OPTIMIZATION METHODS IN TRAINING NEURAL NETWORKS

by

SARATHA A/P SATHASIVAM

Thesis submitted in fulfilment of the requirements

for the degree of Master of Science

July 2003

ACKNOWLEDGEMENTS

This thesis would not have been made possible without the assistance of many individuals whom I am greatly grateful. It is a great pleasure for me to acknowledge those many people who have influenced my thinking and contributed to my often inadequate knowledge. Following are:

Dr. Zarita Zainuddin, my supervisor, and one whose contributions to the thesis are unsurpassed.

Dr. Yahya Abu Hassan, for bearing a huge responsibility of being my co-supervisor.

Prof. Madya Md. Ataharul Islam, who has been helpful in editing the thesis.

University Science Malaysia (USM) for sponsoring my studies, under the Academic Staff Training Scheme (ASTS).

Prof. Madya Dr. Izani Md. Ismail, Dean of the School of Mathematical Sciences.

Prof. Madya Ahmad Abd. Majid, former Dean of the School of Mathematical Sciences, and other members of the school for providing the stimulating and yet relaxed environment in which this thesis was prepared.

Last, but foremost, I wish to dedicate this thesis to my parents, Mr & Mrs. Sathasivam and my fiance, Mr. Muraly Velavan as some small acknowledgement of their unfailing love and affection.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRAK

Terdapat beberapa teknik pengekstremuman bagi menyelesaikan masalah aljabar linear dan tak linear. Kaedah Newton mempunyai sifat yang dipanggil penamatan kuadratik yang bermaksud ia meminimumkan suatu fungsi kuadratik dalam bilangan lelaran yang terhingga. Walaubagaimanapun, kaedah ini memerlukan pengiraan dan pengstoran terbitan kedua bagi fungsi kuadratik yang terlibat. Apabila bilangan parameter n adalah besar, ianya mungkin tidak praktikal untuk mengira semua terbitan kedua. Hal ini adalah benar bagi rangkaian neural di mana kebanyakan aplikasi praktikal memerlukan beberapa ratus atau ribu pemberat. Bagi masalah-masalah sedemikian, kaedah pengoptimuman yang hanya memerlukan terbitan pertama tetapi masih mempunyai sifat penamatan kuadratik lebih diutamakan.

Dalam Kaedah Penurunan Tercuram (Steepest Descent Method) suatu fungsian dibina yang mana apabila fungsian ini diekstremkan, ianya akan memberi suatu penyelesaian. Fungsi tersebut akan mempunyai sifat-sifat konveks, supaya vektor yang mengekstremumkan fungsi tersebut merupakan penyelesaian bagi masalah aljabar yang dipertimbangkan. Ini bermaksud pencarian untuk vektor yang kecerunan fungsinya ialah sifar dapat dilaksanakan secara lelaran. Suatu kaedah penurunan tercuram khas, yang sesuai bagi penyelesaian masalah aljabar linear ialah Kaedah Kecerunan Konjugat (Conjugate Gradient Method).

Prestasi Kaedah Kecerunan Konjugat umumnya sangat sensitif kepada pembundaran dalam pengiraan yang mungkin menjejaskan sifat saling konjugat. Tesis ini menumpu kepada Kaedah Kecerunan Konjugat Prasyarat (Preconditioned Conjugate Gradient Method). Prasyarat ini melibatkan pengubahsuaian bagi sistem linear sasaran $Ax=b$ melalui pengunaan suatu matriks prasyarat tentu positif M yang berhubung rapat dengan A melalui hubungan

$$M^{-1}Ax = M^{-1}b$$

Pentingnya, matriks koefisian baru ialah $M^{-1}A$. Proses prasyarat ini bertujuan menghasilkan suatu struktur nilai eigen yang lebih berkelompok bagi $M^{-1}A$ dan pengurangan nombor syarat bagi A untuk memperbaiki nisbah penumpuan berkaitan.

Kaedah Kecerunan Konjugat Prasyarat yang ditumpukan dalam tesis ini ialah kaedah-kaedah yang berasal dari teori pengoptimuman iaitu Kaedah Kecerunan Konjugat Prasyarat Fletcher-Reeves (PCGF), Kaedah Kecerunan Konjugat Prasyarat Polak Ribiere (PCGP) dan Kaedah Kecerunan Konjugat Prasyarat Powell Beale (PCGB). Kelakuan kaedah-kaedah latihan ini terhadap beberapa masalah aplikasi dunia sebenar dilaporkan. Dengan yang demikian, ianya menjelaskan penumpuan dan kekukuhan kaedah-kaedah tersebut. Masalah-masalah dunia sebenar yang dipertimbangkan adalah masalah klasifikasi tumbuhan iris, masalah pengelasan jantina ketam, masalah pengecaman muka dan masalah pengekstrakan gula dari pokok Maple. Dengan menggunakan algoritma-algoritma ini, kadar penumpuan dapat diperbaiki dengan banyaknya dengan hanya pertambahan minimum dalam kompleksitinya.

ABSTRACT

There are a number of extremizing techniques to solve linear and nonlinear algebraic problems. Newton's method has a property called quadratic termination, which means that it minimizes a quadratic function exactly in a finite number of iterations. Unfortunately, it requires calculation and storage of the second derivatives of the quadratic function involved. When the number of parameters, n, is large, it may be impractical to compute all the second derivatives. This is especially true for neural networks, where practical applications can require several hundred to many thousands weights. For these particular cases, methods that require only first derivatives but still have quadratic termination are preferred.

