2000

Development of self-adaptive back propagation and derivative free training algorithms in artificial neural networks

Shamsuddin Ahmed
Edith Cowan University

Development Of Self-adaptive Back Propagation And Derivative Free Training Algorithms In Artificial Neural Networks

Shamsuddin Ahmed
Edith Cowan University

Edith Cowan University

Copyright Warning

USE OF THESIS

The Use of Thesis statement is not included in this version of the thesis.

Development of Self-Adaptive Back Propagation and Derivative Free Training Algorithms in Artificial Neural Networks

by

Shamsuddin Ahmed

B.Sc.Engg. Honours (Mechanical Engineering), M.A.Sc (Industrial Engineering)

A Thesis Submitted for the degree of

Doctor of Philosophy

At the School of Engineering and Mathematics
Faculty of Communications Health and Science

EDITH COWAN UNIVERSITY

PERTH WESTERN AUSTRALIA

June 2000

Abstract

Three new iterative, dynamically self-adaptive, derivative-free and training parameter free artificial neural network (ANN) training algorithms are developed. They are defined as *self-adaptive back propagation, multi-directional and restart* ANN training algorithms. The descent direction in *self-adaptive back propagation* training is determined implicitly by a central difference approximation scheme, which chooses its step size according to the convergence behavior of the error function. This approach trains an ANN when the gradient information of the corresponding error function is not readily available. The self-adaptive variable learning rates per epoch are determined dynamically using a constrained interpolation search. As a result, appropriate descent to the error function is achieved.

The *multi-directional* training algorithm is self-adaptive and derivative free. It orients an initial search vector in a descent location at the early stage of training. Individual learning rates and momentum term for all the ANN weights are determined optimally. The search directions are derived from rectilinear and Euclidean paths, which explore stiff ridges and valleys of the error surface to improve training. The *restart* training algorithm is derivative free. It redefines a de-generated simplex at a re-scale phase. This multi-parameter training algorithm updates ANN weights simultaneously instead of individually. The descent directions are derived from the centroid of a simplex along a reflection point opposite to the worst vertex. The algorithm is robust and has the ability to improve local search. These ANN training methods are appropriate when there is discontinuity in corresponding ANN error function or the Hessian matrix is ill conditioned or singular.

The convergence properties of the algorithms are proved where possible. All the training algorithms successfully train exclusive OR (XOR), parity, character-recognition and forecasting problems. The simulation results with XOR, parity and character recognition problems suggest that all the training algorithms improve significantly over the standard back propagation algorithm in average number of epoch, function evaluations and terminal function values. The multivariate ANN calibration problem as a regression model with small data set is relatively difficult to train. In forecasting problems, an ANN is trained to extrapolate the data in validation period. The extrapolation results are compared with the actual data. The trained ANN performs better than the statistical regression method in mean absolute deviations; mean squared errors and relative percentage error. The *restart* training algorithm succeeds in training a problem, where other training algorithms face difficulty. It is shown that a seasonal time series problem possesses a Hessian matrix that has a high condition number. Convergence difficulties as well as slow training are therefore not atypical. The research exploits the geometry of the error surface to identify self-adaptive optimized learning rates and momentum terms. Consequently, the algorithms converge with high success rate. These attributes brand the training algorithms as self-adaptive, automatic, parameter free, efficient and easy to use.

Key words: *Self-adaptive back propagation, Derivative free training, Multi-directional training, Restart training, Oriented search, self-adaptive momentum term and learning rate, Artificial Neural Network.*

Declaration

I certify that this thesis does not, to the best of my knowledge and belief:

i.) incorporate without acknowledgment any material previously submitted for a degree or diploma in any institution of higher education;

ii.) contain any material previously published or written by another person except where due reference is made in the text; or

iii.) contain any defamatory material.

Signature _____
(Shamsuddin Ahmed)

Date June 2000

Acknowledgments

I begin in the name of God, Most Gracious, Most Merciful.

I express my gratitude to Associate Prof. Dr. Jim Cross who has been the principal supervisor. Despite his busy schedule, I had the privilege of being able to interrupt him in his office when I needed him and he was there to sort out the issues. His timely and active communication before joining the program prompted me to undertake the research work at ECU. His admirable quality to accommodate the many events and situations that I came across is heart-felt.

I recall the stimulating discussions with Associate Prof. Dr. A. Bouzerdoum who is the co-supervisor. His comments and suggestions have been instrumental in shaping the thesis.

The administrative support provided by the scholarship officer Ms. J. Clarke and administrative officer Ms. Ann Johnsen has been indispensable and without their coordinated support I would not have accomplished this result in such a time frame.

The financial support provided by the school of engineering and mathematics in various forms of scholarships and conference travel grants throughout the research period is greatly acknowledged.

