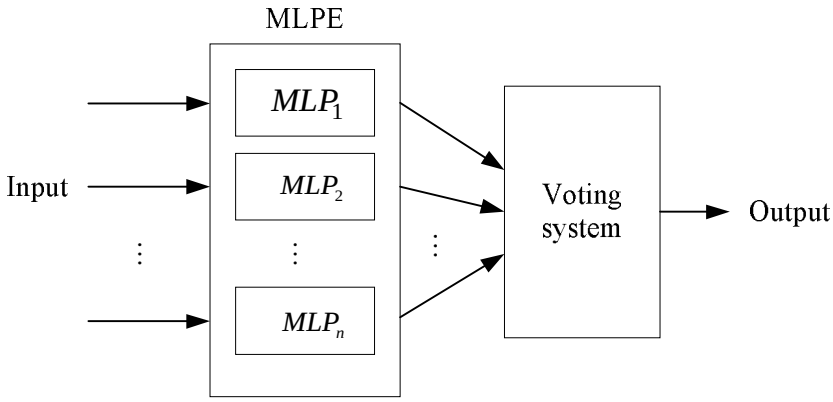

Figure 1.2 Structure of an integrated MLPE

The most widely used voting technique is majority voting (Bouzane *et al.*, 2011; Oliveira, 2009; Binsaeid *et al.*, 2009; Bhattacharia and Chaudhuri, 2003). It is a non-statistical technique that mainly depends on the agreement among individual MLPs in an ensemble. When there is a consensus vote on one output class, the output will be assigned to that class. However, when the draw number of votes occurs, the data will be classified as reject class. This happens because MLPE fails to classify it as any of the available output classes.

Another voting algorithm is trust voting (TV). It is a statistical-based approach that uses confidence measure to determine which MLP output is to be used as the output of the ensemble (Kumar *et al.*, 2000). This trust voting scheme has been employed by Kumar *et al.* (2000), Hartono and Hashimoto (2004) and Mohamad-Saleh *et al.* (2011) to construct MLPE to improve the classification accuracy.

1.3 Problem Statements and Motivation

MLPE can be developed using two different methods, it is either focused on the pre-training or the post-training stage. At the pre-training stage, MLPs are trained using

different training subsets to create distinct individual MLPs. The bootstrap and boosting are the statistical approaches that can be used to create those subsets. However, this requires more time since the MLP has to be trained several times to create distinct individual MLPs. On the other hand, the post-training approach creates distinctive individual MLPs by using the same training set with different initial training parameters. A single output from an ensemble is chosen based on the voting algorithm. This work uses the post-training approach in developing an MLPE to reduce training time.

Another method to construct an ensemble has been proposed by Bishops and Svensen (2003). They propose the mixture of experts as one of the methods in constructing ensemble. The term 'expert' here refers to the combination of different types of ANN architectural models. Such ensemble has been given better performance, but it increases the complexity of the system. Hence, its development was time-consuming because of the complexity of dealing with different output representations of different ANN architectures. Thus, this work only employs MLP to construct an ensemble.

MV is one of the popular voting algorithms used to develop an ensemble (Bouzane *et al.*, 2011; Oliveira *et al.*, 2009; Binsaeid *et al.*, 2009; Bhattacharia and Chaudhuri, 2003). The advantages of this voting algorithm lie in the fact that it is easy to understand and it is simple to implement regardless of the form of the MLP output representation. The drawback of this voting algorithm is that it requires an odd number of MLPs to avoid the occurrence of tie vote that leads to reject class. To avoid such problem, it is best to use more than one MLP to construct an ensemble. In this work, the minimum number of MLPs employed is three. The TV approach is rather more complex compared to MV because it involves the calculation of

confidence measure. It uses two most significant output bits to calculate each of the MLP output confidence measure. The confidence measure represents how confident an MLP is with its output. The TV method has been extensively used for a variety of applications. One limitation of this voting algorithm is that it considers only two bits, i.e. the highest and second highest, and abandon the others, even though there are more than two outputs in an MLP. It has become an encouragement to design a new voting algorithm that has good performance and easy to implement as there are a limited number of voting schemes that have been employed in constructing an MLPE.

