Quantum Neural Network Application for Weather Forecasting

Thesis submitted in partial fulfillment of the requirements for the award of degree of

Master of Engineering
in
Software Engineering

By:

Gurwinder Singh
(80731008)

Under the supervision of:
Dr. V. P. Singh
Assistant Professor

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
THAPAR UNIVERSITY
PATIALA – 147004

JULY 2009

Certificate

I hereby certify that the work which is being presented in the thesis entitled, **"Quantum Neural Network Application for Weather Forecasting"**, in partial fulfillment of the requirements for the award of degree of Master of Engineering in Software Engineering and submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of Dr. V. P. Singh and refers other researcher's works which are duly listed in the reference section.

The matter presented in this thesis has not been submitted for the award of any other degree of this or any other university.

Gurwinder Singh.

(Gurwinder Singh)

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

15/7/09

(V.P. Singh)
Assistant Professor
Computer Science and Engineering Department,
Thapar University, Patiala.

Countersigned by

(RAJESH KUMAR BHATIA)
Assistant Professor & Head
Computer Science & Engineering Department,
Thapar University,
Patiala.

(R.K.SHARMA) 21/7/09
Dean (Academic Affairs),
Thapar University,
Patiala.

Acknowledgement

First and foremost, I would like to express my sincere gratitude to my guide **Dr. V. P. Singh,** Assistant Professor, Computer Science and Engineering Department for immense help, guidance, stimulating suggestions, and encouragement all the time with this thesis work. This work would have not been possible without her encouragement. He always provided a motivating and enthusiastic atmosphere to work with; it was a great pleasure to do this thesis under his supervision.

I am equally grateful to **Dr. Rajesh Kumar Bhatia,** Assistant Professor and Head, Computer Science and Engineering Department for their appreciation and satisfactorily healing me off my inexperienced inquisitions about the new subject.
I am grateful to **Dr. R. K. Sharma**, Dean of Academic Affair for his constant encouragement that was of great importance in the completion of the thesis

I would also like to thank all the staff members and PhD scholar Ms. Shashi Bhanwar who were always there at the need of the hours and provided with all the help and facilities, which I required for the completion of my thesis. I am deeply indebted to my parents and friends for their inspiration and ever-encouraging moral support, which enabled me to pursue my studies.

I am also very thankful to the entire faculty and staff members of Computer Science and Engineering Department for their direct-indirect help, cooperation, love and affection that made my stay at Thapar University memorable.

Gurwinder Singh

Gurwinder Singh

(80731008)

Abstract

Weather forecasts are made by collecting quantitative data about the current state of the atmosphere and using scientific understanding of atmospheric processes to project how the atmosphere will evolve. This research examines and analyzes the use of Quantum neural networks as a forecasting tool. Specifically, neural network's features such as parallel distributed processing, self-learning and fault tolerance are explored and the idea is then to combine Quantum Computation with the Neural Networks producing Quantum Neural Networks. Quantum computing is a proposed means of using quantum-mechanical effects to achieve efficient computation. The potential power of a quantum computing is in the superposition of states, allowing exponentially many computations to be done in parallel. During a calculation, the bits (called qubits) that are being manipulated are never in a definite one or zero state, instead can be thought of as being both a one and a zero simultaneously, which allows quantum neural networks to explore many solutions at the same time. While only briefly discussing neural network theory, this research determines the feasibility and practicality of using Quantum neural networks as a forecasting tool for the weather system. From the simulation results, it can be seen that the proposed model produces a reasonable accuracy in training which is conducted with 5 parameters (100 days) of the recent temperature, dew point, humidity, sea level pressure, wind speed with historical data. Once the model is trained, it is used to forecast the weather data for next time period, each time forecasting one point ahead.

Contents

List of Figures

List of Tables

Chapter 1 Introduction

Forecasting involves techniques designed to reduce uncertainty and to make better estimates of what will happen in the future. The entire process may be considered subjective, which involves intuition and years of experience or quantitative such as moving averages, trend projections etc.

In General, weather analysis and forecasting considered as the succession of the tasks based on to clearly understand the recent evolution and the actual situation of the atmosphere at all time scales and the pertinent information from at least one numerical model. This information used to assess the future evolution of the atmosphere in order to determine the most likely scenario synoptic weather forecasting. The other tasks are also considered to deduce finally the consequences of the expected situation in terms of weather elements (weather elements forecasting) and to evaluate the risk of the occurrence of hazardous phenomena. This information is used to prepare the meteorological information (including possible warnings) to be directed toward the various internal or external users [1].

Weather forecasting is not a purely mechanical linear process, for which standard practices and procedures can be directly applied. The work of the forecasters has evolved significantly over the years to take advantages of both scientific and technological improvements. The skill of numerical models has improved so much that some centers are automating routine forecasts to allow forecasters to focus on high impact weather or areas where they can add significant value. So it is not easy to determine a standard way to achieve weather forecasts.

The way the forecaster currently works depends on several factors such as the forecast range (medium-range, short-range, very short-range, now casting) and the size of the domain to be covered (large portion of the globe, regional domain, small country, city) and the geographical context and related climatologic information. Similarly, the potential risk associated with the expected weather at various ranges, the organization of the forecast service and the end-user who receives the forecasts (civil defense, aviation, marine, hydrologic and water management service, road administration,

medias etc) also the technical environment (available external and/or internal in site observations, satellite and radar images, Web access) [1].

These steps present a systematic way of initiating, designing, and implementing a forecasting system. The forecasting system is to be used to generate forecasts regularly over time, and data must be collected routinely. The major concerns in the forecasting of severe weather include: Possibility of occurrence, intensity and locations to be affected. In areas where severe weather forecasting is well developed, the various elements associated with the occurrence are defined, such as (a) The specific parameters and threshold for occurrence, (b) The evolution of meteorological parameters and patterns associated with the occurrence and (c) The climatology, which include the potential areas of occurrence [21].

Weather forecasts are made by collecting quantitative data about the current state of the atmosphere and using scientific understanding of atmospheric processes to project how the atmosphere will evolve. The chaotic nature of the atmosphere, the massive computational power required to solve the equation that describe the atmosphere and forecasts become less accurate as the range of the forecast increases. The need for accurate weather prediction is apparent when considering the benefits that it has. Meteorologist use different methods for weather prediction. Observation of atmospheric pressure, temperature, wind speed, wind direction, humidity and precipitation are made near the earth's surface by trained observers, automatic weather stations.

Artificial neural networks are parallel computational models, comprising closely interconnected adaptive processing units. The important characteristic of neural networks is their adaptive nature, where 'learning by example replaces programming', which makes the techniques very appealing in application domains for solving highly nonlinear phenomena. The various complex problems like weather prediction have been proved to be areas with ample scope of application of this sophisticated mathematical tool. A multilayer neural network can approximate any smooth, measurable function between input and output vectors by selecting a suitable set of connecting weight and transfer functions [2].

Recently, quantum computing has been indicated as the new research area. The point of quantum computing is that, during a calculation, the bits (called qubits) that are being manipulated are never in a definite one or zero state. Instead, they can be thought of as being both a one and a zero simultaneously, which allows a quantum computer to explore many solutions at the same time. The upshot is that, for a limited set of problems, quantum computers may offer a substantial speed up over normal computers. In recent, scientists have made use of the similarities between a certain type of quantum computation and neural networks to construct a very simple quantum neural network. The result may offer a faster and more robust form of computational domain.

In normal quantum computing, the qubit values might be encoded in the states of a bunch of atoms. These are then individually manipulated by direct operations on their states to perform the calculation; the answer is obtained by measuring the final state of the atoms. A quantum computation takes advantage of the strange behavior of quantum physics, and research in this area is still in an developing state. A quantum computer parallels the philosophy of a classical digital computer in approach, where the fundamental unit of information for the digital computer is a bit, for a quantum computer it is a quantum bit or qubit. But unlike the binary bit that is represented by a string of 1s and 0s. A qubit can exist in the 0 and 1 state corresponding to that of the classical bit as well as in a superposition of these states. It can be a 0 or 1, or it can exist simultaneously as both. This feature enables a quantum computer to perform a computation on many different numbers at once and then execute an "interference" with those results to arrive at a single solution.

Chapter 2 Literature Review

The development of Quantum Neural Network (QNN) starts all over the world which is in the state that the researcher explores it individually. In 1995, it's Kak who firstly presented the concept of quantum neural computation. It generated a new paradigm upon the combination of conventional neural computation and quantum computing. Perus pointed out that there is an absorbing comparability between neural networks and quantum parallelism in 1996. In 1998, a first systematic examination of quantum artificial neural network (QANN) was conducted by T Menneer in his PhD dissertation. At the same time, many QNN models were developed, in 1995, quantum inspired neural nets were proposed by Narayanan et al. In 2000, Ventura introduced quantum associative memory based on the Grover's quantum search algorithm and entangled neural network was presented by Li wei-gang. In 2005, Noriaki Kouda introduced qubit neural networks [12]. These QNNs have very high theory value and application potential due to that they combine the respective advantage of neural computation and quantum computation.

A general quantum gate network is made up of basic quantum gates, such as Hadamard gate, AND gate and Controlled-NOT gate etc. Gupta and Zia have proved that this network can solve the problems using only poly-logarithmic entangled qubits and at most poly-logarithmic steps. And it realizes the function of network by constructing unitary matrix using these basic quantum gates to evolve quantum state. Moreover, M.Andrecut has proved the minimum delay schedule T for the adiabatic evolution of Hamiltonian operator, which pledges the feasibility of this time-dependent quantum gate network. While the idea of such a computer was first proposed as early as the 1970s, it was not until 1994 when Peter Shor described his algorithm for factoring very large numbers did quantum computing emerge from academic curiosity to a potential reality. Shor's algorithm was designed specifically for a quantum computer. Quantum computers have a tremendous promise as a computational device and one day no doubt will surpass the classical digital computer. But in the interim, much research remains on error correction and quantum hardware before such systems enter the practical world. For the time being, quantum computing is still considered as a special discipline within theoretical physics [35].

Quantum computation and quantum information encompasses processing and transmission of data stored in quantum states. The central novelty of quantum theory lies in the description of the state of these particles. In some ways, the particles of quantum theory are like little tiny points of matter, as the name "particle" suggests. In others, they are like little bundles of waves. Modern quantum theory has enjoyed enormous empirical success, accounting for a huge array of phenomena and making striking predictions. It is possible to describe the basic posits of quantum theory compactly.