I am thankful to the anonymous referees from the journals and the conferences. Their suggestions and comments have improved the content of the research findings in many ways. My thanks are also due to Dr. Malcolm Walsh at EMU for his valuable suggestions. In no way I do forget to thank the panel of examiners who evaluated this research work with great care.

Finally, I dedicate this work to my late father Khalil Ahmed whose entrepreneurial skills have inspired me to venture into the unknown with dedication, courage and sincerity.

Publications emerging from this work

(Refereed conference proceedings and refereed journals)

1 Ahmed, S. and Cross, J., (1999a). Derivative Free Training in Seasonal Time Series Using Grid Search. *MOSIM99, International Congress on modeling and Simulation Proceedings, vol. 4*, Editors Les Oxley and Frank Scrimgeour, 1057-1062.

2 Ahmed, S. and Cross, J., (1999b). Neural Network Model to Study Seasonal Time Series Data Using Derivative Free Search Method, *26 th. International conference on computers and industrial engineering*, Melbourne, Australia., 1999, Editor E. Shayan, 673-679.

3 Ahmed, S. and Cross, J., (1999c). Modeling Accommodation Demand in Australian Hotel Industry. Submitted for publication.

4 Ahmed, S. and Cross, J., (1999d). A Tourist Growth Model to Predict Accommodation Nights Spent in Australian Hotel Industry. *SIRC 99*, Dunedin, New Zealand, December 1999.

5 Ahmed, S., (1999a). A Derivative Free Optimization Method Using Improved Simplex Search in Higher Dimension, T*he 15th National Conference of Australian Society for Operations Research, Gold Coast, Editor: E. Kozan, 97-104.*

6 Ahmed, S., (1999b). Derivative Free Optimization in Higher Dimension, accepted with *revisions International Transaction in Operations Research*, 2000.

7 Ahmed, S., and Cross, J., (2000). Convergence in Artificial Neural Network without Learning Parameter, *Second International Computer Science Conventions on Neural Computations, 2000*, Berlin, Germany.

8 Ahmed, S., Cross, J., and Bouzerdoum, A. (2000a). A New Self-Adaptive Back Propagation Training Method, *Second International Computer Science Conventions on Neural Computations, 2000*, Berlin, Germany.

9 Ahmed, S., Cross, J., and Bouzerdoum, A. (2000b). Derivative Free and Multi-directional Training to Model Seasonal Time Series, accepted in *Decision Support Systems* Journal, (under review).

10 Ahmed, S., and Cross, J., and Bouzerdoum, A. (2000c). Performance Analysis of a new multi-directional training algorithm for Feed-Forward Neural Network, Accepted for publication in *World Neural Network Journal*, 2000.

Contents

List of Figures

List of Tables

Notations

ANN = Artificial Neural Network
BP = Back Propagation
S = A feasible set of solution
I = Total number of neuron in 1st layer (input layer)
N = Total number of neuron in 2nd layer (hidden layer)
O = Total number of neuron in 3rd layer (output layer)
$i = 1,2,.....,I$, $n = 1,2,.....,N$, $o = 1,2,.....,O$
$p = 1,2,.....,P$, P = Total number of training set
y_o = Actual Output of Artificial Neural Network (ANN) at output neuron o in 3rd layer
$\hat{y}_o$ = Estimated output of ANN at output neuron o in 3rd layer
$\bar{y}_o$ = Expected value of output of ANN at output neuron o in 3rd layer
w_{in}^1 = ANN weight connecting neuron i in 1st layer to neuron n in 2nd layer
w_{no}^2 = ANN weight connecting neuron n in 2nd layer to neuron o in 3rd layer
x^p = The training pattern defined as design variables $\{x_1^p, x_2^p, ..., x_I^p\}$
m = Total number of weights in ANN (Dimension of function to be minimized)
w_j = One dimensional weight in ANN in 1st layer and 2nd layer
j = an index, indicating ANN weights, $j = 1,2,...,m$
k = Training cycle counter or epoch
η^1 = Adaptation length or learning rate in 1st layer
η^2 = Adaptation length or learning rate in 2nd layer
η^m = Momentum term to prevent oscillation in training $(1 \le \eta^m \ge 0)$
η = Conjugate gradient coefficient to generate linearly independent directions
η_k = Vector of adaptation length or learning rate in 1st and 2nd layer $\{\eta_j = (\eta_1, \eta_2, ... \eta_m)\}_k$ during iteration k.
∇g = First order gradient change in quasi Newton method
$\nabla f(w_k)$ = Gradient of ANN calculated at iteration k
∂ = Partial derivative
L = Interpolation search interval in parameter space
H = Hessian of ANN calculated at iteration k
$\bar{I}$ = Identity matrix of size mxm
D = Deflected matrix of inverse Hessian
(A,a) = Largest and smallest eigenvalue in Hessian matrix H
w_k = ANN weight vector determined during iteration k
w^* = Optimum training weight value
η^* = Optimized learning rate
$f(w^*)$ = Optimum function value
d_j = Direction of move in parameter space corresponding to index j
Φ = Activation function to be chosen
$\tilde{h}_n$ = Input to the hidden layer neuron n