MLP trained with different training algorithms may have different generalizations over the same presented data. Hence, it drives some inspiration to look into the performance of the MLPE constructed by different kinds of training algorithms and trained on the same training data. By using the same training data, the performance of MLPs is comparable since they only differ in their initial training parameters. Three different kinds of training algorithms used to train MLP neural networks in this work are the Levenberq Marquardt (LM), Resilient Backpropagation (RP) and Bayesian Regularization (BR). Thus, three is the minimum number of MLP used to construct MLPE.

1.4 Research Objectives

The main goal of this research is to design a new method to develop an integrated MLPE consisting of an MLPE that uses a new proposed voting algorithm aimed at improving classification accuracy. To accomplish the aim, this work focuses on the following objectives:

i. To propose and develop an MLPE consisting of MLPs that differ in training algorithm, initial weights and size to attain a variety of intelligent systems.

i. To propose and investigate a new voting scheme to improve classification accuracy and reduce rejects class cases.

ii. To assess the performance of developed MLPE employing proposed voting technique by comparison with existing commonly used voting schemes.

In this research work, an MLPE was developed using MLPs that are differing in their initial weights, training algorithm and architecture. All the MLPs were trained on the same training data. The best MLPs performed from each training algorithm are selected to become the members of MLPE

A new voting algorithm, trust--sum voting (TSV) is proposed as the voting technique for the MLPE system. The developed MLPE (MLPE-TSV) was tested on four benchmark case studies under the domain of classification. Then, the proposed MLPE-TSV performance was compared with MLPE-TV and MLPE-MV to assess its performance.

1.5 Thesis Outline

This chapter briefly introduces some preliminaries on this research work. It discusses the problems associated with MLPE, leading towards the motivation of research. The research objectives are listed and explained.

Chapter 2 reviews the literature on MLP, including its architecture and training algorithm. It also presents the literature of MLPE constructed by using voting

schemes. Two different voting schemes which are majority voting (MV) and trust voting (TV) are discussed.

Chapter 3 concentrates on the methodologies of this research work. The details of the steps in developing an MLPE using the proposed voting scheme and existing voting schemes are explained.

Chapter 4 presents the application of the proposed voting scheme in constructing MLPE. The applicability of the voting scheme was tested on four case studies in the classification domain. The first one is the ECT data with 66 input and 6 output classes, followed by the Landsat image satellite with 36 input and six output classes. The other two case studies have two outputs with German credit data having 24 attributes whilst Pima Indian diabetes has only eight inputs. The performance of MLPE using the new voting scheme in each case study was compared to singular MLPs and MLPE using TV and MV.

Chapter 5 presents the whole conclusion of this research work. From the results obtained, the MLPE new voting scheme shows outstanding performance compared to singular MLPs and MLPE using existing voting schemes. The developed MLPEs using MLPs with different training algorithms demonstrate superior performance compared to singular MLPs. The overall results illustrate that MLPE using the proposed voting scheme is able to perform the classification task for multiple output classes. The areas to be pursued as the future work are also suggested.

CHAPTER 2

A REVIEW OF MULTILAYER PERCEPTRON (MLP) AND MULTILAYER PERCEPTRON ENSEMBLE (MLPE)

2.1 Introduction

An artificial neural network (ANN), often called a neural network, is inspired by biological neurons. In the ANN, the nodes or neurons can be seen as computational units. They receive inputs and process them to produce an output. The neurons can be trained to classify an object according to their feature using examples (Padhy, 2005). In this chapter, the review of the ANN including their features will be discussed. Then, one of the most widely used ANN types which is the multilayer perceptron (MLP) will be discussed.