It is important to note that much of the power of classical artificial neural networks is due to their massively parallel, distributed processing of information and also due to the nonlinearity of the transformation performed by the network nodes (neurons). On the other hand, quantum mechanics offers the possibility of an even more powerful quantum parallelism which is expressed in the principle of superposition. This principle provides quantum computing an advantage in processing huge data sets. Though quantum computing implies parallel processing of all possible configurations of the state of a register composed of N qubits, only one result can be read after the decoherence of the quantum superposition into one of its basis states. However, entanglement provides the possibility of measuring the states of all qubits in a register whose values are interdependent. Though the mathematics of quantum mechanics is fairly well understood and accepted, the physical reality of what the theory means is much debated and there exist different interpretations of quantum mechanics, including: (a) Copenhagen interpretation [7], (b) Feynman path-integral formalism [8], (c) Many universes (many-world) interpretation of Everett, etc. The choice of interpretation is important in establishing different analogies between quantum physics and neurocomputing.

As the evolutionary operators in quantum mechanics must be unitary and certain aspects of any quantum computation must be considered as evolutionary. For example, storing patterns in a quantum system demands evolutionary processes since the system must maintain a coherent superposition that represents the stored patterns. On the other hand, other aspects of quantum computation preclude unitarity (and thus linearity) altogether. In particular, decoherence is a non-unitary process.

In the Copenhagen interpretation, non-unitary operators do exist in quantum mechanics and in nature. For example, any observation of a quantum system can be thought of as an operator that is neither evolutionary nor unitary. In fact, the Copenhagen school of thought suggests that this non-evolutionary behavior of quantum systems is just as critical to our understanding of quantum mechanics as is their evolutionary behavior. Now, since recalling a pattern from a quantum system would require the decoherence and collapse of the system at some point (at the very latest when the system is observed), it can be argued that pattern recall may be considered as a non-unitary process. In which case, the use of unitary operators becomes unnecessary. Since the decoherence and collapse of a quantum wave function is non-unitary and since pattern recall in a quantum system requires decoherence and collapse at some point, explicit use of this non-unitarity, in the pattern recall process is questionable. This decoherence of a quantum state can be considered as the analog of the evolution of a neural network state to an attractor basin. This analogy has been mentioned in the work [11].

As a second approach to reconciling the linearity of quantum mechanics with the nonlinearity inherent in artificial neural networks, consider the Feynman interpretation of quantum mechanics, which is based on the use of path integrals. Here nonlinearity can be both to the nonlinear form of the potential and also to the operation of the exponent. This fact has been used in approaches to modeling quantum neural networks by Elizabeth Behrman and coworkers [12, 13] and Ben Goertzel [14].

It is interesting to note that the symbol-manipulation methodology has been adopted as the basic paradigm within the field of Artificial Intelligence throughout the last several decades of its development, whereas intelligence itself is so integrally dependent upon adaptation to the environment.

As early as 1943 McCulloch and Pitts observed that finite logical relationships could be expressed with combinations of neuron models, for which they had proposed a simple mathematical derivation. McCulloch and Pitts proposed that a neuron could be approximated as a logic gate or switch that could be designed to function within the principles of Boolean algebra. By combining these individual gates into multiple

parallel sets, they further reasoned that these new neural networks could be made to perform any combination of logical functions. These computers are often merely referred to as neural networks. The neural systems of McCulloch and Pitts were only two-layer networks wherein information was introduced through the first layer and dispersed from the second layer. But in the late 1960's, the Cornell University psychologist Frank Rosenblatt introduced a structural variation that he called the "perceptron." [27]. Perceptron "learning" was based on modifying the weights of the non-active lines in response to an error – a form of "punishment" for guessing wrong. David Rumelhart and James McClelland first introduced a third layer, or hidden layer, of neurons between the input and output layers, called the multilayer perceptron, each neuron connected to every other neuron between adjacent layers, but connections are not allowed across more than one layer or within any given layer. Furthermore, multiple hidden layers are also admissible. Only the input and output layers are in direct contact with the environment. Next, they introduced the concept of "back propagation" of errors to enable the network to learn more effectively.

A wide range of interesting applications of neural networks motivated and helped in research. Lu et al. compared the effectiveness of neural networks and concluded that the ANNs perform better than regressions in decision-making. Fletcher & Goss used back-propagation (BP) neural network model to predict the weather, and reported improvement in the prediction accuracy over the model. Salchenberger, Cinar & Nicholas used a neural network and the probalistic model to forecast space weather using different parameters as inputs. The neural network performed as well or better than the logic model. Wu et al. applied neural network technology for the decision surface modeling of Climatologic domains. Denton compared neural networks to linear regression forecasting models and used the mean square error of their forecasts as a measurement of their relative performance. The analysis was performed with, (a) an ordinary linear regression, (b) in the presence of an outlier, (c) in the presence of multicollinearity, and (d) with a misrepresentation of the model [23]. Their results show that there was no significant difference in the performances in the first case, but the neural network performed better than the regression model in the last three cases. Kuo and Reitsch have compared the forecasting performance of neural networks and several conventional forecasting models. Neural networks showed a superior performance in almost all cases.

Chapter 3 Neural Networks

Artificial Neural Networks, also known as connectionist models or simply neural nets, refer to a collection of computational systems whose architecture is inspired by the structures and processes that are thought to occur in biological nervous systems.

3.1 Neuron

The symbol manipulations, component configurations and interfaces, and serial processing of the digital computer are a long way from approximating the architecture and neural functions of the brain. The numerous nodes that approximate neuronal behavior represent the dominant feature of neural-network architecture. A short description of the actual biological unit itself would help shed light on what the artificial system is trying to imitate.

3.1.1 Biological Model

The aggregate of the central nervous system is comprised of literally millions of interacting behavioral nodes called neurons. The functionality of each neuron, like that of an individual ant in an ant colony, can be reasonably approximated by a small number of rules. But the totality of these simple rules when applied over millions of neurons leads to a much more complex and rich behavior than ever could be predicted from the individual neurons

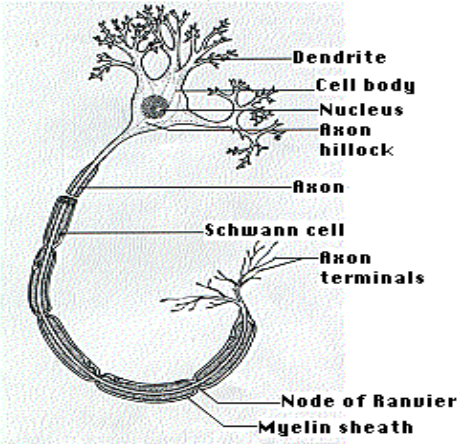

Figure 3.1 Biological Neuron [4]

A typical neuron is a cell with numerous short root-like fibers spraying out from it called dendrites and a long single fiber with a brush-like tip called an axon. The axon carries impulses away from the cell body, or soma, while the dendrites accept incoming impulses from other neurons and carry the signals toward the soma. Signal interchanges between neurons are accomplished at the synapses, located primarily at the dendrite terminals. Each signal pulse initiates the release of a neurotransmitter chemical that travels across the synaptic crack from the axon side and is received at the post-synaptic receptor site on the dendrite. In other words, the neuron integrates the impulses it receives from its dendrites through its synapses and either generates an action potential or does not. Each post-synaptic-potential pulse travels along its dendrite and spreads over the cell body until it eventually reaches the base of the axon, a region called the axon-hillock. The receiving neuron integrates the effects of thousands of such pulses received at the axon-hillock from its dendritic tree over a given time. When the integrated potential exceeds a predetermined threshold, the receiving cell then fires, generating an electrical impulse that travels down its axon to initiate this same sequence of events with neurons to which it is connected [4].

3.1.2. Mathematical Model

The architecture of neural networks simulates nerve cells or neurons connected together to form a massively parallel system network. Artificial neurons attempt to approximate linear as well nonlinear functions. Typically, neural networks are organized in layers comprised of numerous interconnected nodes, where each node is a simulated neuron represented by a system of multiple inputs to a processing element followed by a single output.

In the biological analogy, the neuron cell is represented by a component comprising the threshold logic required to receive and integrate input signals and to decide when the accumulated signal has reached the desired voltage threshold to release an output pulse. The threshold logic is denoted by the Greek letter sigma (Σ) in Figure 3.2. Its logic pattern behaves very much like a sigmoid, a mathematical function like that depicted in the insert. The dendrites are represented by the multiple input lines at the left of the logic fragment in the figure. The signal arriving along each input line is

modified by a weighting factor, a fragment capable of being adapted to a particular training pattern that may be desired.

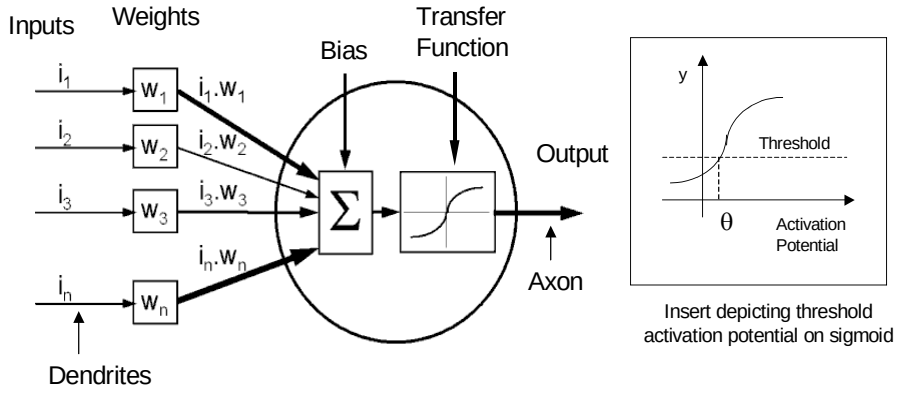

Figure 3.2 An Artificial Neuron [4]

This modified potential is analogous to the post-synaptic potential seen in the dendritic membrane of a biological neuron. The weighted signals from all of the input lines enter the threshold logic fragment where they are summed to produce activation potential for the artificial neuron. This activation potential continues to build until it reaches some predetermined threshold voltage (as θ in Figure 3.2), at which time it sends out a pulse along its axon to the next set of neurons.