$\tilde{g}_n$ = Output from the hidden layer neuron n

z_o = Input to the outer layer neuron o

ε = Square error in ANN due to the difference of output in neuron κ and actual output

$\alpha_1, \alpha_2, \alpha_3$ = Control parameters

$c_1, c_2,$ = Control parameters

b_1, b_2 = Bounded intervals in interpolation search

$\|\bullet\|$ = Norm of a vector, usually the Euclidean norm

$|\bullet|$ = Absolute value

MAPE = Mean absolute percentage error, $MAPE = \frac{1}{P}\sum_{p=1}^{P}\left(y^p - \hat{y}^p\right)/y^p$

MPE = Mean percentage error, $MSE = \frac{1}{P}\sum_{p=1}^{P}\left(y^p - \hat{y}^p\right)^2$

MAE = Mean absolute error, $MAE = \frac{1}{P}\sum_{p=1}^{P}\left|y^p - \hat{y}^p\right|$

ME = Mean error, $ME = \frac{1}{P}\sum_{p=1}^{P}\left(y^p - \hat{y}^p\right)$

SSE = Sum of squared-error, $SSE = \sum_{p=1}^{P}\left(y^p - \hat{y}^p\right)^2$

$v^t \equiv \{v^1, v^2, \ldots\ldots, v^{m+1}\}$ = Vertices are representing the simplex in E^{m+1}.

v^1 = Initialization at vertex $t = 1$, with initial guess; $\{w_j : w_1, w_2, \ldots\ldots, w_m\}$.

d^t = Vectors specifying search directions at vertex t: $d^t = \begin{cases} 1 & \text{at } t \text{ th position} \\ 0 & \text{elsewhere} \end{cases}$

d^m = Vectors specifying restart search directions

λ^t = A scalar quantity that characterize simplex size at vertex t

$f(v^t)$ = Objective function value of the function f being minimized in restart training

α = Reflection coefficient, $\alpha > 0$

β = Expansion coefficient, $\beta > 0$

γ = Contraction coefficient, $\gamma > 0$

ξ = Scaling coefficient, $\xi > 0$

ζ = Scaling factor, $\zeta > 0$

ρ = Termination criteria to stop algorithm

c = Centroid of the simplex

r = Reflection of the simplex

e = Expansion of the simplex

g = Contraction of the simplex

v^* = Minimum point found by simplex

$l = \arg\{\min_t f(v^t)\} \Rightarrow$ Low index for function value at vertex t

$h = \arg\{\max_t f(v^t)\} \Rightarrow$ High index for function value at vertex t

$s = \arg[\{\max_t f(v^t)\} t \neq h] \geq f(v^t) \Rightarrow 2^{nd}$ high index for function value at vertex t

Chapter 1

Introduction

1.1 Background of the Gradient Based Back Propagation Training

Historically, the study of artificial neural network (ANN) as computational model started with the pioneering paper of McCulloch and Pitts (1943) who described the theory of formal neural networks. The work on simulation of a neuron is also attributed to Hebb (1949). Rosenblat (1952; 1962) introduced the concept of the artificial neuron called the perceptron whose dendrites are replaced by ANN connection weights. ANN is classified as an intelligent system, which has the capability to represent knowledge (Haykin, 1994). It is Rumelhart et al. (1986), who popularized the back propagation (BP) algorithm, which has been widely used to train multi-layer feed forward Ann's. This algorithm can evolve a set of weights to produce an arbitrary mapping from input to output vectors (Li and Da, 2000). It is an iterative gradient descent algorithm designed to minimize a measure of the difference between the actual output and the ANN output. The ANN is a form of function approximation tailored by the designer to reflect the particular problem to be solved. The BP training differs from the conventional optimization methods by the fact that in BP, ANN are several magnitudes higher in dimension and contain gradient information in two or more layers. The gradient descent BP uses first derivative information that is tangent to the contour of an error function and the negative gradient points towards the minimum of the error function. In classical optimization one works with single gradient vector, while in ANN online computation the gradient vector is different (Jacobs, 1988; Salomon et al., 1996). The standard BP training algorithm is not self-adaptive and the training parameters are determined by trial and error methods to accelerate training (Jang et al., 1997).