The next section presents a review on the multilayer perceptron ensemble (MLPE). The ensemble is an integration of several MLP neural networks to produce a single system. The aim of the MLPE is to generate more certain, precise and accurate system results. Various researches have been conducted and the findings have proven that the ensemble has superior performance to any singular ANN (Bhattacharya and Chaudhuri, 2003; Dietrich, 2002; Brown, 2004). In order to develop an MLPE, a voting technique is needed. The last section of this chapter presents a review on two existing voting techniques and they are the majority and trust voting.

2.2 Neuron Physiology

The neuron is the fundamental element of the nervous system, particularly the brain (Padhy, 2005). A biological neuron consists of three main components: cell body, dendrites and axon (see Figure 2.1). There is a cell body or soma that contains a nucleus in a neuron and each of the neuron has dendrites that receive connections from the other neurons. Neurons also have an axon which makes its way out from the neuron and in the end splits into a number of strands to make a connection with the other neurons. Synapses are the points where the neurons interact with the other neurons. A neuron can receive 10,000 or more synaptic contacts and can be ventured onto thousands of target cells (Haykin, 2008).

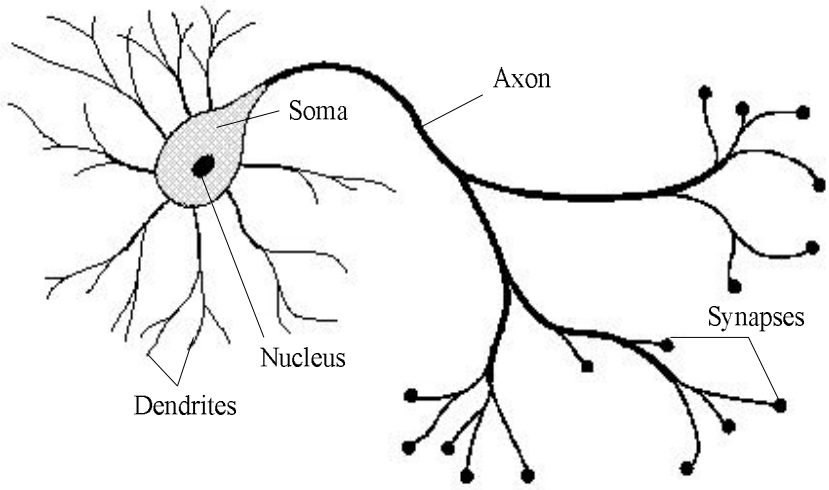

Figure 2.1 A biological neuron

Due to the electrical properties of the neuronal membranes, the signals that reach the dendrite rapidly decay in strength in time (temporal) and over distance (spatial), and thus lose the facility to stimulate the neuron, except for the fact that they are supported by another signal occurring at almost the same time and/ or nearby the locations (Ham and Kostanic, 2001). The soma sums the arriving signals (inputs)

from the dendrites and also sums the signals from numerous synapses on its surface. When the threshold level of the sum of the received signals is reached, the neuron generates an action potential which fire and transmit an action potential of its axon to other neurons or target cells outside the nervous system. Nevertheless, if the threshold level of the inputs is not reached, the inputs will quickly decay and will not generate an action potential. The strength of the inputs is measured by the number of action potential generated per second.

2.3 Artificial Neuron

An artificial neuron is an information processing unit that is essential to the operation of the ANN (Padhy, 2005). Figure 2.2 shows the schematic representation of an artificial neuron.

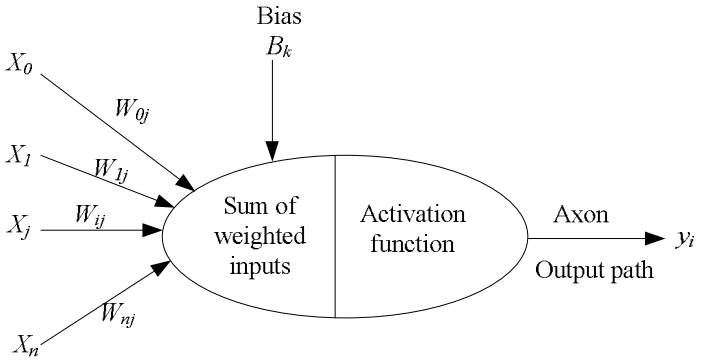

Figure 2.2 An artificial neuron

It consists of a set synapse or a connecting link and each of the links is characterized by a weight or strength of its own. The values of weights, w_0, w_1, w_2,…,w_n are to determine the strength of the input vector $\boldsymbol{X} = [x_0, x_1, x_2, \ldots, x_n]^T$. Each input is multiplied by an associated weight of the neuron connection $\boldsymbol{X}^T\boldsymbol{W}$. The synaptic