In steepest descent methods, we construct a functional which when extremized will deliver a solution. The function will have convexity properties, so that the vector that extremizes the function is the solution of the algebraic problem in question. This means the search for the vector for which the gradient of the function is zero can be done in an iterative fashion. A special steepest descent method which is appropriate for the solution of the linear algebraic problem is the 'Conjugate Gradient Method'.

Performance of the Conjugate Gradient method is generally very sensitive to roundoff in the computations that may destroy the mutual conjugacy property. This thesis concentrates on Preconditioned Conjugate Gradient methods. Preconditioning involves modification of the target linear system $Ax = b$ through application of a positive-definite preconditioner M that is closely related to A through the relation

$$M^{-1}Ax = M^{-1}b$$

Essentially the new coefficient matrix is $M^{-1}A$. Preconditioning aims to produce a more clustered eigenvalues structure for $M^{-1}A$ and/or lower condition number than for A to improve the relevant convergence ratio.

The Preconditioned Conjugate Gradient methods focused in this thesis are Preconditioned Fletcher-Reeves conjugate gradient (PCGF), Preconditioned Polak-Ribiere conjugate gradient (PCGP) and Preconditioned Powell-Beale restart (PCGB) . The behaviour of these training methods on several real life application problems is reported, thereby illuminating convergence and robustness. The real world problems that have been considered include Classification of Iris Plant, Gender Classification of Crabs, Classification of Face Images and Pulping of Sugar Maple problem. By using these algorithms, the convergence rate can be improved immensely with only a minimal increase in the complexity.

CHAPTER 1

INTRODUCTION

1.1 WHAT IS A NEURAL NETWORK

An Artificial Neural Network (ANN) is an information processing paradigm that is inspired by the way biological nervous system, such as the brain, process information. It is composed of a huge number of highly interconnected processing elements (neurons) working in unison to solve specific problems. ANNs, alike people, learn by example. An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process. Learning in biological systems involves adjustment to the synaptic connections that exist between the neurons. This is true for ANNs as well.

Some Other Definitions of a Neural Network includes:

According to the DARPA Neural Network Study (1988),

.....a neural network is a system composed of many simple processing elements operating in parallel whose function is determined by network structure, connection strengths, and the processing performed at computing elements or nodes.

According to Haykin (1994) ,

A neural network is a massively parallel distributed processor that has a natural propensity for storing experiential knowledge and making it available for use. It resembles the brain in two respects:

1. Knowledge is acquired by the network through a learning process.

2. Interneuron connection strengths known as synaptic weights are used to store the knowledge.

ANNs have been applied to an increasing number of real world problems of considerable complexity. Their most important advantage is in matter of solving problems, which are too complex for conventional technologies-problems that do not have an algorithmic solution or for which an algorithmic solution is too complex to be found. Generally, due to their abstraction from the biological brain, ANNs are well suited to problems that people are better at solving it rather than computers. These problems include pattern recognition and forecasting (which acquired the recognition of pattern in data).

1.2 HISTORICAL BACKGROUND

Neural network simulations seem to be a latest development for solving problems that are too complex for conventional technologies (example: problems that do not have an algorithmic solution or for which an algorithmic solution is too complex to be found) and are often well suited to problems that people are good at solving, but for which traditional methods are not. Following an initial period of enthusiasm, the field survived a period of frustration and disrepute. During this period when funding and professional support was minimal, important advances were made by relatively a small number of researchers. These pioneers were able to develop convincing technology, which surpassed the limitations identified by Minsky and Papert (1969). They summed up a general feeling of frustration (against neural networks) among researchers, and it was accepted by most researchers without further analysis. Currently, the neural network field enjoys a resurgence of interest and a corresponding increase in funding.

The history of neural networks that has been illustrated above can be divided into several periods:

1. First Attempts: There were some initial simulations using formal logic. McCulloch and Pitts (1943) developed models of neural networks based on their understanding of neurology. These models made several assumptions about how neurons worked. Their networks were based on simple neurons, which were considered to be binary devices with fixed thresholds. The outcomes of their model were simple logic functions such as "a or b" and "a and b". Another attempt was by using computer simulations. There are two groups of researchers : Farley and Clark(1954) and Rochester, Holland, Haibit and Duda, (1956) use computer simulations. The first group (IBM researchers) maintained closed contact with neuroscientists at McGill University. So whenever their models fail to work, they consulted the neuroscientists. This interaction established a multidisciplinary trend that continues to the present day.

2. Promising & Emerging Technology: Not only was neuroscience influential in the development of neural networks, but psychologists and engineers also played an important role in the progress of neural network simulations. Rosenblatt (1958) stirred considerable interest and activity in the field when he designed and developed the *Perceptron*. The Perceptron had three layers with the middle layer known as the association layer. This system could learn to connect or associate a given input to a random output unit. Another system was the ADALINE (*ADAptive LInear Element*) which was developed in 1960 by Widrow and Hoff (of Stanford University). The ADALINE was an analogue electronic device made from simple components. The method used for learning

was different to the Perceptron, it employed the Least-Mean-Squares (LMS) learning rule.

3. Period of Frustration & Disrepute: In 1969, Minsky and Papert wrote a book in which they generalized the limitations of single layer Perceptrons to multilayer systems. In the book they said: "...our intuitive judgment that the extension (to multilayer systems) is sterile". The significant result of their book was to eliminate funding for research with neural network simulations. The conclusions created various feedbacks from the researchers in the field. Due to this, considerable prejudice against this field was activated.