1.2 Difficulties with Self-Adaptive ANN Training Algorithms

The dynamically self-adaptive training algorithms are those classes in which the learning rates and momentum terms are determined automatically in every stage of training. Mostly such algorithms take into account the local condition of the error surface to compute the suitable learning rates and the momentum term that accelerate convergence and prevents oscillation during training. There are few algorithms that can be considered fully self-

adaptive. Such algorithms have the advantage of improving training solutions. They also converge fast while preventing severe oscillations.

The standard back propagation-training algorithm updates ANN weights w_k at iteration k according to the following expression:

$$w_{k+1} = w_k + \eta_k [-\nabla f(w_k)]. \tag{1.1}$$

The amount of learning rate, η_k, or step size, which is also called the adaptation length to be taken in BP for training, is arbitrary. Proposing a suitable value of the adaptation length is a controversial research issue. Mostly, ad hoc methods exist to select this parameter. In standard back propagation, the direction is computed using the gradient information, $d_k = -\nabla f(w_k)$, from a single training pattern. In on line training, the gradient components $\nabla f(w_k)$ may result in different directions for a particular weight (Kamarthi and Pittner, 1999), therefore, a single descent direction is not generated. The fixed value of learning rate η_k does not always decrease the error function value. It also depends on the shape of the error function (Jacobs, 1988) from one iteration k to the next iteration $k+1$, $k+2$,.... and so on. Such iterative methods therefore, do not produce a convergent sequence in the strict sense. The fixed value of learning rate, η_k, may misdirect the search towards the minimum and the directions d_k generated from $\nabla f(w_k)$ are different for a single weight component in standard back propagation (Jacobs, 1988; Weir, 1991; and Vogl, 1988).

The Newton type training algorithm performs well if the training is initiated close to the optimum points. Convergence becomes difficult if the starting point is far from the minimum. Levenberg (1944) and Marquardt (1963) type algorithms are modified versions of Newton's method in which the Hessian matrix H is replaced with a corresponding modified matrix $[H+\eta_k \bar{I}]$ to maintain positive definite property. In another class, the direction of movement d_k is taken to be $-D\nabla f(w_k)$ where D is a positive definite matrix that approximates the inverse of the Hessian matrix. This class is referred to as quasi Newton training methods. Broyden (1967) proposed a useful generalization of the Davidon-Fletcher-Powell method. He introduced a measure of degrees of freedom in updating the matrix D (Bazaraa et al., 1993). The Davidon-Fletcher-Powell method converges to an optimal solution if the objective function is convex, if the Hessian matrix is positive definite at the solution point, and if an exact line search is used. The method converges super-linearly. In Newton type training methods the approximation to the Hessian matrix itself is a source of difficulty.

The method of conjugate direction is an efficient technique. The method of Fletcher and Reeves (1964) generates conjugate directions by taking a suitable combination of the current gradient and the direction used at the previous iteration. The original idea presented by Hestenes and Stiefel (1952) led to the development of this method, as well as to the conjugate gradient algorithms of Polyak (1969) and Sorenson (1969). These methods become indispensable when the problem size increases. Polak and Ribiere (1969) propose another conjugate gradient scheme, which is preferable for non-quadratic ANN error functions. The possible source of difficulty with this training method is the requirement of linear independence of search and conjugate directions.

1.3 Development of Specialized Training Algorithms

Consider the case when the ANN error function is discontinuous or ill conditioned. Convergence is a potential problem in such case. The difficulty also arises, as the ratio of maximum and minimum eigenvalues of the Hessian matrix becomes large (Bishop, 1995). The convergence problem occurs due to scaling of variables (Oren and Luenberger, 1974). It also occurs in parameter identification problems. Most of the ANN applications are closely related to such problems (Warner et al., 1996). As the smallest eigenvalue approaches a value nearly close to zero, the convergence becomes slow except for stationary points that lie in the range of the local minimum. The Hessian matrix in this case is either ill conditioned or singular (Wang et al., 1998). These situations are common in parameter identification problems (Mehra and Stepner, 1973 and Gupta and Mehra, 1974). The derivative free training methods can resolve some of these difficulties. These developments provide an alternative means to train an ANN other than the purely derivative based training methods. Therefore, new self-adaptive training algorithms that will have important features in resolving some difficulties in existing ANN training methods are proposed. In particular, the research intends to:

a.) train an ANN with self-adaptive learning rate parameter;
b.) abandon the arbitrarily selection of the training parameters;
c.) relieve the user/ANN trainer from pre-optimizing the learning rate parameters;
d.) train an error function that is discontinuous or ill conditioned or develops stiff ridges in error surface;
e.) train an ANN for which the corresponding error function is not explicit (Conn et al., 1997);
f.) develop multi-directional self-adaptive training algorithm without the derivative information of the error function;
g.) develop central difference approximation scheme that provides the derivative information of the error function implicitly. The step size of the central difference scheme is controlled by the convergence behavior of the error function;
h.) train an ANN without oscillations;

i.) develop simplex search derivative free restart training algorithm to improve local convergence.