weights of an artificial neuron can have positive or negative values according to the acceleration or inhibition of the electrical signals flow (Padhy, 2005).

The processing element consists of two parts. The first part is an adder, used to sum up the input signals. The second part consists of an activation function which is used to limit the output of a neuron. The activation function is also referred as the squashing function, which performs a mathematical operation to squash the amplitude of the output signal into some finite ranges (Chakraborty, 2010). An external bias, B_k is also applied to the neuron. It is used to raise or to reduce the net input of the activation function (Padhy, 2005).

There are many different types of activation functions and the selection of one type over another depends on the problem that the ANN network needs to solve. The current ANN model often uses a sigmoid (S-shaped) activation function (Acharya *et al.*, 2003; Nkwogu and Allen, 2012).

Figure 2.3 shows the logistic sigmoid activation function. For the range $-\infty < v_q < \infty$where v_q is the internal activity potential of neuron q, $f(v_q)$ is given by (Haykin, 2008)

$$f\left(v_q\right)=\frac{1}{1+e^{-v_q}} \tag{2.1}$$

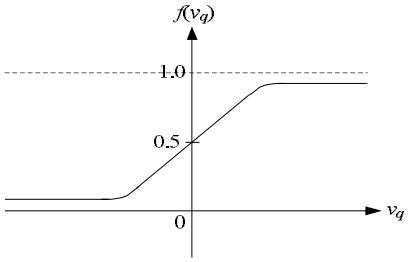

Figure 2.3 The logistic sigmoid activation function

Alternatively, MLP can use the hyperbolic tangent sigmoid activation function (see Figure 2.3) and can be written as (Haykin, 2008)

$$f\left(v_q\right)=\frac{e^{v_q}-e^{-v_q}}{e^{v_q}+e^{-v_q}} \tag{2.2}$$

The range of the activation function for the tangent sigmoid is -1 to +1.

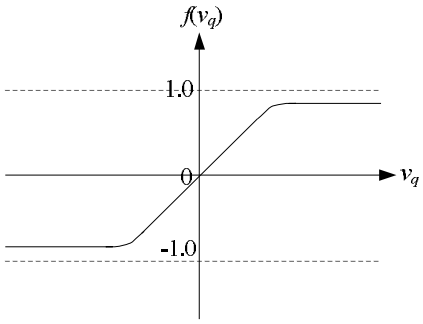

Figure 2.4 The hyperbolic tangent activation function

Occasionally, the ANN uses the linear activation function (refer to Figure 2.5) and this is given by

$$f\left(v_q\right)=kv_q \tag{2.3}$$

where *k* is the slope of the straight line.

However, the use of the linear activation function will remove the nonlinear behavior of the ANN (Padhy, 2005). Thus, the ANN cannot perform on the non-linear problem.

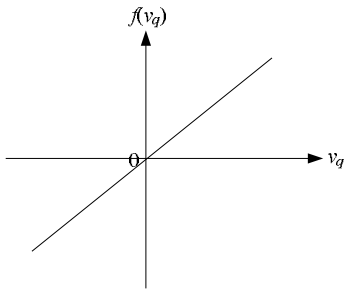

Figure 2.5 The linear activation function.

2.4 Artificial Neural Network (ANN)

An ANN is a huge parallel distributed processor that has a natural tendency for storing experiential knowledge and making it available for use (Haykin, 2008). It consists of highly interconnected processing elements (artificial neuron) in an architecture, which is inspired by the cerebral cortex structure of the brain (Padhy, 2005).