4. Innovation: Although public interest and available funding were minimal, several researchers continued working to develop neuromorphically based computational methods for problems such as pattern recognition. During this period several paradigms were generated which modern work continues to enhance. Grossberg & Carpenter (1988) influence founded a school of thought, which explores resonating algorithms. They developed the ART (Adaptive Resonance Theory) networks based on biologically plausible models. Kohonen (1988) developed associative techniques independent of each other. Klopf (1972) developed a basis for learning in artificial neurons based on a biological principle for neuronal learning called *heterostasis*. Werbos (1974) developed and used the *backpropagation* learning method, however several years passed before this approach was recognized. Backpropagation nets are probably the most famous and widely applied of the neural networks today. In essence, the backpropagation net is a Perceptron with multiple layers, a different threshold function in the artificial neuron, and a more robust and capable learning rule. Fukushima (1975) developed a step wise trained multilayer neural network for

interpretation of handwritten characters. The original network was published in 1975 and was called the *Cognitron*.

5. Re-Emergence: Progress during the late 1970s and early 1980s was important to the re-emergence on interest in the neural network field. Several factors influenced this movement. For example, comprehensive books and conferences provided a forum for people in diverse fields with specialized technical languages, and the response to conferences and publications was quite positive. The news media picked up on the increased activity and tutorials helped disseminate the technology. Academic programs appeared and courses were introduced at most major Universities (in United States and Europe). Attention is now focused on funding levels throughout Europe, Japan and the United States and as this funding becomes available, several new commercial with applications in industry and financial institutions are emerging.

6. Today: Significant progress has been made in the field of neural networks enough to attract a great deal of attention and fund further research. Advancement beyond current commercial applications appears to be possible, and research is advancing the field on many fronts. Neurally based chips are emerging and applications to complex problems developing. Clearly, today is a period of transition for neural network technology.

1.3 HUMANS AND ARTIFICIAL NEURONS

Much is still unknown about how the brain trains itself to process information. In the human brain, a typical neuron will be collecting signals from others through a host of

fine structures called *dendrites*. The neuron sends out spikes of electrical activity through a long, thin stand known as an *axon*, which splits into thousands of branches.

At the end of each branch, a structure called a *synapse* converts the activity from the axon into electrical effects that inhibit or excite activity from the axon into electrical effects that inhibit or excite activity in the connected neurons. When a neuron receives excitatory input that is sufficiently large compared with its inhibitory input, it sends a spike of electrical activity down its axon. Learning occurs by changing the effectiveness of the synapses so that the influence of one neuron on another changes.

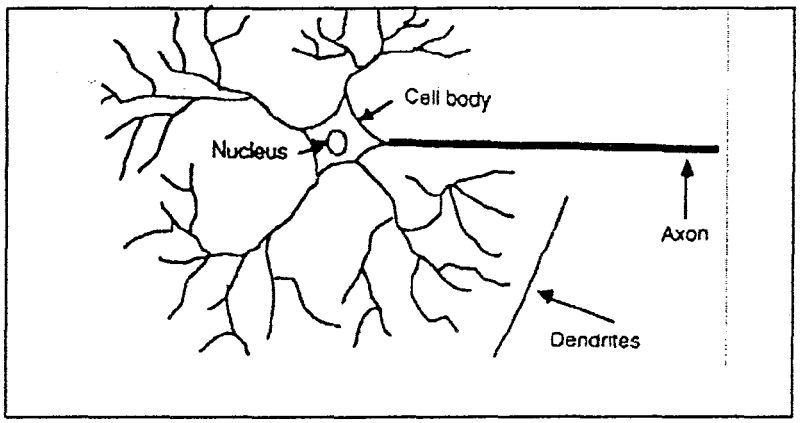

Figure 1.1: Components of a neuron

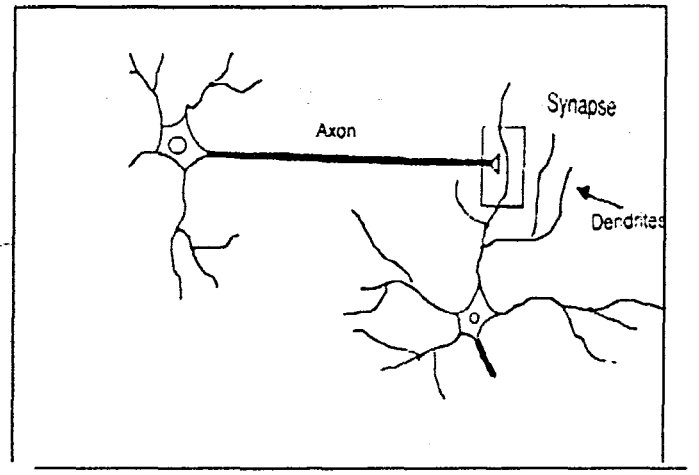

Figure 1.2: Structure of a synapse

We conduct these neural networks by first trying to deduce the essential features of neurons and their interconnections. We then typically program a computer to simulate these features. However because our knowledge of neurons is incomplete and our computing power is limited, our models are necessarily gross idealisations of real networks of neurons.