All these issues are resolved by designing training algorithms and software developed using the FORTRAN programming language.

1.4 Content of the Thesis

The development of the self-adaptive derivative free training algorithms is the major concern of this research. To this end three new different derivative free training algorithms are developed. In Chapter 1, the research issues and research problems are briefly discussed. The salient feature of the research is presented and the significant contributions of this research are listed. Chapter 2 provides a critical review of the current state of the art in self-adaptive ANN training algorithms. Computational schemes are reviewed following an introduction to the feed forward artificial neural network. The first order back propagation theory that is most popular, is described to facilitate the discussions in the developments of self-adaptive back propagation training algorithms. The second order training algorithms are presented from theoretical viewpoint to address the difficulties in training. The research problems are identified in Chapter 3. It also includes the research methodology, test problems and the data set for the proposed problems. Approaches to develop the training algorithms are briefly discussed in Chapter 3. A self-adaptive gradient descent training algorithm is developed in Chapter 4. It provides theoretical base to the development of descent directions and self-adaptive training algorithms. A specialized interpolation search and a central difference gradient computation scheme are developed. The self-adaptive parameters are determined from a constrained interpolation search. Therefore, the ad hoc nature of training parameter selection is abandoned. The derivative information is provided implicitly by a central difference approximation scheme, which changes the finite difference step size according to the local condition of the error surface. As a result, a suitable descent direction is obtained. The exclusive OR (XOR) problem is solved to compare the results with the standard BP training. An improved multi-directional self-adaptive derivative free training algorithm is developed in Chapter 5. A search vector first orients the initial starting point to a descent location before attempting the main phase of training. The search follows rectilinear directions. The algorithm accelerates training with a restricted momentum search. The momentum term is determined in a self-adaptive manner. To test the convergence properties of the algorithm, the XOR problem is solved. A new restart simplex training algorithm is presented in Chapter 6. It is also a derivative free and parameter free training algorithm, which improves the simplex search by maintaining a non-degenerate simplex at a defined restart phase. The simplex is prevented form degenerating during search. A re-scale phase constructs the simplex, which confines finite volume. The proposed algorithm finds a good local minimum. The algorithm is compared with a standard test function before

entering into analysis with XOR problem. In Chapter 7, the performances of the newly developed training algorithms are discussed with the parity, letter recognition, seasonal time series and multi-variate statistical analysis problems. The performances of the training algorithms are compared with the standard back propagation training algorithm and the standard regression method. The results suggest that the developed training algorithms are an order of magnitude faster than the standard back propagation algorithm. The restart algorithm improves over the statistical method in a seasonal time series-forecasting problem. The convergence difficulty of the self-adaptive back propagation algorithm is discussed with a seasonal time series problem. The condition number of the corresponding error function is high in magnitude and therefore, convergence difficulty is expected. Finally, in Chapter 8 important conclusions of the research are drawn.

I.5 The Important Contributions

The research contributes in developing three new self-adaptive training algorithms in neural computations and the following contributions are highlighted.

a.) A self-adaptive back propagation-training algorithm that does not require training parameter is developed. The training parameters are derived from prior information of the error function in a constrained search space. Thus, instead of having a fixed value of the training parameter, the algorithm determines variable learning rates dynamically. This approach makes the ANN training method simpler, easier and efficient. The algorithm computes derivative implicitly to accommodate the situations where an analytical expression of the derivative of an error function is difficult to obtain.

b.) A central difference derivative approximation scheme takes into account the local convergence behavior of the ANN error function. During forward pass through the network, the step size is controlled in order to improve training performance.

c.) A multi-directional self-adaptive and derivative free training algorithm is developed. An initial positioning vector is designed to accelerate training. The step length of a restricted momentum search is optimized such that the search direction is not lost. The training algorithm is suitable for the case where the corresponding ANN error function is discontinuous or ill conditioned.

d.) A simplex restart training algorithm that produces a non-degenerate simplex at a re-scale phase is developed. The simplex is prevented from degenerating during search. In addition, a restart phase is designed to align the simplex in a descent location. The algorithm does not require any training parameters.

Unlike other training algorithms, it does not examine the behavior of the error function for one weight at a time. It updates weights concurrently in a single epoch or iteration.

e.) The related training softwares for the new training methods are developed. The parity, letter recognition and statistical analysis problems are solved successfully. The algorithms are faster than the standard back propagation training algorithm and avoid oscillations during training.

f.) All the training algorithms are dynamically self-adaptive and derivative free. There is no user intervention to pre-optimize learning rate, momentum and any other training parameters. The training algorithms self-adapt these parameters according to the geometry of the error function.

g.) A comprehensive review in the state of the art in first and second order self-adaptive back propagation ANN training is provided. Sources of difficulties in ANN computations are discussed. Standard reference in self-adaptive training method is not readily available. This review brings the information together as a source of reference in self-adaptive back propagation training methods.