2.4.1 Applications of Artificial Neural Networks

ANNs have been used in many diverse applications because of their ability to generalize and describe non-linear processes. The applications of the ANN can be classified into three major categories; classification, pattern association and function approximation (Ham and Kostanic, 2001).

Classification –The ANN is trained to be able to classify the input patterns presented. As one type of classifiers, the ANN can serve numerous areas for different purposes such as for the medical (Kamruzzaman *et al.* 2004; Chai *et al.* 2004), and

industrial purposes for control (Balabin and Safieva 2008; Yan *et al.,* 2004; Xia and Yang, 2000; Ren *et al*., 2000).

Pattern association **-** Pattern association can be classified into two types; autoassociation and heteroassociation. The association entails constantly showing the ANN a certain pattern and the ANN should be able to store it and when a distorted image of the same pattern is presented, the ANN should retrieve it. Heteroassociation differs from autoassociation in the sense that it is supervised. Some examples of works that are related to pattern association are business transactions (Kar and De, 2009) and robot controller (Zin *et al.,* 2009).

Function approximation – The ANN can be used as the function approximator where the ANN is able to receive an input and desired output and then, approximate the function that has been used. The work done by Lee *et al.* (2004) is one of the examples of solving the function approximation problem using the ANN.

2.4.2 Multilayer Perceptron (MLP)

MLP is an important class of ANN (Haykin, 2008). Basically, the MLP neural network consists of three layers; the input layer, hidden layer and output layer. The input signal transmits through the MLP neural network in a forward direction on a layer-by-layer basis (see Figure 2.6). The first and second hidden layers consist of hidden processing elements (PEs) also known as neurons which process the information sent from the input layer. This single hidden layer is sandwiched between the input and output layers. For n input neuron, the input vector, $\boldsymbol{x} = [x_0, x_1 \ldots x_{n-1}]^{\mathrm{T}}$ and $\boldsymbol{x} \in \Re^{n-1 \times 1}$ Meanwhile, $\boldsymbol{y}$ is the vector response of the MLP neural

network where $\boldsymbol{y} \in \Re^{k \times 1}$. The neuron is regarded by $n + 1$ weights which multiply each input and activation function that are applied to the weighted sum of the inputs in order to produce the neuron's output. The weighted sum of inputs includes the bias often called the net input or internal activation potential, v. The neuron output is the function of the net input, $f(v)$ and can be written as

$$y = f(v) = \sum_{i=0}^{n-1} x_i w_i + w_n \tag{2.4}$$

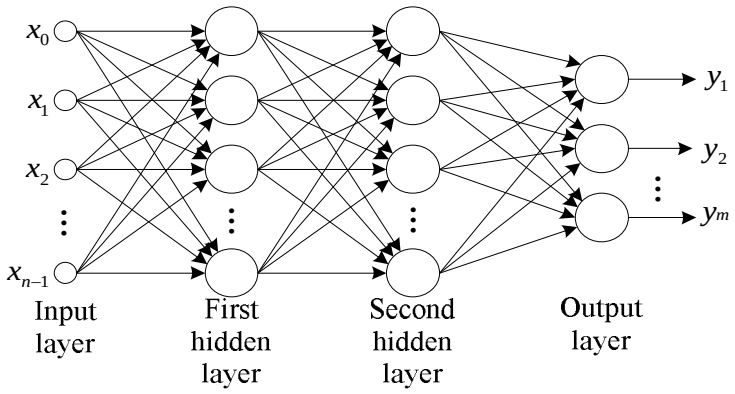

Figure 2.6 An architecture of the MLP with two hidden layers.

An MLP has three distinctive features

1. The model of each neuron in the MLP neural network comprises of nonlinearity at the output end. The nonlinearity is crucial or else the input-output relation of the MLP neural network could be decomposed to that of a single layer perceptron (Haykin, 2008). Indeed, this nonlinearity of the neuron is smooth (i.e. differentiable everywhere).