The elegance of the back propagation network lies in how the network learns or is trained. A back-propagation network is trained with examples for which a pattern of activities for both the input and output are known. If the network is trying to recognize the behavior of seasonal parameters, it initially guesses a result. The guessed output is then matched with the known answer of a training set, and the difference is determined between the desired and actual output – defined as the prediction error. With this completed, the error is now propagated backward through the network to the input and is minimized by adjusting the various connection strengths, or weights, between the neurons (in Figure 3.2). While the weights can be any positive or negative value, usually randomly chosen small numbers are picked initially. This process is iterated until a global-error minimum is found.

The network is considered to have been trained when this global-error minimum is reached and can now be applied to general problems, like predicting weather. The learning rule applied to back propagation first estimates and then uses the error contributed within each pathway through the network and modifies its response via predetermined weight-change rules. The weight-change rules for the layers closer to the input layer include all of the weights of the more rearward layers (i.e., closest to the output layer).

Because neural networks have the capacity to learn from their environment, they are capable of performing a variety of tasks found to be very difficult or impossible to execute with a conventional digital computer. Like their biological counterparts, they are capable of dealing with non-linearity and can readily handle incomplete, noisy, or even missing data. Neural networks utilize inductive reasoning wherein input and output data are given as training examples and the neural network itself develops the rules. They automatically produce associations based on results of known situations learned during training and adjust or "adapt" themselves to new situations until the new situation is eventually generalized. Computations performed by digital computers are serial, centralized, and synchronous, while neural networks function in a parallel, collective, and asynchronous mode.

3.2 Components of the Neural Network

Neural networks, sometimes referred to as connectionist models, are parallel-distributed models that have several distinguishing features [16]:

1) A set of processing units;
2) An activation state for each unit, which is equivalent to the output of the unit;
3) Connections between the units. Generally each connection is defined by a weight w_{jk} that determines the effect that the signal of unit *j* has on unit *k;*
4) A propagation rule, which determines the effective input of the unit from its external inputs;
5) An activation function, which determines the new level of activation based on the effective input and the current activation;
6) An external input (bias, offset) for each unit;
7) A method for information gathering (learning rule);

8) An environment within which the system can operate, provide input signals and, if necessary, error signals.

The basic processing element of the Neural Network is known as the neuron or node. This processing element corresponds in theory to the individual neurons of the nervous system. The processing element is broken into two components, the internal activation function and the transfer function. Internal activation can be calculated in a number of ways. But it typically operates through a summation type function which adds up the values of incoming messages multiplied by a specific connection weight. The resultant output of the internal activation is sent to the transfer function which determines whether or not the processing element will send an output message.

Processing elements or nodes are usually grouped together in a network of layers for the processing of data. This architecture typically consists of an input layer, to which data to be interpreted is sent, one or more hidden layers, and an output data, onto which the correct interpretation is mapped for later use by the user. Typically, every node in a particular layer receives input from all the nodes in the layer below it and/or sends its output, it makes one, to every node in the layer above. The net may be set up so that competition between the nodes of a particular layer determines that only one node will send a message to the next layer.

Neural Networks must be trained by being shown a series of examples which include samples of potential input values and the appropriate output values. The training data may be sent through the network one time only or over and over again until a calculated error value drops below a certain value. The user may select one of a number of learning rules which adjust the weights of the individual nodes during training until the output of the net matches the proper output in the training examples. It is important to ensure that the training examples given to the net both cover the range of possibilities and is representative of the frequency of occurrences of the data.

Once the network has been trained on the training set to respond with the correct output to a variety of inputs, then the net can be fed the full data set for interpretation. The network will go through the complete data set only once, imputing each set of

data and providing the appropriate output. Neural Networks can be trained to interpret data in a variety of forms including, categorical data, discrete data or continuous data. Neural networks are able to handle data, which is noisy or incomplete.

Neural networks are computational models that consist of a number of simple processing units that communicate by sending signals to each other over a large number of weighted connections. The processing units transport incoming information on their outgoing connections to other units. The "electrical" information is simulated with specific values stored in those weights that make these networks have the capacity to learn, memorize, and create relationships amongst data.

A very important feature of these networks is their adaptive nature where "learning by example" replaces "programming" in solving problems. This feature renders these computational models very appealing in application domains where one has little or incomplete understanding of the problems to be solved, but where training data are available.

3.2.1 Processing Unit

A processing unit (Figure 3.3), also called a neuron or node, performs a relatively simple job; it receives inputs from neighbors or external sources and uses them to compute an output signal that is propagated to other units.

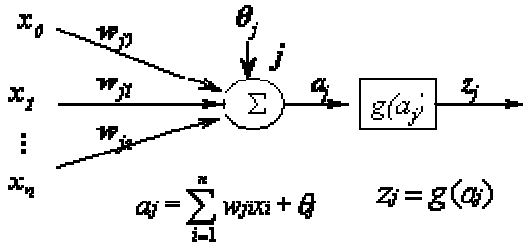

Figure 3.3 Processing unit

Within the neural systems there are three types of units such as (a) Input units, which receive data from outside of the network, (b) Output units, which send data out of the network, (c) Hidden units, whose input and output signals remain within the network [16].

Each unit *j* can have one or more inputs $x_0, x_1, x_2, \ldots x_n$, but only one output z_j. An input to a unit is either the data from outside of the network, or the output of another unit, or its own output.

3.2.2 Combination Function

Each non-input unit in a neural network combines values that are fed into it via synaptic connections from other units, producing a single value called net input. The function that combines values is called the combination function, which is defined by a certain propagation rule. In most neural networks each unit provides an additive contribution to the input of the unit with which it is connected. The total input to unit *j* is simply the weighted sum of the separate outputs from the connected units plus a threshold or bias term θ_j :

$$a_j = \sum_{i=1}^{n} w_{ji} x_i + \theta_j \tag{3.1}$$

The contribution for positive w_{ji} is considered as an excitation and an inhibition for negative w_{ji}. units with the above propagation rule sigma units.

In some cases more complex rules for combining inputs are used. One of the propagation rules known as sigma-pi has the following format:

$$a_j = \sum w_{ji} \prod x_{ik} + \theta_j \quad : i=1 \text{ to } n \ , \ k = 1 \text{ to } m \tag{3.2}$$

Lots of combination functions usually use a "bias" or "threshold" term in computing the net input to the unit. For a linear output unit, a bias term is equivalent to an intercept in a regression model. It is needed in much the same way as the constant polynomial '1' is required for approximation by polynomials [16].

3.2.3 Activation Function

Most units in neural network transform their net inputs by using a scalar-to-scalar function called an activation function, yielding a value called the unit's activation. Except possibly for output units, the activation value is fed to one or more other units. Activation functions with a bounded range are often called squashing functions. Some of the most commonly used activation functions are:

(1) Identity function

$$g(x) = x \tag{3.3}$$

It is obvious that the input units use the identity function. Sometimes a constant is multiplied by the net input to form a linear function.

2) Binary step function

Also known as threshold function or Heaviside function. The output of this function is limited to one of the two values:

$$g(x) = \begin{cases} 1 & \text{if}(x \geq \theta) \\ 0 & \text{if}(x < \theta) \end{cases} \tag{3.4}$$

This kind of function is often used in single layer networks.

3) Sigmoid function

$$g(x) = \frac{1}{1+e^{-x}} \tag{3.5}$$

This function is especially advantageous for use in neural networks trained by back-propagation, because it is easy to differentiate, and thus can dramatically reduce the computation burden for training. It applies to applications whose desired output values are between 0 and 1.

4) Bipolar sigmoid function

$$g(x) = \frac{1-e^{-x}}{1+e^{-x}} \tag{3.6}$$

This function has similar properties with the sigmoid function. It works well for applications that yield output values in the range of [-1,1].

Activation functions for the hidden units are needed to introduce non-linearity into the networks. The reason is that a composition of linear functions is again a linear function. However, it is the non-linearity (i.e., the capability to represent nonlinear functions) that makes multi-layer networks so powerful. Almost any nonlinear function does the job, although for back-propagation learning it must be differentiable and it helps if the function is bounded. The sigmoid functions are the most common

choices . For the output units, activation functions should be chosen to be suited to the distribution of the target values. For binary [0,1] outputs, the sigmoid function is an excellent choice. For continuous-valued targets with a bounded range, the sigmoid functions are again useful, provided that either the outputs or the targets to be scaled to the range of the output activation function. But if the target values have no known bounded range, it is better to use an unbounded activation function, most often the identity function (which amounts to no activation function). If the target values are positive but have no known upper bound, an exponential output activation function can be used.

3.2.4 Objective Function

To train a network and measure how well it performs, an objective function (or cost function) must be defined to provide an unambiguous numerical rating of system performance. Selection of an objective function is very important because the function represents the design goals and decides what training algorithm can be taken. A few basic functions are very commonly used, one of them is the sum of squares error function,

$$E = \frac{1}{NP}\sum_{p=1}^{P}\sum_{i=1}^{N}(t_{pi} - y_{pi})^2 \tag{3.7}$$

where p indexes the patterns in the training set, i indexes the output nodes, and t_{pi} and y_{pi} are, respectively, the target and actual network output for the ith output unit on the pth pattern. In real world applications, it may be necessary to complicate the function with additional terms to control the complexity of the model.

3.3 Network Learning

Neural networks learn the relationship between the input and output information through training. The training process takes the known input and output parameters and adjusts the various weights and thresholds of the network's many neurons until all of the prediction errors of the training set have been minimized. The network when properly trained has discovered the correct relationship between the input and output variables, the neural computer can then be used with different input data and produces a completely new output not known ahead of time. The neural networks can be

trained to predict next week's temperature by exposing the network to pressure and density taken from the past few months' historical data.

The functionality of a neural network is determined by the combination of the topology (number of layers, number of units per layer, and the interconnection pattern between the layers) and the weights of the connections within the network. The topology is usually held fixed, and a certain training algorithm determines the weights. The process of adjusting the weights to make the network learn the relationship between the inputs and targets is called learning, or training. Many learning algorithms have been invented to help find an optimum set of weights those results in the solution of the problems. They can roughly be divided into two main groups:

1) Supervised Learning - The network is trained by providing it with inputs and desired outputs (target values). These input-output pairs are provided by an external teacher, or by the system containing the network. The algorithm to adapt the weights in the network uses the difference between the real outputs and the desired outputs. It is often posed as a function approximation problem - given training data consisting of pairs of input patterns x, and corresponding target t, the goal is to find a function f (x) that matches the desired response for each training input.