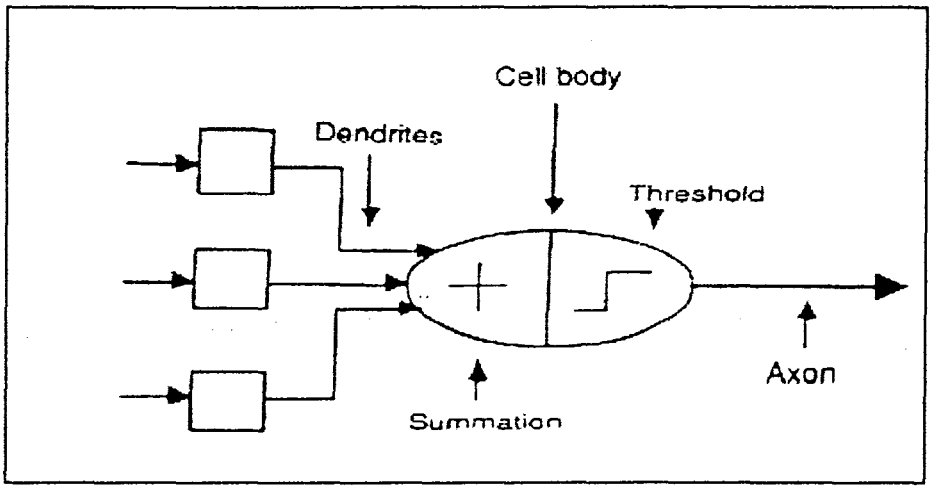

Figure 1.3: The neuron model

1.4 WHY USE NEURAL NETWORKS ?

Remarkable ability and usage of neural networks, derive meaning from complicated or imprecise data, can be used to extract patterns and to detect trends that are too complicated to be notified by either humans or any other computer techniques. A trained neural network can be thought of as an "expert" in the category of information it has been given to analyse. Other advantages include:

1. Adaptive learning: An ability to learn the proper way how to do tasks based on the data provided for training or initial experience.

2. Self-Organisation: An ANN can build its own organisation or representation of the information it gathers during learning period.

3. Real Time Operation: ANN computations might be carried out in parallel, and special hardware devices are being designed and manufactured which take advantage of this capability.

4. Fault Tolerance via Redundant Information Coding: Partial destruction of a network leads to the corresponding degradation of performance. In spite of this, some network capabilities may be retained even with major network damage.

1.5 APPLICATIONS OF NEURAL NETWORKS

Based on description of neural networks and how they work:the major question " what real world applications are they suited for?" has arised. Neural networks have extensive use in real world business problems. In fact, they have already successfully been applied in many industries.

Since neural networks are best in locating patterns or trends in data, they are well suited for prediction or forecasting needs including:

- sales forecasting
- industrial process control
- customer research
- data validation
- risk management

- target marketing

To be more specific, ANN are also used in the following specific paradigms: recognition of speakers in communications, diagnosis of hepatitis, recovery of telecommunications from faulty software, interpretation of multimeaning Chinese words, under sea mine detection, texture analysis, three-dimensional object recognition, hand-written word recognition, and facial recognition.

1.6 THE FUTURE OF NEURAL NETWORKS

Because gazing into the future is somewhat like gazing into a crystal ball, so it is better to quote some "predictions". Each prediction rests on some sort of evidence or established trend, which, with extrapolation, clearly takes us into a new realm.

- Neural Networks will fascinate user-specific systems for education, information processing, and entertainment. Neural networks produced comprehensive environments, are attractive in terms of their potential for systems control, education, and entertainment. This is not just a far-out research trend, but is something that is becoming an increasingly important part of our daily existence, as witnessed by the growing interest in comprehensive "entertainment centers" in each home. This "programming" would require feedback from the user in order to be effective but simple and "passive" sensors (examples are fingertip sensors, gloves, or wristbands to sense pulse, blood pressure, skin ionisation, and others), could provide effective feedback into a neural control system. This could be achieved, for example, with sensors that would detect pulse, blood pressure, skin ionisation, and other variables, which the system could learn to correlate with a person's response state.

- Neural networks, integrated with other artificial intelligence technologies, methods for direct culture of nervous tissue, and other exotic technologies such as genetic engineering, will allow us to develop radical and exotic life-forms whether man, machine, or hybrid.

- Neural networks will allow us to explore new realms of human capability realms previously available only with extensive training and personal discipline. So a specific state of consciously induced neurophysiologically observable awareness is necessary in order to facilitate a man machine system interface.

1.7 GUIDE TO THESIS

The main purpose of this thesis is to propose a new derivative methods- Preconditioned Conjugate Gradient methods (PCG) which will accelerate convergence in neural networks training. The investigation includes analyzing the theoretical part, which explains how PCG clustered the eigenvalues . The performance of PCG methods has been verified by implementing it on some real life problems. In order to evaluate the performance of PCG methods, four real world problems that have been considered includes- Gender Classification of Crabs, Classification of Iris Plant, Human Face Recognition problem and Pulping of Sugar Maple.

Literature survey on neural networks are presented in this chapter. In this chapter, some basic ideas regarding neural networks have been discussed. Basic mathematical and neural networks concepts in training multilayer perceptrons (MLP) with the backpropagation algorithm will be explained in Chapter 2. In Chapter 3, a survey of methods for the optimization of the backpropagation algorithm will be presented. In

Chapter 4, CG and PCG algorithms will be analyzed. Formulation and derivation of the CG and PCG algorithms are mainly focused in this chapter.

The implementation of CG and PCG methods in Neural Network toolbox in Matlab R12 version 6 constitutes Chapter 5. This thesis would not be complete without real world problems. Four real world problems : Gender Classification of Crabs, Classification of Iris Plant, Human Face Recognition problem and Pulping of Sugar Maple have been considered. Firstly a description of the data set is given. Then, numerical results employing the CG and PCG methods on convergence and generalization capabilities are given.

Finally, a discussion and suggestions for future work related to this research will be presented in Chapter 7.

CHAPTER 2

BASIC CONCEPTS AND THEORY OF MULTILAYER PERCEPTRONS

An important class of architecture of neural networks, namely, multilayer feedforward networks is discussed in this chapter. Overall, this network consists of a set of sensory units (source nodes) which constitute the 'input layer', one or more hidden layers of computational nodes. In a forward direction, the input signals propagates through the network, on a layer-by-layer basis. Commonly, these NNs are referred as a multilayer perceptrons (MLPs) , which represent a generalization of the single-layer perceptron.