Chapter 2

State of the Art in Self-Adaptive ANN Computations and Training Algorithms

2.1 Introduction

A comprehensive review of the back propagation self-adaptive training method is provided in this chapter. Sections 2.2 through 2.5 provide the descriptions of the standard back propagation method. The self-adaptive training methods that are based on the theory of first order training are discussed in Section 2.6 through 2.7. The second order training methods are discussed in Sections 2.9 and 2.10. In Section 2.11 the sources of difficulties with the self-adaptive training are provided and finally Section 2.12 provides some relevant discussions on self-adaptive training methods.

2.2 Computations in ANN

Rumelhart et al. (1986), McClelland et al. (1986), Werbos (1974), Bryson and Ho (1969) and Parker (1985) introduce the BP algorithm. Hecht-Nielsen (1990) formally described the ANN function mapping through an error measure. It is stated as "*given any scalar sufficiently small* $\varepsilon \to 0^+$ *and any* ℓ_2 *function* $f:[0,1]^p \to E^m$, *there exists a three-layer back propagation ANN that can approximate* f *within* ε *mean squared error accuracy*". The function f belongs to ℓ_2 if each of f's coordinate function is square integrable on the unit cube $[0,1]^p$ (Hecht-Nielsen, 1990). Although the theorem mentions a three-layer ANN, more number of layers can be used to approximate a function (Hecht-Nielsen, 1990). The function-mapping concept is also attributed to Kolmogorov (1957).

Feed forward ANN with neurons in one hidden layer is a class of ANN structure that has one input and output layer neurons. Figure 2.1 depicts a three-layer feed forward ANN. It consists of autonomous processing units called neurons. Directed arcs join these neurons. Each arc has a numerical weight that specifies the influence of one neuron in one layer on the neuron in other layer. Based on connections; ANN architecture is classified as feed forward and recurrent ANN. In feed forward ANN, the neurons are arranged in layers and have connections in one direction. Input from first layer gradually passes to the hidden layer and finally to the outer layer and connections are only allowed to project forward.

Connections between neurons on the same layer and feed back connections are not allowed in such structure. In recurrent ANN architecture, feed back connections or back coupling are common (Hopfield, 1982; Hopfield, 1984 and Deco et al., 1994). The training methods for this class of ANN are different and are not discussed henceforth. To facilitate the discussions in theoretical issues with self-adaptive ANN computations, throughout the discussions here a *three-layer feed forward neural network* is considered.