2) Unsupervised Learning - With unsupervised learning, there is no feedback from the environment to indicate if the outputs of the network are correct. The network must discover features, regulations, correlations, or categories in the input data automatically. In fact, for most varieties of unsupervised learning, the targets are the same as inputs. In other words, unsupervised learning usually performs the same task as an auto-associative network, compressing the information from the inputs.

3.3.1 Learning Algorithms

There are different variations of the backpropagation training algorithm, each of them having a variety of different computation and storage requirements. The table 3.1 summarizes the training algorithms used in the seeking procedure of the model with the highest level of accuracy.

Table 3.1 Learning Algorithms [31]

Algorithm	**Description**
Gradient Descent (GD)	Slow response, can be used in incremental training mode
Gradient Descent with Momentum (GDM)	Faster training than GD, can be used in incremental training mode
Gradient Descent with Adaptive Learning Rate (GDX)	Faster training than GD, but can only be used in Batch training mode
Resilient Backpropagation (RP)	Simple batch training mode algorithm with fast convergence and minimal storage requirements
Polak-Ribiere Conjugate Gradient (CGP)	Slightly larger storage requirements and faster convergence on some problems
Levenberg-Marquardt (LM)	Faster training algorithm for networks with moderate size, with ability of memory reduction for use when the training data set is large
Bayessian Regularization (BR)	Modification of the Levenberg-Marquardt training algorithm to produce networks that improves generalization and reduces the difficulty of the determination of the optimum network architecture.

3.4 Back-Propagation

Back-propagation is the most commonly used method for training multi-layer feed-forward networks. It can be applied to any feed-forward network with differentiable activation functions. For most networks, the learning process is based on a suitable error function, which is then minimized with respect to the weights and bias. If a network has differential activation functions, then the activations of the output units become differentiable functions of input variables, the weights and bias. If define a differentiable error function of the network outputs such as the sum-of-square error function, then the error function itself is a differentiable function of the weights. Therefore, evaluating the derivative of the error with respect to weights, and these derivatives can then be used to find the weights that minimize the error function, by either using the popular gradient descent or other optimization methods. The

algorithm for evaluating the derivative of the error function is known as back-propagation, because it propagates the errors backward through the network.

3.4.1 Error Function Derivative Calculation

Consider a general feed-forward network with arbitrary differentiable non-linear activation functions and a differential error function.

Each unit *j* is obtained by first forming a weighted sum of its inputs of the form,

$$a_j = \sum w_{ji} z_i \tag{3.8}$$

where z_i is the activation of an unit, or input. Then apply the activation function

$$z_j = g(a_j) \tag{3.9}$$

Note that one or more of the variables z_j in equation (3.8) could be an input, in which case denote it by x_i. Similarly, the unit j in equation (3.9) could be an output unit, denoted by y_k.

The error function will be written as a sum, over all patterns in the training set, of an error defined for each pattern separately,

$$E = \sum E_p \quad E_p = E(Y;W) \tag{3.10}$$

where p indexes the patterns, Y is the vector of outputs, and W is the vector of all weights. E_p can be expressed as a differentiable function of the output variable y_k.

3.4.2 Weight Adjustment with the Gradient Descent Method

Once the derivatives of the error function with respect to weights use them to update the weights so as to decrease the error. There are many varieties of gradient-based optimization algorithms based on these derivatives. One of the simplest of such algorithms is called gradient descent or steepest descent. With this algorithm, the weights are updated in the direction in which the error *E* decreases most rapidly i.e. along negative gradient. The weight updating process begins with an initial guess for weights (which might be chosen randomly) and then generates a sequence of weights using the following formula,

$$\Delta w_{ji}^{(r+1)} = -\eta \frac{\partial E}{\partial w_{ji}} \tag{3.17}$$

where η is a small positive number called the learning rate, which is the step size need to take for the next step. Gradient descent only tells the direction move to, but the step size or learning rate needs to be decided as well. Too low a learning rate makes the network learn very slowly, while too high a learning rate will lead to oscillation. One way to avoid oscillation for large η is to make the weight change dependent on the past weight change by adding a momentum term,

$$\Delta w_{ji}^{(r+1)} = -\eta \frac{\partial E}{\partial w_{ji}} + \alpha \Delta w_{ji}^{(r)} \tag{3.18}$$

That is, the weight change is a combination of a step down the negative gradient, plus a fraction α of the previous weight change, where $0 \leq \alpha < 1$ and typically $0 \leq \alpha < 0.9$ When no momentum term is used, it typically takes a long time before the minimum is reached with a low learning rate (a), whereas for large learning rates the minimum may be never reached because of oscillation (b). When adding a momentum term, the minimum will be reached faster (c).

There are two basic weight-update variations: batch learning and incremental learning. With batch learning, the weights are updated over all the training data. It repeats the following loop: a) Process all the training data; b) Update the weights. Each such loop through the training set is called an epoch. While for incremental learning, the weights are updated for each sample separately. It repeats the following loop: a) Process one sample from the training data, b) Update the weights.

Chapter 4 Quantum Computation in Neural Networks

There has been a growing interest in Artificial neural networks (ANNs) based on quantum theoretical concepts and techniques due to cognitive science and computer science aspects. The so-called Quantum Neural Networks (QNNs) is an exciting area of research in the field of quantum computation and quantum information, which encompasses processing and transmission of data stored in quantum states.

From this scenario, emerge the field of artificial neural networks based on quantum theoretical concepts and techniques. They are called Quantum Neural Networks (QNNs)**.** The basic approach is inspired on the multiple universes view of quantum theory: the neural network is seen as a physical system whose multiple occurrences (component networks) are trained according to the set of patterns of interest. The superposition of the trained components gives the final QNN. The network can be seen as a physical system instantiated from a set of physical parameters. The many dimensional phase space corresponding may have a set of local minima. Each one of these critical points is associated with a particular pattern, which can be said to be stored by the network physical parameters. The advantage of using a quantum net is that the number of stable states can be much larger than the classical counterpart, because of quantum superposition of states and because the connectivity is much more complex.

4.1 Quantum Computation

The most useful model for quantum computation is the Quantum Computational Network also called Deutsch's model, the basic information unit in this model is a qubit, which can be considered a superposition of two independent states $|0\rangle$ and $|1\rangle$ denoted by $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, where α, β are complex numbers such that $|\alpha|^2 + |\beta|^2$ is equal to 1. A composed system with n qubits is described using $N = 2^n$ independent states obtained through the tensor product of the Hilbert Spaces associated with the qubits. Thus, the resulting space has a natural basis that can be denoted $\{|\, i_0\, i_1 \ldots\ldots i_{n-1}\rangle$; $i_j \in \{0, 1\}$. By Quantum Mechanics Postulates, the system state, $|\psi\rangle$ in any time t can be expanded as a superposition of the basis states: $|\psi\rangle = \sum_0^{N-1} \alpha_i |i\rangle$; $\Sigma |\alpha_i|^2 = 1$.

Entanglement is another important concept for quantum computation with no classical counterpart. A composed system with two qubits., the resulting Hilbert Space has $N = 2^2$ independent states. The Hilbert Space associated with the first qubit (indexed by 1) denoted by H_1 and the Hilbert Space associated with the second qubit (indexed by 2) denoted by H_2. The computational basis for these spaces are given by: $\{|0\rangle_1, |1\rangle_1\}$ and $\{|0\rangle_2, |1\rangle_2\}$ respectively.

If qubit 1 is in the state $|\psi\rangle_1 = a_{10}|0\rangle_1 + a_{11}|1\rangle_1$ and

qubit 2 in the state $|\psi\rangle_2 = a_{20}|0\rangle_2 + a_{21}|1\rangle_2$ then the composed system is in the state: $|\psi\rangle = |\psi\rangle_1 \otimes |\psi\rangle_2$ explicitly given

$$|\psi\rangle = \sum_{i,j\in\{0,1\}} a_{1i}a_{2j} |i\rangle_1 \otimes |j\rangle_2.$$

Every state that can be represented by a tensor product $|\psi\rangle_1 \otimes |\psi\rangle_2$ belongs to tensor product space $H_1 \otimes H_2$. However there are some states in $H1 \otimes H_2$ that can not be represented in the form $|\psi\rangle_1 \otimes |\psi\rangle_2$ they are called entangled states.

4.2 Quantum theory

It characterized by how it represents (physical) states, observables, measurements, and dynamics (evolution in time) [15].

4.2.1 States

A state is a complete description of a physical system. In quantum mechanics, a state is a ray in a Hilbert space. A Hilbert space is a vector space over the field of complex numbers C, with vectors denoted by $|\psi\rangle$ (Dirac's ket notation). A ket $|\psi\rangle$ is represented as n × 1 matrix (or a n element vector), where n is the dimension of the Hilbert space, and its corresponding bra $\langle\psi|$ is the transpose conjugate of the ket. It has an inner product $\langle\psi||\varphi\rangle$ (may be seen as matrix multiplication) that maps an ordered pair of vectors to C. A vector with unit norm $\langle\psi|\psi\rangle = 1$ so $e^{i\alpha}|\psi\rangle$ (where $\alpha \in R$) represent the same physical state for all α, a real number, can be seen as an overall phase, which is physically insignificant. New states formed by superposition $a|\varphi\rangle + b|\varphi\rangle$, and the relative phase between the two components are physically significant. That is, whereas $a|\varphi\rangle + b|\varphi\rangle$ and $e^{i\alpha}(a|\varphi\rangle + b|\varphi\rangle)(a|\varphi\rangle + b|\varphi\rangle)$ represent the same physical state, $a|\varphi\rangle + e^{i\alpha} b|\varphi\rangle$ is generally a different physical state.

4.2.2. Observables

An observable is a property of a physical system that in principle can be measured. In quantum mechanics, an observable is a self-adjoint operator (matrix). An operator is a linear map taking vectors to vectors, which can be represented by matrices. For an operator A, A: $|\psi\rangle \rightarrow A|\psi\rangle$, and $A(a|\psi\rangle + b|\varphi\rangle) = aA|\psi\rangle + bA|\varphi\rangle$ And the adjoint of an operator $A^{\uparrow}$ is defined by $\langle \varphi|A\psi\rangle = \langle A^{\uparrow}\varphi|\psi\rangle$, where $|A\psi\rangle$ denotes $A|\psi\rangle$ for all vectors $|\psi\rangle, |\varphi\rangle$ In matrix representation of the adjoint is the transpose conjugate. A is self-adjoint if $A = A^{\uparrow}$. The eigenstates of an observable (eigenvectors of the corresponding self-adjoint matrix) form a complete orthonormal basis in the Hilbert space H.