Multilayer perceptrons have successfully been applied to solve some difficult and diverse problems by training them in a supervised manner with a algorithm known as the error backpropagation (BP) algorithm. BP algorithm is based on the error-correction learning rule and hence, it might be viewed as a generalization of the least-mean-square algorithm for the special case of a single linear neuron model.

There are three distinctive characteristics:

1. The model of each neuron in the network involves a nonlinearty at the output end. In spite of the usage of hard-limiting in Rosenblatt's perceptron, the nonlinearty is smooth that is, differentiable everywhere. Sigmoidal nonlinearty defined by the logistic function is the most commonly used form:

$$y_j = \frac{1}{1+\exp(-v_j(n))}$$

where v_j is defined as the net internal activity level of neuron j and y_j is the output of the neuron. These nonlinearities make the input-output relation of the network different from a single-layer perceptron.

2. The network consists of one or more layers of hidden neurons which are not part of the input or output of the network. These hidden neurons play an important role in providing the network the ability to learn complex tasks by successively extracting meaningful features from the input patterns.

3. The networks display a high degree of connectivity where this criteria is determined by the synapses of the network. A change in the connectivity of the networks requires a change in the population of synaptic weights or their connections.

These three characteristics together with the ability to learn from experience endows the multilayer perceptron's computing power. On the other hand, these same characteristics are also responsible for the deficiencies in the behaviour of the network. Firstly, the presence of a distributed form of nonlinearty and the high connectivity of the network make the theoretical analysis of a multilayer perceptron difficult to undertake. Secondly, the usage of hidden neurons makes the learning process harder to visualize. The learning process must determine which features of the input pattern should be represented by the hidden neurons. The learning process becomes a difficult task due to the search which has to be conducted in a much larger space of possible functions, and a choice has to be made between alternative representations of the input pattern (Hinton,1987).

The development of the backpropagation algorithm represents a 'landmark' in neural networks. It is because it provides a computationally efficient method for the training of multilayer perceptrons. Even though it cannot be claimed that the backpropagation algorithm can provide a solution for all solvable problems, it is fair to say that it has put to rest the pessimism about learning in multilayer machines that may have been inferred from the book by Minsky and Papert (1969).

2.1 DERIVATION OF THE BACKPROPAGATION LEARNING ALGORITHM

Figure 2.1 indicates the architectural graph of a multilayer perceptron with two hidden layers. A neuron in a layer of the network is connected to all the nodes in the previous layer. The signal will be flowing through the network progresses in a forward direction from left to right and on a layer by layer basis. The first layer are the input nodes. The output nodes constitute the output layer while the remaining nodes constitute the hidden layer of the network. The input vector is presented to the input layer and the signals will be propagated forward to the first hidden layer; the resulting output of the first hidden layer are in turn applied to the next hidden layer and so on for the rest of the network. Each output neuron or hidden neuron of a multilayer perceptron is designed to perform two computations:

1. The computation of the function signal appearing at the output of a neuron, which is expressed as a continuous nonlinear function of the input signals and synaptic weights associated with that neuron.

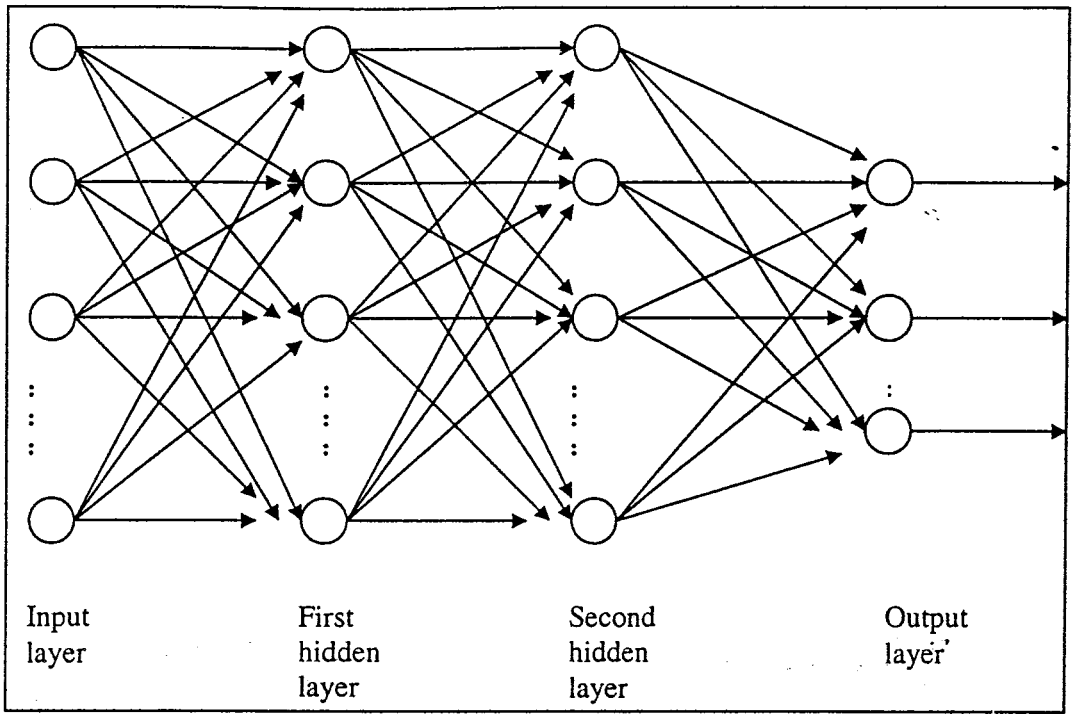

Figure 2.1 Architectural graph of a multilayer perceptron with two hidden layers.