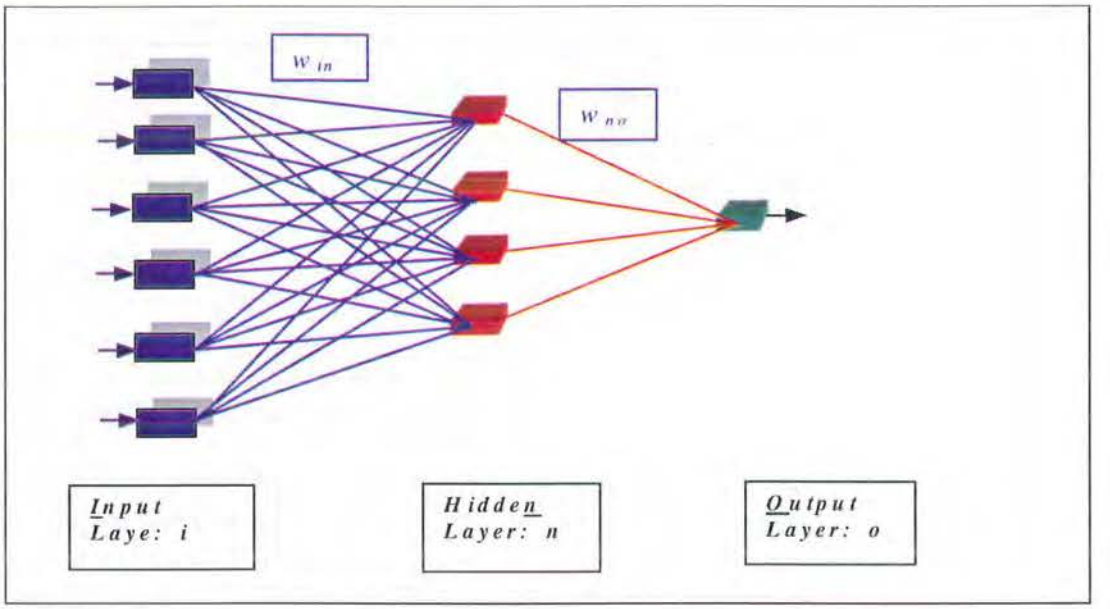


Figure 2.1 BP Neural Network Architecture

In order to develop the neural computational algorithms and discuss different self-adaptive training algorithms, this chapter attempts to present the mathematical computational schemes comprehensively. The input layer consists of I neurons for I dimensional input space. The purpose of this layer is to distribute the I components of the input vectors to the input layer neurons and then to the hidden layer neurons. An ANN is obtained by connecting a neuron from one layer to the next layer through weighted arc connections. The I input layer neurons transmit signals through synapses (Haykin, 1994), which represent connection weights w_{in}^{1}, from neuron i in first layer to neuron n in the hidden layer in which there are N neurons. The input layer neurons receive a pattern at instance p in the form $x^p \equiv (x_1, x_2, \ldots\ldots, x_I)$ from an external source and propagate all the input to the hidden layer neuron through the weighted arc connections. There are a total of P patterns in the training set. For example $I=6$, $N=4$, $O=1$ in Figure 2.1. The net input, $\breve{h}_n^p$, to the hidden layer neuron n is given by:

$$\breve{h}_n^p = \sum_{i=1}^{I} w_{in}^{1} x_i^p \,. \qquad (2.1)$$

The net input, $\tilde{h}_n^p$, coming from the first layer neurons through the weighted arc connections, passes through the exponential type activation function, Φ_1, (Haykin, 1994) present in hidden layer neuron n to produce an output, $\tilde{g}_n^p$, as shown in the following equation:

$$\tilde{g}_n^p = \Phi_1\left(\tilde{h}_n^p\right) = \frac{1}{1+e^{-\tilde{h}_n^p}} = \frac{1}{1+e^{\left(-\sum_{i=1}^{I} w_{ni}^1 x_i^p\right)}}. \tag{2.2}$$

This output from the hidden layer neuron n is fed to the output layer neuron o as input. The net input, z_o^p, to neuron o in the output layer is shown in Equation 2.3, when it passes through the hidden layer weighted arc connection w_{no}^2:

$$z_o^p = \sum_{n=1}^{N} w_{no}^2 \tilde{g}_n^p. \tag{2.3}$$

The neuron o in the output layer produces an output, $\hat{y}_o^p$, using an exponential activation function Φ_2, is shown in Equation 2.4. Here $\Phi_1 = \Phi_2 = \Phi$ indicating the same function.

$$\hat{y}_o^p = \Phi_2\left(z_o^p\right) = \frac{1}{1+e^{\left(-\sum_{n=1}^{N} w_{no}^2 \tilde{g}_n^p\right)}}. \tag{2.4}$$

This ANN output, $\hat{y}_o^p$, is compared to the actual observation, y_o^p, and the difference is measured. The difference between the ANN output $\hat{y}_o^p$ and the actual observation y_o^p is shown as an error in Equation 2.5:

$$\varepsilon = \sum_{p=1}^{P} \sum_{o=1}^{O} \left(y_o^p - \hat{y}_o^p\right)^q. \tag{2.5}$$

Usually q is taken as 2. The back propagation training method suggested in Rumelhart et al. (1986) minimizes the squared error ε, and is shown in Equation 2.6:

$$\varepsilon = \sum_{p=1}^{P} \sum_{o=1}^{O} \left(y_o^p - \hat{y}_o^p\right)^2. \tag{2.6}$$

To control this error, ANN connection weights are changed based on the magnitude of the error. Once again the error in the network is compared for the next input pattern x^p to change the weights. This process of changing weights is repeated until a desired level of accuracy is achieved. As the error becomes smaller the ANN closely approximates the actual response.

2.3 Choice of Activation Function

The activation function $\Phi(\bullet)$ is used to transform information in the hidden layer and output layer neuron. Different neurons can use different activation functions. The typical choices of activation functions are exponential (2.7), tanh (2.8), logistic (2.9), ramp (2.10), step (2.11) and linear (2.12) (Deco and Obradoric, 1996). The example of a sigmoid activation function used in the hidden layer neuron with input $\tilde{h}_n^p$ is shown in Equation 2.7:

$$\text{Sigmoid activation: } \Phi(\tilde{h}_n^p) = \frac{1}{1+e^{-(\tilde{h}_n^p)}}. \tag{2.7}$$