4.2.3. Measurements

In quantum mechanics, the numerical outcome of the measurement of the observable A is an eigenvalue (An eigenvector of a linear operator A on a vector space is a non-zero vector $|v\rangle$ such that $A|v\rangle = \langle v|v\rangle$, where v is a complex number known as the eigenvalue of A corresponding to $|v\rangle$ of A, and right after the measurement, the quantum state becomes the eigenstate of A corresponding to the measurement result. If the quantum state before measurement is $|\psi\rangle$, then outcome an is obtained with probability

$$\text{Prob}(a_n) = ||P_n|\psi\rangle||^2 = \langle\psi|P_n|\psi\rangle,$$

and the (normalized) quantum state becomes in this case

$$P_n|\psi\rangle \;/\; \langle\psi|P_n|\psi\rangle^{1/2}$$

Which means if the measurement is immediately repeated, the same result would be obtained with probability one.

4.2.4. Dynamics

Time evolution of a quantum state is unitary (unitary transformation can be seen as a rotation in Hilbert space); a self-adjoint operator, called the Hamiltonian of the system, generates it. In the Schrödinger picture of dynamics, the vector (state) describing the system evolves in time according to the Schrödinger equation $d/dt|\psi(t)\rangle = -iH|\psi(t)\rangle$ where H is the Hamiltonian.

4.3 Quantum Concepts

Several necessary ideas that form the basis for the study of quantum computation are briefly reviewed here [26].

4.3.1 Linear superposition

It is closely related to the familiar mathematical principle of linear combination of vectors. Quantum systems are described by a wave function ψ that exists in a Hilbert space. The Hilbert space has a set of states, $|\varphi_i\rangle$ that form a basis, and the system is described by a quantum state $|\psi\rangle$,

$$|\psi\rangle = \sum_i c_i |\varphi_i\rangle$$

$|\psi\rangle$ is said to be in a linear superposition of the basis states $|\varphi_i\rangle$ and in the general case, the coefficients c_i may be complex. Use is made here of the Dirac bracket notation, where the ket $|.\rangle$ is analogous to a column vector, and the bra $\langle .|$ is analogous to the complex conjugate transpose of the ket $|.\rangle$.

In quantum mechanics the Hilbert space and its basis have a physical interpretation, and this leads directly to perhaps the most counterintuitive aspect of the theory. The counter intuition is this -- at the microscopic or quantum level, the state of the system is described by the wave function ψ, that is, as a linear superposition of all basis states (i.e. in some sense the system is in all basis states at once). However, at the macroscopic or classical level the system can be in only a single basis state. For example, at the quantum level an electron can be in a superposition of much different energy; however, in the classical realm this obviously cannot be.

4.3.2 Coherence and decoherence

They are closely related to the idea of linear superposition. A quantum system is said to be coherent if it is in a linear superposition of its basis states. A result of quantum mechanics is that if a system that is in a linear superposition of states interacts in any way with its environment, the superposition is destroyed. This loss of coherence is called decoherence and is governed by the wave function ψ. The coefficients c_i are called probability amplitudes, and $|c_i|^2$ gives the probability of $|\psi\rangle$ collapsing into state $|\varphi_i\rangle$ if it decoherence. Note that the wave function ψ describes a real physical system that must collapse to exactly one basis state. Therefore, the probabilities

governed by the amplitudes c_i must sum to unity. This necessary constraint is expressed as the unitarity condition $\sum_i |c_i|^2 = 1$.

In the Dirac notation, the probability that a quantum state $|\psi\rangle$ will collapse into an eigenstate $|\varphi_i\rangle$ is written $|\langle\varphi_i|\psi\rangle|^2$ and is analogous to the dot product (projection) of two vectors. Consider, for example, a discrete physical variable called spin. The simplest spin system is a two-state system, called a spin-1/2 system, whose basis states are usually represented as $|\uparrow\rangle$ (spin up) and $|\downarrow\rangle$ (spin down). In this simple system the wave function ψ is a distribution over two values (up and down) and a coherent state $|\psi\rangle$ is a linear superposition of $|\uparrow\rangle$ and $|\downarrow\rangle$. One such state might be

$$|\psi\rangle = \frac{2}{\sqrt{5}}|\uparrow\rangle + \frac{1}{\sqrt{5}}|\downarrow\rangle$$

As long as the system maintains its quantum coherence it cannot be said to be either spin up or spin down. It is in some sense both at once. Classically, of course, it must be one or the other, and when this system decoherence the result is, for example, the $|\uparrow\rangle$ state with probability

$$\left|\langle\uparrow|\psi\rangle\right|^2 = \left(\frac{2}{\sqrt{5}}\right)^2 = 0.8$$

A simple two-state quantum system, such as the spin-1/2 system just introduced, is used as the basic unit of quantum computation. Such a system is referred to as a quantum bit or qubit, and renaming the two states $|0\rangle$ and $|1\rangle$ it is easy to see why this is so.

4.3.3 Operators

Operators on a Hilbert space describe how one wave function is changed into another. Here they will be denoted by a capital letter with a hat, such as $\hat{A}$, and they may be represented as matrices acting on vectors. Using operators, an eigenvalue equation can be written $\hat{A}|\varphi_i\rangle = a_i|\varphi_i\rangle$, where a_i is the eigenvalue. The solutions $|\varphi_i\rangle$ to such an equation are called eigenstates and can be used to construct the basis of a Hilbert space as discussed previously. In the quantum formalism, all properties are represented as operators whose eigenstates are the basis for the Hilbert space associated with that property and whose eigenvalues are the quantum allowed values for that property. It is important to note that operators in quantum mechanics must be linear operators and further that they must be unitary so that $\hat{A}^\dagger\hat{A} = \hat{A}\hat{A}^\dagger = \hat{I}$, $\hat{I}$ is the identity operator, and $\hat{A}^\dagger$ is the complex conjugate transpose, or adjoint of $\hat{A}$.

4.3.4. Interference

It is a familiar wave phenomenon. Wave peaks that are in phase interfering constructively (magnify each other's amplitude) while those that are out of phase interfere destructively (decrease or eliminate each other's amplitude). This is a phenomenon common to all kinds of wave mechanics from water waves to optics. The well-known double slit experiment demonstrates empirically that at the quantum level interference also applies to the probability waves of quantum mechanics. As a simple example, suppose that the wave function described above is represented in vector form as $|\psi\rangle = \frac{1}{\sqrt{5}}\begin{pmatrix}2\\1\end{pmatrix}$

and suppose that it is operated upon by an operator described by the following matrix,

$$\hat{O} = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\1 & -1\end{bmatrix}$$

The result is

$$\hat{O}|\psi\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\1 & -1\end{bmatrix}\frac{1}{\sqrt{5}}\begin{pmatrix}2\\1\end{pmatrix} = \frac{1}{\sqrt{10}}\begin{pmatrix}3\\1\end{pmatrix}$$

and therefore now

$$|\psi\rangle = \frac{3}{\sqrt{10}}|\uparrow\rangle + \frac{1}{\sqrt{10}}|\downarrow\rangle$$

Notice that the amplitude of the $|\uparrow\rangle$ state has increased while the amplitude of the $|\downarrow\rangle$ state has decreased. This is due to the wave function interfering with itself through the action of the operator -- the different parts of the wave function interfere constructively or destructively according to their relative phases just like any other kind of wave.

4.3.5 Entanglement

It is the potential for quantum states to exhibit correlations that cannot be accounted for classically. From a computational standpoint, entanglement seems intuitive enough -- it is simply the fact that correlations can exist between different qubits -- for example if one qubit is in the $|1\rangle$ state, another will be in the $|1\rangle$ state. However, from a physical standpoint, entanglement is little understood. The questions of what exactly it is and how it works are still not resolved. What makes it so powerful (and so little understood) is the fact that since quantum states exist as superposition's, these

correlations exist in superposition as well. When the superposition is destroyed, the proper correlation is somehow communicated between the qubits, and it is this "communication" that is the crux of entanglement. Mathematically, entanglement may be described using the density matrix formalism. The density matrix ρ_ψ of a quantum state $|\psi\rangle$ is defined as

$$\rho_\psi = |\psi\rangle\langle\psi|$$

For example, the quantum state

$$|\xi\rangle = \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|01\rangle$$

appears in vector form as

$$|\xi\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\1\\0\\0\end{pmatrix}$$

and it may also be represented as the density matrix

$$\rho_\xi = |\xi\rangle\langle\xi| = \frac{1}{2}\begin{pmatrix}1&1&0&0\\1&1&0&0\\0&0&0&0\\0&0&0&0\end{pmatrix}$$

while the state

$$|\psi\rangle = \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$$

is represented as

$$\rho_\psi = |\psi\rangle\langle\psi| = \frac{1}{2}\begin{pmatrix}1&0&0&1\\0&0&0&0\\0&0&0&0\\1&0&0&1\end{pmatrix}$$

and the state

$$|\zeta\rangle = \frac{1}{\sqrt{3}}|00\rangle + \frac{1}{\sqrt{3}}|01\rangle + \frac{1}{\sqrt{3}}|11\rangle$$

is represented as

$$\rho_\zeta = |\zeta\rangle\langle\zeta| = \frac{1}{3}\begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 \end{pmatrix}$$

where the matrices and vectors are indexed by the state labels 00, ..., 11. Now, notice that can be factorized as ρ_ξ

$$\rho_\xi = \frac{1}{\sqrt{2}}\left(\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \otimes \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}\right)$$

where $\otimes$ is the normal tensor product. On the other hand ρ_ψ cannot be factorized. States that cannot be factorized are said to be entangled, while those that can be factorized are not. Notice that $\rho\xi$ can be partially factorized two different ways, one of which is

$$\rho_\zeta = \frac{1}{\sqrt{3}}\left(\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \otimes \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}\right)$$

(the other contains the factorization of ρ_ξ and a different remainder) however, in both cases the factorization is not complete. Therefore ρ_ξ, is also entangled, but not to the same degree as ρ_ψ (because ρ_ξ can be partially factorized but ρ_ψ cannot) [26]. It is interesting to note from a computational standpoint that quantum states that are superpositions of only basis states that are maximally far apart in terms of hamming distance are those states with the greatest entanglement.