2. The computation of an instantaneous estimate of the gradient vector (example:the gradient of the error surface with respect to the weights connected to the inputs of a neuron), which is needed for the backward pass through the network.

The error signal $e_j(n)$ at the output of neuron j at iteration n (example: presentation of the n th training pattern) is defined by

$$e_j(n) = d_j(n) - y_j(n) \tag{2.1}$$

where $d_j(n)$ and $y_j(n)$ is the desired and the actual response of neuron j at iteration n respectively.

Hence, the instantaneous value $\xi(n)$ of the sum of squared errors over all neurons can be written as

$$\xi(n) = \frac{1}{2}\sum_{j \in C} e_j^2 \ (n) \tag{2.2}$$

where C indicates all the neurons in the output layer of the network. The average squared error over the total number of patterns N is given by

$$\xi_{av} = \frac{1}{N}\sum_{n=1}^{N} \xi(n) \tag{2.3}$$

The main purpose of the learning process is to adjust the free parameters (i.e. synaptic weights and thresholds) of the network so as to minimize ξ_{av}. The gradient descent method is being used to perform the minimization where the weights are updated (adjusted) accordingly to respective errors computed for each pattern to the network. By taking the arithmetic average of these independent weight changes over the training set would therefore depicts the true change that resulted from modifying the weights based on minimizing the cost function ξ_{av} over the training set.

Figure 2.2 depicts neuron j being fed by a set of function signals produced by neurons in the previous layer. The net internal activity level $v_j(n)$ produced at the input of neuron j is

$$v_j(n) = \sum_{i=0}^{p} w_{ji}(n) y_i(n) \tag{2.4}$$

where p is the total number of inputs (excluding the threshold) applied to neuron j and $w_{ji}(n)$ indicates the synaptic weight connecting the output of neuron i to the input

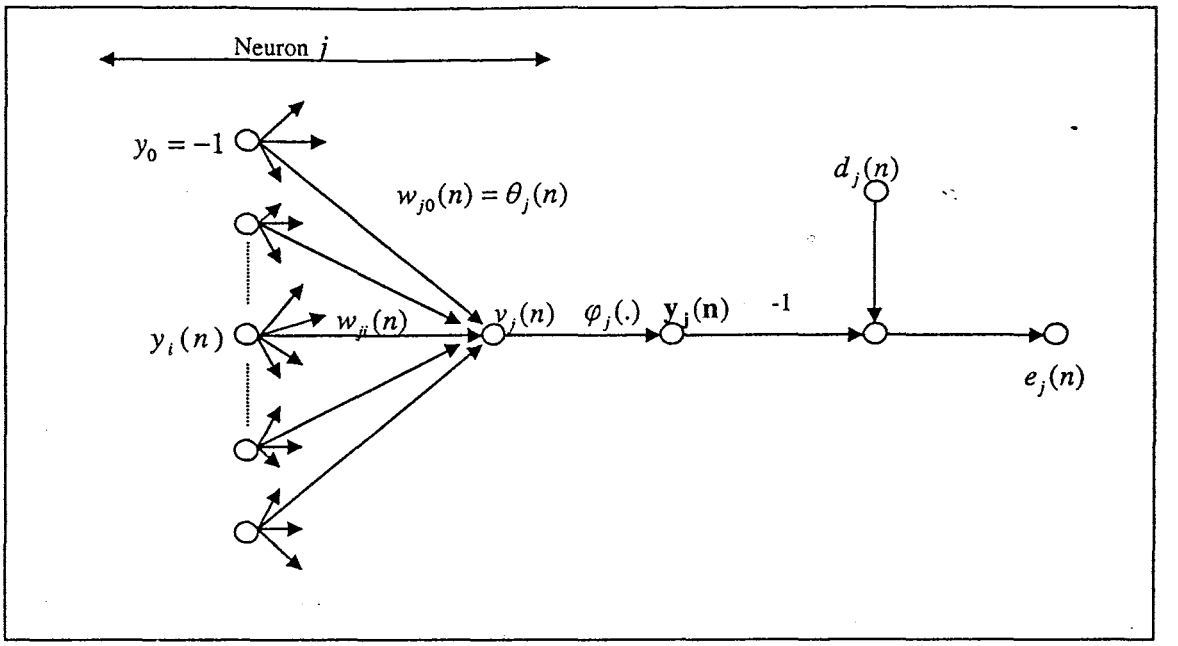

Figure 2.2: Signal-flow graph highlighting the details of output neuron j

of neuron j at iteration n. The synaptic weight w_{j0} (corresponding to the fixed input $y_0 = -1$) equals the threshold θ_j applied to neuron j.

Hence, the function signal $y_j(n)$ appearing at the output of neuron j at iteration n is

$$y_j(n) = \varphi_j(v_j(n)) \tag{2.5}$$

where $\varphi_j(.)$ is the activation function describing the input-output functional relationship of the nonlinearty associated with neuron j.

The correction $\Delta w_{ji}(n)$ applied to the synaptic weight $w_{ji}(n)$ is proportional to the instantaneous gradient $\partial\xi(n)/\partial w_{ji}(n)$. By using the chain rule, the gradient can be expressed as follows:

$$\frac{\partial \xi(n)}{\partial w_{ji}(n)} = \frac{\partial \xi(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} \frac{\partial v_j(n)}{\partial w_{ji}(n)} \tag{2.6}$$

The gradient $\partial \xi(n) / \partial w_{ji}(n)$ determines the direction of search in weight space for the synaptic weight w_{ji}.