2. The MLP neural network consists of one or more hidden layers of PE that are not part of the input or output of the network. These hidden PE allow the MLP neural network to learn a complex task of extracting evolutionary significant aspects from the input patterns (vectors).
3. The MLP neural network demonstrates high degree of connectivity, determined by the synapses of the MLP neural network (Haykin, 2008). A change in the connectivity of the network needs a change in the population of synaptic connections or their weights.

MLP has been successfully applied to various classification problems by training it in a supervised manner using a popular algorithm known as error backpropagation (Haykin, 2008; Valdovinos and Sanchez, 2006; Adhikari and Agrawal, 2012). Hence, it is chosen to be employed in this work. The term 'supervised' refers to the existence of a 'teacher' during the training. The term 'teacher' is in reference to the desired outputs that are paired up with the corresponding inputs. The weights are adjusted according to the error obtained during the learning process.

2.4.3 Learning in MLP

The MLP training process starts by initializing all weights to a small non-zero value and frequently these weights are generated randomly. One complete presentation of the entire training set during the learning process is called an epoch. The learning process remains on the epoch-by-epoch basis until the threshold levels and the synaptic weights of the network stabilize and the average squared error over the entire training set converges to some minimum values. For a given training set, the MLP network may learn in one of two basic approaches; Pattern mode and batch

mode. In the pattern mode, the weight updating is done after the presentation of each training input. In the batch mode learning, the weights are updated after a sequence of inputs is presented.

2.5 Backpropagation Training Algorithm

Learning in MLP is almost always carried out using the backpropagation algorithm. The algorithm was first developed by Werbos (1974) and rediscovered by Parker in 1982, LeCun in 1985 and Rumelhart *et al.* in 1986. The work done by Rumelhart *et al.* proposes the use of error backpropagation to set the weights and to train the MLP neural network (Graupe, 2007).

The backpropagation can be applied to MLP in any number of hidden layers. The aim of the training is to adjust the weights, so that the application of a set of inputs can well generate the desired output. The MLP training involves two phases. In the forward pass, an activity pattern (input vector) is applied to the sensory nodes of the MLP neural network and its effects transmit through the MLP neural network, layer by layer (Haykin, 2008). Finally, a set of outputs is generated as the MLP neural network's actual response. Throughout the forward pass, the synaptic weights of the MLP neural network are all fixed. On the other hand, during the backward pass, the synaptic weights are all adjusted in accordance with the error correction rule. Particularly so, the actual response of the MLP neural network is subtracted from a desired output to produce an error signal. Then, this error signal transmits backwards through the MLP neural network against the direction of the synaptic connections. The synaptic weights are amended in such a way to make the actual response of the MLP neural network move closer to the desired response (Haykin, 2008).

The backpropagation algorithm offers an 'approximation' to the trajectory in weight space by the scheme of the steepest descent. The smaller value of the learning parameter, μ, the smaller will be the changes to the synaptic weights in the MLP neural network from one iteration to the next and the smoother will be the trajectory in weight space (Haykin, 2008). However, this improvement will result in a slower rate of learning. If the learning parameter, μ is too large, this will accelerate the learning rate, but unfortunately it will result in large changes in the synaptic weights in such a way that the network may become unstable.

Although the backpropagation algorithm has less computational complexity, it suffers from slow convergence rate and is easily trapped in the local minima and cannot converge to the global minimum (Ng *et al.,* 2012). A MLP neural network is caught in local minima when the changes of weights become negligible. This leads an insignificant change in the error function through a large number of epochs and hence, there is no change in the output of a MLP. Therefore, the target error value cannot be obtained and thus the training will be unsuccessful. A lot of researches have been done to improve the backpropagation algorithm to overcome the local minimum problem and accelerate the learning process (Wang *et al.*, 2004; Ng *et al.,* 2004).

There are several training algorithms adopted to accelerate the learning of backpropagation algorithm such as Levenberq Marquardt (LM), Resilient backpropagation (RP) and Bayesian regularization (BR). These several training algorithms are the modification of the standard backpropagation algorithm.