Chapter 5 Problem description

Quantum neural networks have its origin in arguments for the essential role which quantum processes play in the living brain. For example, new physics binding quantum phenomena with general relativity can explain such mental abilities as understanding, awareness and consciousness. Secondly, the quantum domain by diverse combination of that field with the promising new field of quantum computing which is the generalized field of classical artificial neural networks [3].

Both considerations suggest new understanding of mind and brain function as well as new unprecedented abilities in information processing. Consider quantum neural networks as the next natural step in the evolution of neurocomputing systems, focusing attention on artificial rather than biological systems. Different approaches to the realization of quantum distribute processing and argue that, as in the case of quantum computing has been considered in this research work. A Quantum Neural Network (QNN) with features that make it easy to model, such as to:

- use known quantum algorithms and gates
- have weights measured for each node
- work in simulations of reasonable size
- be able to transfer knowledge to classical systems [26]

A QNN that operates much like a classical ANN composed of several layers – an input layer, one or more hidden layers and an output layer. Each layer is fully connected to the previous layer. Each hidden layer computes a weighted sum of the outputs of the previous layer. If this is sum above a threshold, the node goes high, otherwise it stays low. The output layer does the same thing as the hidden layer(s), except that it also checks its accuracy against the target output of the network. The network as a whole computes a function by checking which output bit is high.

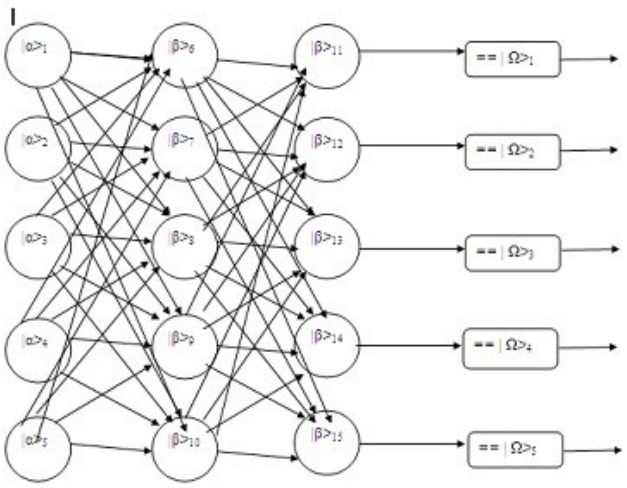

Figure 5.1: Quantum Neural Network

The Quantum Neural Network in Figure 5.1 is an example of such a network in which each input node i is represented by $|\alpha\rangle_i$ and hidden nodes compute a weighted sum of the inputs and compare the sum to a threshold weight $|\psi\rangle_{i0}$. If the weighted sum is greater than the threshold the node goes high. The $|\beta\rangle_k$ represents internal calculations that take place at each node. The output layer works similarly, taking a weighted sum of the hidden nodes and checking against a threshold. The QNN then check each computed output and compare it to the target output, $|\Omega\rangle_j$ sending $|\psi\rangle_j$ high when they are equivalent. The performance of the network is denoted by $|\rho\rangle_{i,}$ which is the number of computed outputs equivalent to their corresponding target output.

The overall network works as follows on a training set, the network has five input parameters, so all n training examples will have five input parameters. Each hidden or output node has a weight vector, represented by $|\psi\rangle$i, each vector containing weights for each of its inputs. After training increment $|\rho\rangle$i by the sum of all $|\psi\rangle$i. When all training examples have been tested, $|\rho\rangle$ will be the sum of the output nodes have the correct answer throughout the training set and will range between zero and the number of training examples times the number of output nodes.

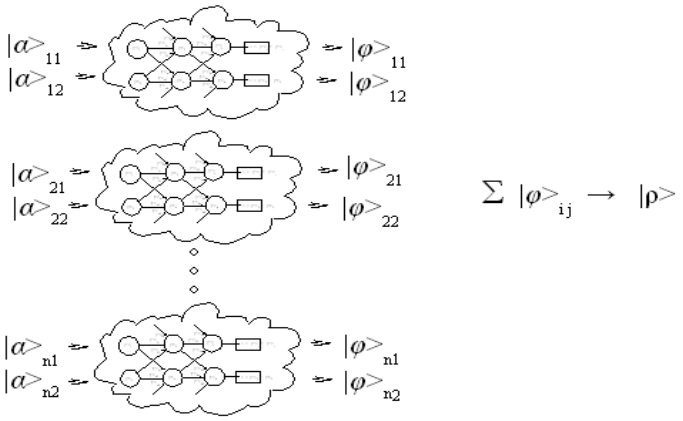

Figure5.2: QNN Trainingf

5.1 State Spaces and Bra/Ket notation

Quantum state spaces and the transformations acting on them can be described in terms of vectors and matrices or in the more compact bra/ket notation invented by Dirac. Kets like $|x\rangle$ denote column vectors and are typically used to describe quantum states. The matching bra, $\langle x|$, denotes the conjugate transpose of $|x\rangle$. For example, the orthonormal basis $\{|0\rangle,|1\rangle\}$ can be expressed as $\{(1, 0)^T, (0, 1)^T\}$. Any complex linear combination of $|0\rangle$ and $|1\rangle$, $a|0\rangle + b|1\rangle$, can be written $(a, b)^T$. The choice of the order of the basis vectors is arbitrary. For example, representing $|0\rangle$ as $(0, 1)^T$ and $|1\rangle$ as $(1, 0)^T$ would be fine as long as this is done consistently. Combining $\langle x|$ and $|y\rangle$ as in $\langle x||y\rangle$, also written as $\langle x|y\rangle$, denotes the inner product of the two vectors. For instance, since $|0\rangle$ is a unit vector have $\langle 0|0\rangle = 1$ and since $|0\rangle$ and $|1\rangle$ are orthogonal $\langle 0|1\rangle = 0$. The notation $|x\rangle\langle y|$ is the outer product of $|x\rangle$ and $\langle y|$. Such as, $|0\rangle\langle 1|$ is the transformation that maps $|1\rangle$ to $|0\rangle$ and $|0\rangle$ to $(0, 0)^T$, since

$$|0\rangle\langle 1||1\rangle = |0\rangle\langle 1|1\rangle = |0\rangle$$

$$|0\rangle\langle 1||0\rangle = |0\rangle\langle 1|0\rangle = 0|0\rangle$$

Equivalently, $|0\rangle\langle 1|$ can be written in matrix form, where $|0\rangle = (1, 0)^T$, $\langle 0| = (1, 0)$, $|1\rangle = (0, 1)^T$, and $\langle 1| = (0, 1)$.

Then

$$|0\rangle\langle 1| = \begin{pmatrix} 1 \\ 0 \end{pmatrix} (0, 1) = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$$

This notation gives a convenient way of specifying transformations on quantum states in terms of what happens to the basis vectors the transformation that exchanges $|0\rangle$ and $|1\rangle$ is given by the matrix.

$$X = |0\rangle\langle 1| + |1\rangle\langle 0|$$

The slightly more intuitive notation, which explicitly specifies the result of a transformation on the basis vectors

$X: |0\rangle \rightarrow |1\rangle,$

$|1\rangle \rightarrow |0\rangle,$

which specifies the result of a transformation on the basis vectors [5].

5.2 Mapping a Number into Quantum State

To represent data from real world problem which generally in real number and binary data to quantum state, and define the mapping function described in equation for binary number and real number respectively, where b is binary and r is real number.

$$f(\theta_b) = \cos(\theta_b) + i\sin(\theta_b)$$

Where $\theta_b = \Pi / 2$, and b is binary value.

$$f(\theta_r) = \cos(\theta_r) + i\sin(\theta_r)$$

Where $\theta r = (\Pi /2).sig(r)$; $sig(r) = 1/(1+e^{-r})$ and r is real value [29].

Chapter 6 Solution of the Problem

To date, the artificial neural networks have been receiving more attention over other algorithms to handle the weather forecast. Determining the properties of the weather is very difficult and it normally requires complex analysis. The main attraction of artificial neural network is that it simplifies this process through machine learning and has been proven to produce weather forecast at superior quality to conventional regression based approach. Perhaps the greatest advantage of neural networks is their ability to be used as an arbitrary function approximation mechanism, which 'learns' from, observed data. However, using them is not so straightforward and a relatively good understanding of the underlying theory is essential.

The neural network can define the result in the classical form, but in the quantum neural network by using the complex valued function can define the result in the approximation or can say that find the result in the accurate form because quantum neural network is the generalized form of the classical neural network. Therefore, the quantum neural network are used to define the performance of weather forecasting parameters like as temperature, pressure, wind, humidity, dew point etc. The application of the quantum neural network to forecast weather is implemented in the Matlab environment and applies the complex function at the diverse training function.

6.1 Data Preprocessing

Theoretically, neural networks are used to map the raw input data directly to required output data. But in practice, it is nearly always beneficial, sometimes critical to apply pre-processing to the input data before they are fed to a network. Pre-processing can vary from simple filtering (as in time-series data), to complex processes for extracting features from raw data. Since the choice of pre-processing algorithms depends on the application and the nature of the data, the range of possibilities is vast. However, the aims of pre-processing algorithms are often very similar, namely:

1) Transform the data into a form suited to the network inputs - this can often simplify the processing that the network has to perform and lead to faster development times. Such transformations may include:

- Apply a mathematical function (logarithm or square) to an input;
- Encode textual data from a database;
- Scale data so that it has a zero mean and a standard deviation of one;
- Take the Fourier transform of a time-series.

2) Select the most relevant data

3) Minimize the number of inputs to the network - Reducing the dimensionality of the input data and minimizing the number of inputs to the network can simplify the problem.

Prediction essentially involves determining the functional relationship(s) between the input-output variables of a system. Once such relationship is established from the available input-output data, it can be used to estimate outputs corresponding to the new inputs for which output values are not available. The set of such input-output patterns is known as the training set.

The network ability of accurately predicting the output of the novel input vectors is known as the generalization ability and is vital for the satisfactory function approximation performance by the network. To ensure that the network possesses good generalization capability, the available data is divided into two parts known as the training set and test set. The training set is used for network learning and the test set to simultaneously evaluate the generalization ability of the network.

6.2 Training and Validation

Now illustrating the forecasting procedure, A three layered feed forward quantum neural network has been implemented. There are five kinds of parameter (temperature, dew point, humidity, sea level pressure, wind speed) as input vectors are given by distribution curve.