By differentiating both sides of Eq. (2.2) with respect to $e_j(n)$, we get

$$\frac{\partial \xi(n)}{\partial e_j(n)} = e_j(n) \tag{2.7}$$

and by differentiating both sides of Eq. (2.1) with respect to $y_j(n)$, we get

$$\frac{\partial e_j(n)}{\partial y_j(n)} = -1 \tag{2.8}$$

By differentiating Eq. (2.5) with respect to $v_j(n)$, we get

$$\frac{\partial y_j(n)}{\partial v_j(n)} = \varphi_j'(v_j(n)) \tag{2.9}$$

Lastly, differentiating Eq. (2.4) with respect to $w_{ji}(n)$ yields

$$\frac{\partial v_j(n)}{\partial w_{ji}(n)} = y_i(n) \tag{2.10}$$

Hence applying Eq. (2.7) to (2.10) in (2.6) yields

$$\frac{\partial \xi(n)}{\partial w_{ji}(n)} = -e_j(n)\varphi_j'(v_j(n))y_i(n) \tag{2.11}$$

Based on the delta rule, the correction $\Delta w_{ji}(n)$ applied to $w_{ji}(n)$ is defined by

$$\Delta w_{ji}(n) = -\eta \frac{\partial \xi(n)}{\partial w_{ji}(n)} \tag{2.12}$$

where η is learning rate parameter.

By using Eq. (2.11) in (2.12) yields

$$\Delta w_{ji}(n) = \eta \delta_j(n) y_i(n) \tag{2.13}$$

where the local gradient $\delta_j(n)$ is itself defined by

$$\delta_j(n) = -\frac{\partial \xi(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} \tag{2.14}$$

$$= e_j(n) \varphi_j'(v_j(n))$$

The local gradient shows the amount of changes required in synaptic weights.

There are two distinct cases associated to the error signal $e_j(n)$ at the output of neuron j:

Case 1: Neuron j is an Output Node.

When neuron j is located in the output layer of the network, it would be supplied with a desired response of its own. Hence,

$$e_j(n) = d_j(n) - y_j(n) \tag{2.15}$$

Case 2: Neuron j is a Hidden Node

When neuron j is located in a hidden layer, there is no exact target response for that neuron. Actually, the error signal for a hidden neural is determined recursively in terms of the error signals of all the neurons to which that hidden neuron is directly connected.

Figure 2.3 shows neuron j as a hidden node of the network. From Eq. (2.14), the local gradient $\delta_j(n)$ for hidden neuron j can also be written as

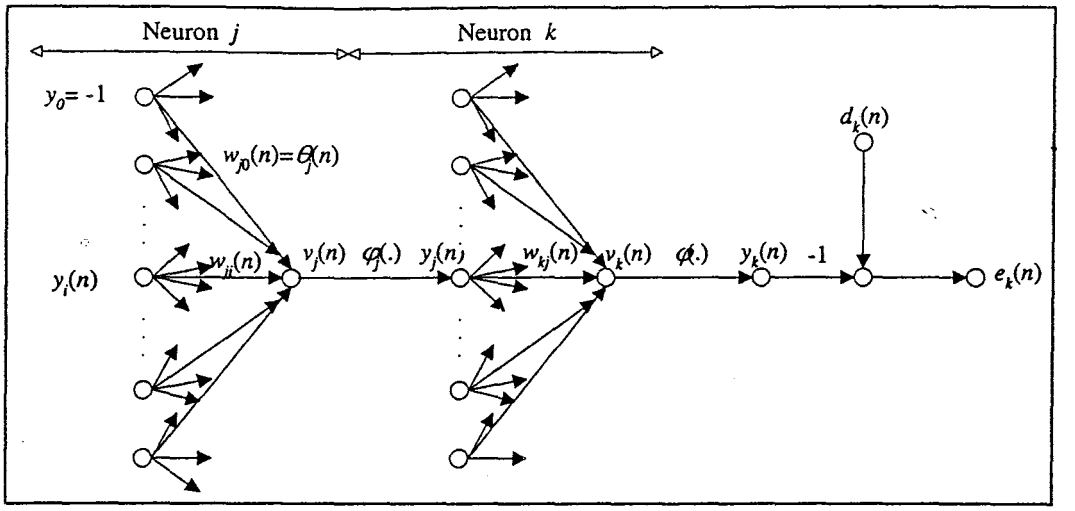

Figure 2.3: Signal-flow graph highlighting the details of output neuron k connected to hidden neuron j.

$$\delta_j(n) = -\frac{\partial \xi(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} \tag{2.16}$$

$$= -\frac{\partial \xi(n)}{\partial y_j(n)} \varphi_j'(v_j(n))$$

The partial derivative $\frac{\partial \xi(n)}{\partial y_j(n)}$ can be calculated as follows. From Fig.2.3, we see that

$$\xi(n) = \frac{1}{2} \sum_{k \in C} e_k^2(n) \tag{2.17}$$

where neuron k is an output node.

By differentiating Eq. (2.17) with respect to the function signal $y_j(n)$, we get

$$\frac{\partial \xi(n)}{\partial y_j(n)} = \sum_k e_k \frac{\partial e_k(n)}{\partial y_j(n)} \tag{2.18}$$

By implementing the chain rule for the partial derivative $\frac{\partial e_k(n)}{\partial y_j(n)}$, Eq. (2.18) can be rewritten in the equivalent form

$$\frac{\partial \xi(n)}{\partial y_j(n)} = \sum_k e_k(n) \frac{\partial e_k(n)}{\partial v_k(n)} \frac{\partial v_k(n)}{\partial y_j(n)} \tag{2.19}$$

From Fig. (2.3),

$$e_k(n) = d_k(n) - y_k(n)$$

$$= d_k(n) - \varphi_k(v_k(n)) \tag{2.20}$$

where neuron k is an output node.