2.5.1 Levenberq Marquardt (LM) Training Algorithm

The Levenberq Marquardt (LM) algorithm represents a simplified version of the Newton's method (Haykin, 2008). Newton's method is a well known method for a numerical optimization technique with quadratic speed of convergence. The LM algorithm was introduced by Levenberq (1984) and Marquardt (1963) and typically serves as the fastest training algorithm (Hagan and Menhaj, 1994).

An apparent problem with Newton's method lies in the computational requirements concerned with calculating the inverse of the Hessian matrix (Haykin, 2008). The LM algorithm provides a feasible alternative to Newton's method with less complexity and roughly the same convergence speed. The problem of training MLP has to be formulated as a nonlinear optimization problem as to be able to apply the LM algorithm. Consider an MLP network shown in Figure 2.6. The task of the ANN training can be viewed as determining a set of network weights that minimizes the error between the target and the actual output of network for all the patterns in the training set. If the number of pattern is finite, the energy function can be written as (Ham and Kostanic, 2001)

$$\mathrm{E}(\mathbf{w})=\frac{1}{2}\sum_{q=1}^{Q}\left(\mathbf{d}_q-\mathbf{y}_q\right)^{\mathrm{T}}\left(\mathbf{d}_q-\mathbf{y}_q\right)=\frac{1}{2}\sum_{q=1}^{Q}\sum_{\mathrm{h}=1}^{\mathrm{m}}\left(d_{qh}-y_{qh}\right)^2 \tag{2.5}$$

where Q is the total number of training pattern, $\boldsymbol{w}$ represents the vector containing all the weights in the network, $\boldsymbol{d}_q$ is the desired output and $\boldsymbol{y}_{qh}$ is the actual network output due to the qth training pattern. Based on Newton's method, the set of optimal weights that minimizes the energy function in (2.5) can be determined by applying

$$\boldsymbol{w}(k+1)=\boldsymbol{w}(k)-\boldsymbol{H}_k^{-1}\boldsymbol{g}_k \tag{2.6}$$

where

$$\boldsymbol{H}_k = \nabla^2 E(\boldsymbol{w})|_{\boldsymbol{w}=\boldsymbol{w}(k)} \tag{2.7}$$

and

$$\boldsymbol{g}_k = \nabla E(\boldsymbol{w})|_{\boldsymbol{w}=\boldsymbol{w}(k)} \tag{2.8}$$

By defining $P=kQ$, (2.5) can be rewritten as

$$E(\boldsymbol{w}) = \frac{1}{2}\sum_{p=1}^{P}\left(d_p - y_p\right)^2 = \frac{1}{2}\sum_{p=1}^{P} e_p^2 \tag{2.9}$$

where e_p is the network error given by

$$e_p = d_p - y_p \tag{2.10}$$

The gradient of the energy function in (2.8) can be computed as follows (Haykin, 2008)

$$\boldsymbol{g} = \frac{\partial E(\boldsymbol{w})}{\partial \boldsymbol{w}} = \begin{bmatrix} \frac{\partial \sum_{p=1}^{P} e_p^2}{\partial w_1} \\ \frac{\partial \sum_{p=1}^{P} e_p^2}{\partial w_2} \\ \vdots \\ \frac{\partial \sum_{p=1}^{P} e_p^2}{\partial w_N} \end{bmatrix} = \boldsymbol{J}^T \boldsymbol{e} \tag{2.11}$$

Where $\boldsymbol{J} \in \Re^{P \times N}$ is the Jacobian matrix defined by

$$\boldsymbol{J} = \begin{bmatrix} \frac{\partial e_1}{\partial w_1} & \frac{\partial e_1}{\partial w_2} \cdots & \frac{\partial e_1}{\partial w_N} \\ \frac{\partial e_2}{\partial w_1} & \frac{\partial e_2}{\partial w_2} \cdots & \frac{\partial e_2}{\partial w_N} \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ \frac{\partial e_P}{\partial w_1} & \frac{\partial e_P}{\partial w_2} \cdots & \frac{\partial e_P}{\partial w_N} \end{bmatrix} \tag{2.12}$$