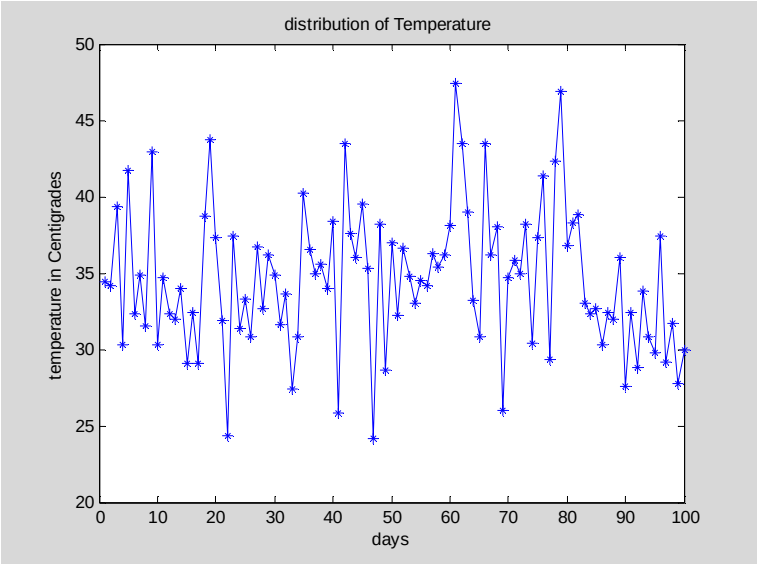

Figure 6.1: Distribution of Temperature in 100 days

The Distribution curve in the Figure 6.1 represents the variation in Temperature over 100 days. Similarly other parameters variation curves can be depicted.

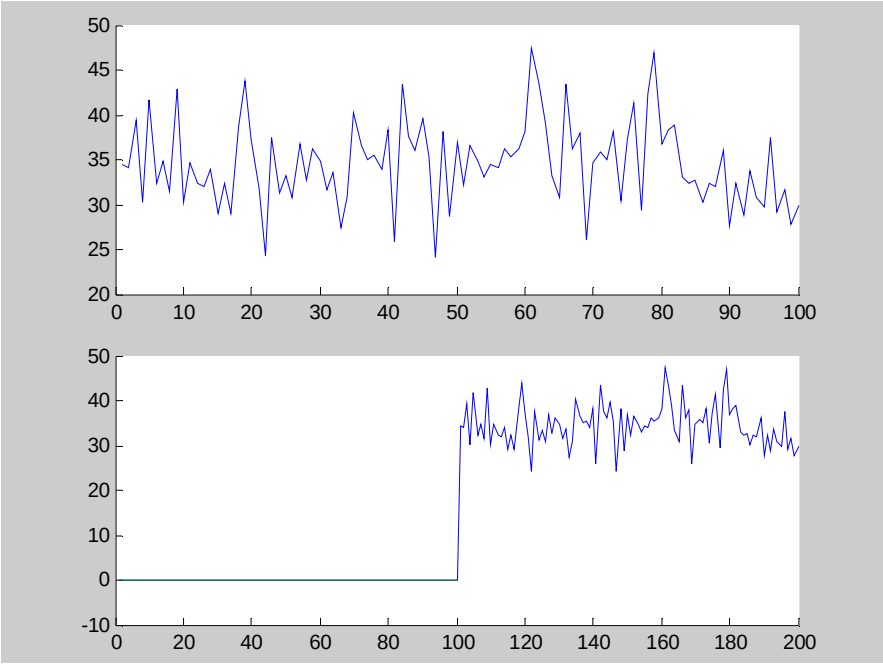

Figure 6.2 Next 100 days forecasting

Training Algorithm:

Step 1: Initialize the inputs.

Step 2: Define the value of theta.

Step 3: Apply the value of theta in the complex function.

$f = \cos\theta + i\sin\theta;$

Step 4: Define the value of the output.(optional)

Step 5: Simulate the network (before the training).

Step 6: Declare the all-training parameters

(a)network.trainParam.show ;

(b)network.trainParam.lr ;

(c)network.trainParam.lr_inc;

(d)network.trainParam.epochs;

(e)network.trainParam.goal

Step 7: Train the network

Step 8: Simulate the network after the training

Step 9: End

Training occurs according to traingdm's training parameters, shown here with their default values:

net.trainParam.epochs 500 Maximum number of epochs to train

net.trainParam.show 5 Epochs between showing progress

net.trainParam.goal 1e-005 Performance goal

net.trainParam.min_grad 1e-6 Minimum performance gradient

net.trainParam.searchFcn Name of line search routine to use.'srchcha'

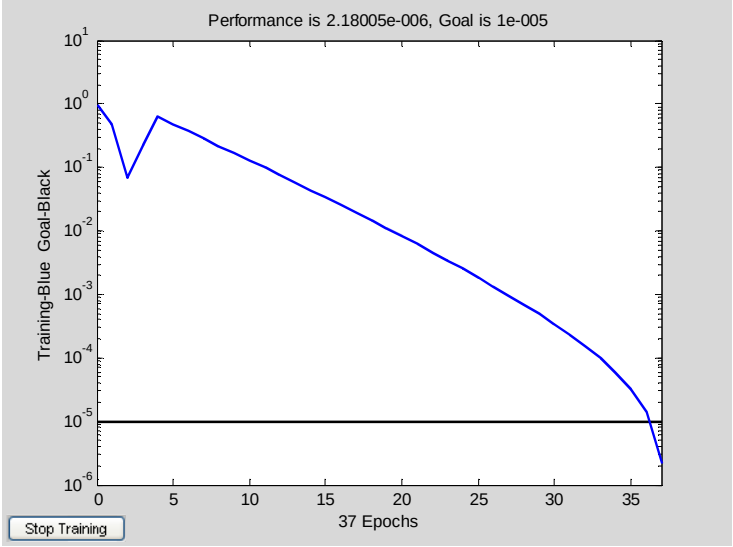

Figure 6.3 Trainng with " TRAINGD"

TRAINGD, Epoch 0/500, MSE 0.987605/1e-005, Gradient 5.61561/1e-010

TRAINGD, Epoch 5/500, MSE 0.478996/1e-005, Gradient 2.50234/1e-010

TRAINGD, Epoch 10/500, MSE 0.129512/1e-005, Gradient 1.5854/1e-010

TRAINGD, Epoch 15/500, MSE 0.0336727/1e-005, Gradient 0.873511/1e-010

TRAINGD, Epoch 20/500, MSE 0.00824659/1e-005, Gradient 0.467435/1e-010

TRAINGD, Epoch 25/500, MSE 0.00183657/1e-005, Gradient 0.245997/1e-010

TRAINGD, Epoch 30/500, MSE 0.000338369/1e-005, Gradient 0.127616/1e-010

TRAINGD, Epoch 35/500, MSE 3.23782e-005/1e-005, Gradient 0.0652184/1e-010

TRAINGD, Epoch 37/500, MSE 2.18005e-006/1e-005, Gradient 0.0496293/1e-010

TRAINGD, Performance goal met.

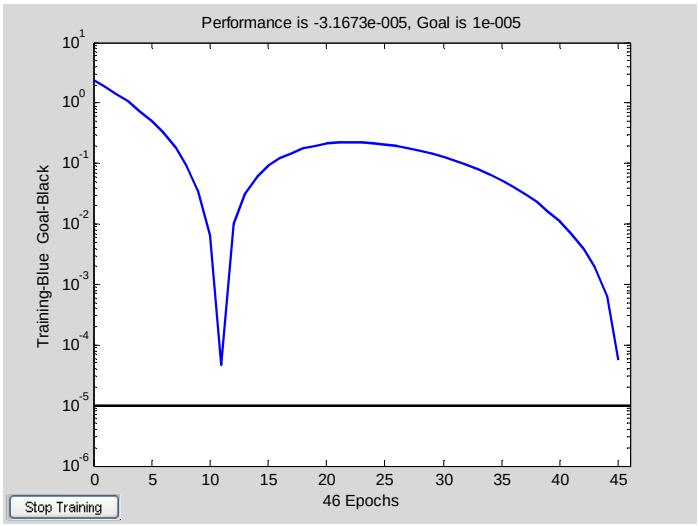

Figure 6.4 training with "TARINGDM"

TRAINGDM, Epoch 0/500, MSE 2.34786/1e-005, Gradient 6.82245/1e-010

TRAINGDM, Epoch 5/500, MSE 0.506308/1e-005, Gradient 3.11321/1e-010

TRAINGDM, Epoch 10/500, MSE 0.00654371/1e-005, Gradient 0.356551/1e-010

TRAINGDM, Epoch 15/500, MSE 0.08996/1e-005, Gradient 1.30839/1e-010

TRAINGDM, Epoch 20/500, MSE 0.211055/1e-005, Gradient 2.00904/1e-010

TRAINGDM, Epoch 25/500, MSE 0.208679/1e-005, Gradient 2.00229/1e-010

TRAINGDM, Epoch 30/500, MSE 0.129975/1e-005, Gradient 1.58316/1e-010

TRAINGDM, Epoch 35/500, MSE 0.0526521/1e-005, Gradient 1.0096/1e-010

TRAINGDM, Epoch 40/500, MSE 0.0110206/1e-005, Gradient 0.463757/1e-010

TRAINGDM, Epoch 45/500, MSE 5.60384e-005/1e-005, Gradient 0.0463542/1e-010

TRAINGDM, Epoch 46/500, MSE -3.1673e-005/1e-005, Gradient 0.0178602/1e-010

TRAINGDM, Performance goal met.

Table 6.1 Five parameters ‘s one month reading.