Hence,

$$\frac{\partial e_k(n)}{\partial v_k(n)} = -\varphi_k'(v_k(n)) \tag{2.21}$$

Referring to Figure 2.3, the net internal activity level for neuron k is given by

$$v_k(n) = \sum_{j=0}^{q} w_{kj}(n) y_j(n) \tag{2.22}$$

where q present the total number of inputs (excluding the threshold) applied to neuron k.

By differentiating Eq. (2.22) with respect to $y_j(n)$ yields

$$\frac{\partial v_k(n)}{\partial y_j(n)} = w_{kj}(n) \tag{2.23}$$

After that, by using Eq. (2.21) and (2.23) in (2.19), we get the desired partial derivative:

$$\frac{\partial \xi(n)}{\partial y_j(n)} = -\sum_k e_k(n) \varphi_k'(v_k(n)) w_{kj}(n) \tag{2.24}$$

$$= -\sum_k \delta_k(n) w_{kj}(n)$$

Lastly, by using Eq.(2.24) in (2.16), we obtain the local gradient $\delta_j(n)$ for hidden neuron j, as follows:

$$\delta_j(n) = \varphi_j'(v_j(n)) \sum_k \delta_k(n) w_{kj}(n) \tag{2.25}$$

neuron j is hidden.

2.1.1 THE TWO PASSES OF COMPUTATION

The backpropagation algorithm involves two distinct passes of computation, the forward pass and backward pass. In the forward pass the synaptic weights remain unchanged throughout the network and the function signals are computed on a neuron-by-neuron basis. The output of neuron j, $y_j(n)$, is computed as

$$y_j(n) = \varphi(v_j(n)) \tag{2.26}$$

where $v_j(n)$ is the net internal activity level of neuron j stated as

$$v_j(n) = \sum_{i=0}^{p} w_{ji}\ (n) y_i(n) \tag{2.27}$$

If neuron j is the first hidden layer, then

$$y_i(n) = x_i(n) \tag{2.28}$$

where $x_i(n)$ is the ith element for the input vector. Meanwhile, if neuron j is in the output layer,

$$y_j(n) = o_j(n) \tag{2.29}$$

where $o_j(n)$ is the jth element for the output vector. When this output is been compared with desired response $d_j(n)$, the error signal $e_j(n)$ for the jth output neuron will be obtained. However the forward pass of computation begins at the first hidden layer by presenting it with the input vector, and terminates at the output layer by computing the error signal for each neuron of this layer.

Meanwhile, the backward pass, begins at the output layer by passing the error signals leftward through the network, layer by layer, and recursively computing the δ for each of the neuron. The synaptic weight will be changing according to the delta rule:

$$\Delta w_{ji}(n) = \eta \delta_j(n) y_i(n) \tag{2.30}$$

For a neuron in the output layer, $\delta_j(n)$ is given by

$$\delta_j(n) = e_j(n)\varphi_j'(v_j(n)) \tag{2.31}$$

Given the delta values (δ s) for the neuron of the output layer, the delta values for all the neurons in the penultimate layer is calculated by using:

$$\delta_j(n) = \varphi_j'(v_j(n)) \sum_k \delta_k(n) w_{kj}(n) \tag{2.32}$$

and the changes in the weights of all connections feeding into it are made. This recursive computation is continued on a layer by layer basis. This is done by propagating the changes to all the synaptic weights.

2.1.2 SIGMOIDAL NONLINEARTY

To compute δ for each neuron, the derivative of the activation $\varphi(.)$ is required. This function must be a continuous function. The most common continuously differentiable nonlinear activation function used in multilayer perceptrons is the sigmoidal nonlinearty given by

$$y_j(n) = \varphi_j(v_j(n))$$

$$= \frac{1}{1+\exp(-v_j(n))}, \qquad -\infty < v_j(n) < \infty \tag{2.33}$$

where $v_j(n)$ is the net internal activity of neuron j. The range of this nonlinearty lies inside the interval of $0 \leq y_j \leq 1$. The symmetric *hyperbolic tangent* is another type of sigmoidal nonlinearty. This function is continuous and the output lies inside the range $-1 \leq y_j \leq 1$. One of the advantages of the sigmoid nonlinearty is that it is differentiable which made it possible to derive a gradient search learning algorithm for networks with multiple layers. Furthermore, a multilayer perceptron trained with the backpropagation algorithm learns faster when the sigmoidal activation function is symmetric than when it is nonsymmetric because of better numerical conditioning.

By differentiating both sides of Eq. (2.33) with respect to $v_j(n)$, we obtain

$$\frac{\partial y_j(n)}{\partial v_j(n)} = \varphi_j'(v_j(n)) \tag{2.34}$$

$$= \frac{\exp(-v_j(n))}{[1+\exp(-v_j(n))]^2}$$

By using Eq. (2.33) to eliminate the exponential term $\exp(-v_j(n))$ from Eq. (2.34), we obtain

$$\varphi_j'(v_j(n)) = y_j(n)[1 - y_j(n)] \tag{2.35}$$

There are two distinct cases depending on the location of neuron j.

Case 1: Neuron j is an output node; hence $y_j(n) = o_j(n)$:

$$\delta_j(n) = e_j(n)\varphi_j'(v_j(n)) \tag{2.36}$$

$$= [d_j(n) - o_j(n)]o_j(n)[1 - o_j(n)]$$

Case 2: Neuron j is a hidden node:

$$\delta_j(n) = \varphi_j'(v_j(n))\sum_k \delta_k(n)w_{kj}(n) \tag{2.37}$$