By using the expression in (2.12), the Hessian can be expressed as

$$[\nabla^2 E(\boldsymbol{w})] = \boldsymbol{J}^T\boldsymbol{J} + \boldsymbol{S} \tag{2.13}$$

Where matrix $\boldsymbol{S} \in \Re^{N\times N}$ is the matrix of the second order derivatives given by

$$\boldsymbol{S} = \sum_{p=1}^{P} e_p \nabla^2 e_p \tag{2.14}$$

When approaching the minimum of the energy function, the elements of matrix $\boldsymbol{S}$ become small, and the Hessian matrix can be closely approximated by

$$\boldsymbol{H} \approx \boldsymbol{J}^T\boldsymbol{J} \tag{2.15}$$

Substitute (2.11) and (2.15) into the expression of Newton's method expressed in (2.5) result in

$$\boldsymbol{w}(k+1) = \boldsymbol{w}(k) - [\boldsymbol{J}_k^T\boldsymbol{J}_k]^{-1}\boldsymbol{J}_k^T e_k \tag{2.16}$$

where subscript k indicates the evaluation of the suitable matrices at $\boldsymbol{w} = \boldsymbol{w}(k)$. However, the iterative update given in (2.16) needs the inversion of matrix $\boldsymbol{H} = \boldsymbol{J}^T\boldsymbol{J}$which may be ill-conditioned or even singular (Ham and Kostanic, 2001). This problem can be resolved using the following adjustment of (2.15)

$$\boldsymbol{H} \approx \boldsymbol{J}^T\boldsymbol{J} + \mu\boldsymbol{I} \tag{2.17}$$

whereμ is a small number and $\boldsymbol{I} \in \Re^{N\times N}$ is the identity matrix. Substituting (2.11) and (2.14) constructs the LM algorithm for updating the network weights given by (Ham and Kostanic, 2001)

$$\boldsymbol{w}(k+1) = \boldsymbol{w}(k) - [\boldsymbol{J}_k^T\boldsymbol{J}_k + \mu_k\boldsymbol{I}]^{-1}\boldsymbol{J}_k^T e_k \tag{2.18}$$

For a small value of μ_k, (2.17) approaches, the Newton's algorithm is given in (2.16). If the value of μ_k is increased, the second term inside the square bracket (refer to (2.18) becomes dominant and the updated equation can be written as

$$\boldsymbol{w}(k+1) = \boldsymbol{w}(k) - [\mu_k \boldsymbol{I}]^{-1} \boldsymbol{J}_k^T e_k = \frac{1}{\mu_k} \boldsymbol{J}_k^T e_k \tag{2.19}$$

The major problem in executing the LM algorithm can be seen in the calculation of the Jacobian matrix J (***w***). Each term in the matrix has

$$J_{ij} = \frac{\partial e_i}{\partial w_j} \tag{2.20}$$

The simplest method to compute the derivative in (2.20) is using the approximation

$$J_{ij} \approx \frac{\Delta e_i}{\Delta w_j} \tag{2.21}$$

where Δe_i represents the change in the output error due to the small perturbation of the weights Δw_j. The value of $\Delta w_j$is kept small, at least an order of magnitude smaller than the current learning rate parameter, μ_k. The weight update can be performed using (2.19) after computing the Jacobian matrix.

2.5.2 Resilient Backpropagation (RP) Training Algorithm

Resilient Backpropagation (RP) proposed by Riedmiller and Braun in 1993, is a training scheme that performs a direct adaptation of the weight step based on the local gradient information. The size of the actual weight perturbation, Δw_{ij} is not only dependent on the learning rate, but also on the partial derivative $\frac{\delta e}{\delta w_{ij}}$. The effect of the vigilantly adapted learning rate can be drastically disturbed by the unpredictable behaviour of the derivative itself (Riedmiller and Braun, 1993). The purpose of using the RP algorithm is to eliminate the harmful effect caused by the magnitude of partial derivatives (Haykin, 2008).