Temp. (°C)			Dew Point (°C)			Humidity (%)			Sea Level Pressure (hPa)			Visibility (km)			Wind (km/h	
high	avg	Low	high	avg	low	high	avg	low	high	avg	low	high	avg	low	high	avg
26	21	16	18	15	13	88	75	61	1009	1008	1007	6	5	4	19	5
30	22	13	21	15	11	88	69	43	1009	1008	1007	6	5	3	13	2
26	21	15	20	16	12	88	69	42	1011	1009	1007	6	5	4	14	8
28	22	15	17	15	12	88	61	37	1012	1011	1009	6	5	3	26	10
28	22	16	17	14	12	83	64	51	1012	1010	1009	6	5	4	11	10
23	18	13	18	16	12	88	82	73	1010	1009	1009	6	5	4	19	11
27	21	13	17	14	12	94	69	39	1012	1009	1006	6	5	4	11	6
22	20	16	18	16	15	88	83	73	1011	1007	1004	6	5	4	26	10
27	21	13	17	13	12	88	67	39	1016	1014	1011	6	5	4	130	14
30	22	15	17	14	12	83	61	37	1015	1013	1011	8	5	4	26	10
30	23	15	17	15	12	77	59	38	1012	1010	1008	6	5	3	23	8
30	23	15	17	15	12	77	59	38	1012	1010	1008	6	5	3	23	8
35	26	17	17	14	12	73	49	29	1008	1008	1005	6	5	4	24	5
35	26	17	20	15	15	88	60	30	1006	1005	1004	6	5	3	11	6
33	25	17	17	15	12	78	58	26	1008	1007	1005	6	5	3	26	18
35	26	17	17	16	15	88	59	32	1011	1009	1006	6	5	4	37	6
35	26	17	16	15	12	78	51	30	1008	1007	1004	6	5	4	23	10
35	26	17	18	13	11	64	46	23	1007	1006	1005	8	5	3	29	13
35	26	17	17	14	11	83	48	22	1007	1006	1004	6	5	4	29	8
36	28	21	21	15	11	65	46	25	1007	1003	1001	6	4	4	32	10
37	27	17	17	15	12	83	54	22	1004	1002	999	8	5	3	26	18
35	26	18	20	16	15	83	56	38	1007	1005	1003	6	5	3	14	10
37	28	20	17	15	12	78	47	25	1011	1009	1008	8	5	4	26	10
33	25	17	18	15	12	83	55	28	1011	1010	1009	8	5	4	23	8
36	30	23	18	14	11	51	40	27	1013	1011	1009	8	6	4	19	8
37	28	20	18	16	15	83	46	27	1011	1010	1008	6	5	3	19	10
40	30	20	17	11	3	53	35	10	1010	1008	1005	8	5	3	11	2
41	31	20	18	13	2	73	44	8	1007	1006	1004	7	5	3	29	6
40	30	20	17	15	13	78	39	24	1008	1007	1005	5	4	3	11	5
41	31	20	17	11	0	73	35	8	1008	1006	1004	7	5	4	26	5

Chapter 7 Conclusions and Future Work

Quantum Neural Network trained to approximate the desired function by using example data that is representative of the desired task, which are capable of solving complex problem based on the presentation of a large number of training data. Quantum Neural Network estimates a function without mathematical description of how the outputs functionally depend on the inputs. In this research work, a weather forecasting system is implemented using Quantum neural network models in MATLAB environment. The required parameters such as temperature, dew point, humidity, sea level pressure, wind speed etc. are possible candidates to be used as inputs for neural net training and forecasting have been incorporated as influencing variables.

From the simulation results, it is observed that the proposed model produces a reasonable accuracy in training which is conducted with 5 parameters (100 days) of the recent temperature, dew point, humidity, sea level pressure, wind speed with historical data. The model is trained and used to forecast the weather data for next time period, each time forecasting one point ahead. A total of 100 patterns are divided into the training and test sets consisting of 75 and 25 patterns respectively. The network is trained using the error-backpropagation (EBP) algorithm with transfer functions like 'traingd' and 'traingdm'. The network training consisted of 500 iterations (sweeps) through the training set. It is observed that the learning function 'traingd' converge fastly as compared to the 'traingdm'.

Future Work

This system can be scaled for all branches of climate or space weather in an area by incorporating historical data from these branches. Such a system will help for proper and efficient forecasting and management. Further, since the Neural Networks simply interpolate among the training data, it will give high errors with the test data that is not close enough to any one of the training data. This accuracy can further be improved if we take more qualitative data as input, which is large enough to incorporate all the effects, which can be quantified algorithms, the forecasting will become more accurate and reliable.

References

[1] Jean Coiffier, "Weather Forecasting Technique Considered as a Sequence of Standard Processes from the Forecast's point of View", WMO consultant report Geneva, November 2004.

[2] Fausett L. "Fundamentals of Neural Networks", Prentice-Hall, Englewood Cliffs, NJ, 1994.

[3] Haykin, S., "Neural Networks: A Comprehensive Foundation", Prentice Hall, Upper Saddle River, NJ, 1999.

[4] Philip I. Moynihan, "Synthetic Sentience: the Emergence of Computer Awareness Through Evolving Complexity in Nanotechnology", Columbia Commonwealth University 2002, available at http://www.ccwu.edu/Thesis_Moynihan/.

[5] Eleanor Rieffel and Wolfgang Polak "An Introduction to Quantum Computing for Non- Physicists" ACM Computing Surveys (CSUR) Volume 32 , Issue 3 (September 2000) Pages: 300 - 335

[6] Jean Faber , Gilson A. Giraldi, "Quantum models for artificial neural network". Natioanl Laboratory for scientific computing, RJ Brazil. 2004**,** *44* (6), pp 2047–2050

[7] Everett, H. (1957), "Relative state formulation of quantum mechanics", Review of Modern physics, vol.29, pp.454-462.

[8] P A M Dirac, "The principles of quantum mechanics" International Series of Monographs on Physics , Oxford University Press, USA (February 4, 1982)

[9] Mohan Hayati and Zahara Mohebi , "Application of Artificial Neural Networs for Temperature Forecasting", proceedings of World Academy of Science, Engineering and Technolgy volume 22 july 2007 ISSN 1307-6884.

[10] Bennet C., Bernstein E., Brassard G., and Vazirani U., " Strengths and weaknesses of quantum computation" , Special issue on Quantum Computation of the Siam Journal of Computing, Oct. 1997.

[11] Perus M, "Neuro-Quantum parallelism in brain-mind and computers", Informatica, vol. 20, pp.173-183.

[12] Behrman, E.C., Niemel, J., Steck, J.E., and Skinner, S.R. (1996) "A quantum dot neural network". Proceedings of the 4th Workshop on Physics of Computation, Boston, pp.22- 24, November.

[13] Behrman, E.C., Steck, J.E., and Skinner, S.R. (1999) "A spatial quantum neuralcomputer", Proceedings of the International Joint Conference on Neural Networks, to appear.

[14]Goertzel,B.” Quantum Neural Networks” at http://goertzel/org/ben/quantnet.html.

[15] Jyh Ying Peng, “Quantum Computation Lecture Notes” , 2003.

[16] Paras, Sanjay Mathur, Avinash Kumar and Mahesh Chandra “A Feature Based Neural Networks for Weather Forecasting”, proceedings of World Academy of Science, Engineering and Technolgy volume 23 august 2007 ISSN 1307-6884

[17] G. Bonnell I and G. Papini , “Quantum Neural Network” , International Journal of Theoretical Physics, vol. 36, issue 12, pp. 2855-2875.

[18] Lexandr Ezhov and Dan Ventura. “Quantum neural networks”. In Ed. N. Kasabov,editor, Future Directions for Intelligent Systems and Information Science. Physica-Verlang, 2000.

[19] E. C. Behrman, V. Chandrasheka, Z.Wank, C. K. Belur, J. E. Steck, and S. R. Skinner,“A Quantum Neural Network Computes Entanglement”. Technical report, 2002,http://xxx.lanl.gov/quantph/ 0202131.

[20] Mutaz M Jafar, Ali Zilouchian “Fundamentals of Neural Networks: Architectures, Algorithms And Applications” Prentice Hall; US ed edition (December 19, 1993)

[21] D. I. Tseles, A. I. Dounis and J. Zisos “Meteorological Parameters Forecasting for Renewable Energy Systems Using Soft Computing Techniques” Technological Educational Institute of Piraeus, Department of Automation, 250, P. Ralli & Thivon Str., Egaleo, 122 44 Greece.

[22] Zhou Rigui, Jiang Nan and Ding Qiulin “Model and Training of QNN with Weight”, Department of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, Jiangsu 210016, China. e-mail:riguizhou@nuaa.edu.cn.

[23] Eric A. Plummer “Time Series Forecasting With Feed-Forword Neural Networks Guidlines And Limitations”, thesis of The University of Wyoming July, 2000.

[24] Kuo, C., and A. Reitsch, “Neural Networks vs. Conventional methods of forecasting,” Journal of Business Forecasting, 14, 1995/1996, 17-22.

[25] Vlatko Vedral, Adriano Barenco, and Artur Ekert. “Quantum networks for elementary arithmetic operations”. In Physical Review A, volume 54 no. 1, pages 147–153, 1996.

[26] Bob Ricks and Dan Ventura, “Training a Quantum Neural Network”, Department of Computer Science, Brigham Young University, Provo UT 84602, Published in NIPS 2003.

[27] Minsky, Marvin L., and Papert, Seymour S., “Perceptrons: An Introduction to Computational Geometry”, MIT Press, Cambridge, MA 1969.

[28] LI Fei Zengh Baoyu, “A Study of Quantum Neural Networks” IEEE International Conference Neural Networks & Signal Processing Nanjing, China, December 14-17, 2003.

[29] Jarernsri. L. Mitrpanont , Ananta Srisuphab, “The Realization of Quantum Complex-Valued Backpropagation Neural Networks in Pattern Recognition Problem”,Proceedings of the 9th International Conference on Neural Information Processing (ICONIP’OZ) , Vol. 1

[30] Kuo, C., and A. Reitsch, “Neural Networks vs. Conventional methods of forecasting,” Journal of Business Forecasting, 14, 1995/1996, 17-22.

[31] Howard Demuth and Mark Beale, “Neural Network Toolbox for Use with MATLAB”.

[32] Sipser M., “ Introduction to the Theory of Computation” . PWS Publishing Company, 1997.

[33] Kitaev A. Yu., “Quantum measurements and the Abelian stabilizer problem”, ECCC Report TR96-003. Los Alamos archive, e-print quant-ph/ 9511026, 1996.

[34] Eric Benjamin, Kenny Huang, Amir Kamil, Jimmy Kittiyachavalit, “Quantum Computability and Complexity and the Limits of Quantum Computation” , University of California, Berkeley, Dec 2003.

[35] Rigui Zhou, "Quantum Gate Network Based on Adiabatic Theorem," ICNC, vol. 3, pp.510-514, 2008 Fourth International Conference on Natural Computation, 2008

Gurwinder Singh, Dr. V.P. Singh,"*QNN models Application for Weather Forecasting in MATLAB Environment".* 7th International Conference on Information Technology: New Generations, April 12-14, 2010, Las Vegas, Nevada, USA [Communicated].