A few examples of activation functions in multi-layer perceptron are given below:

$$\text{Tanh activation: } \Phi(\tilde{h}_n^p) \equiv \tanh(\tilde{h}_n^p) \equiv \frac{e^{(\tilde{h}_n^p)} - e^{-(\tilde{h}_n^p)}}{e^{(\tilde{h}_n^p)} + e^{-(\tilde{h}_n^p)}}; \tag{2.8}$$

$$\text{Logistic activation: } \Phi(\tilde{h}_n^p) = \frac{1}{1+\log(|\tilde{h}_n^p|)}; \tag{2.9}$$

$$\text{Ramp activation: } \Phi(\tilde{h}_n^p) = \begin{cases} c_1, & \text{if } \tilde{h}_n^p \geq c_1 \\ \tilde{h}_n^p, & \text{if } |\tilde{h}_n^p| < c_1 \\ -c_1, & \text{if } \tilde{h}_n^p \leq -c_1 \end{cases}; \tag{2.10}$$

$$\text{Step activation: } \Phi(\tilde{h}_n^p) = \begin{cases} +c_1, \text{if } \tilde{h}_n^p > 0 \\ -c_2, \text{otherwise} \end{cases}; \tag{2.11}$$

$$\text{Linear activation: } \Phi(\tilde{h}_n^p) = \alpha \tilde{h}_n^p, \ \alpha = \text{a constant}. \tag{2.12}$$

Similarly such activation function can be used in the output layer. The widely used activation function is sigmoid activation (2.7). It takes a value in the interval [0,1]. The activation functions in equation (2.10) and (2.11) are discrete. An ANN modeler chooses an appropriate activation function depending on the type of the problem under consideration.

2.4 Gradient Calculations in ANN

The BP algorithm is dependent on the first to hidden and hidden to outer layer gradient of the error function in a three-layered feed forward artificial neural network. The gradients of an ANN error function provide information about the direction of move in error space. Equation 2.6 is rewritten using equations 2.2, 2.3, 2.4 and 2.7 to calculate the input and hidden layer gradients and is expressed in equation 2.13:

$$\varepsilon = \sum_{p=1}^{P}\sum_{o=1}^{O}\left[y_o^p - \Phi\left(\sum_{n=1}^{N} w_{no}^2 \Phi\left(\sum_{i=1}^{I} w_{in}^1 x_i^p\right)\right)\right]^2 . \tag{2.13}$$

The minimization of the squared error, ε, approximates an unknown error function to the actual problem (Hecht-Nielsen, 1990). The error function in Equation 2.13 is of a convex type and unconstrained optimization principles minimize such an error function (Rothe, 1998). The negative value of input and hidden layer gradient information points towards the direction of the minimum function value. The following sections show gradient calculations.

2.4.1 Hidden Layer Gradient

The gradient of the hidden layer ANN error function in equation 2.13 is given by $\nabla f(w_{no}^2) = \left[\frac{\partial \varepsilon}{\partial w_{no}^2}\right]$. The partial derivatives of the error surface are calculated below and shown in Equation 2.14:

$$\frac{\partial \varepsilon}{\partial w_{no}^2} = \frac{\partial \varepsilon}{\partial \hat{y}_o^p}\frac{\partial \hat{y}_o^p}{\partial z_o^p}\frac{\partial z_o^p}{\partial w_{no}^2} . \tag{2.14}$$

The individual components of the partial derivatives are calculated in Equations 2.15 through 2.17:

$$\frac{\partial \varepsilon}{\partial \hat{y}_o^p} = -2\sum_{p=1}^{P}\sum_{o=1}^{O}\left(y_o^p - \hat{y}_o^p\right); \tag{2.15}$$

$$\frac{\partial \hat{y}_o^p}{\partial z_o^p} = \frac{\partial}{\partial z_o^p}\left(\Phi\left(z_o^p\right)\right) = \nabla_z\left(\Phi\left(z_o^p\right)\right) = \hat{y}_o^p\left(1-\hat{y}_o^p\right); \tag{2.16}$$

$$\frac{\partial z_o^p}{\partial w_{no}^2} = \frac{\partial}{\partial w_{no}^2}\left(\sum_{n=1,o=1}^{N,O} w_{no}^2 \bar{g}_n^p\right) = \sum \bar{g}_n^p . \tag{2.17}$$

Using Equations 2.14 through 2.17, a directional vector $-\nabla f(w_{no}^2)$ that points toward the minimum is defined and is expressed in Equation 2.18 with a simplified notation.

$$-\nabla f(w_{no}^2) = (2)\sum\sum\left(y_o^p - \hat{y}_o^p\right)\hat{y}_o^p\left(1-\hat{y}_o^p\right)\sum \bar{g}_n^p . \tag{2.18}$$

The hidden layer weights during training cycle k are updated for the next iteration $k+1$ according to the Equation 2.19. The suffix notation indicates that the weights are different in different training cycles.

$$\left(w_{no}^{2}\right)_{k+1}=\left(w_{no}^{2}\right)_{k}+\eta^{o}\left\{-\nabla f\left(w_{no}^{2}\right)\right\}_{k}. \tag{2.19}$$

The parameter η^{o} is called the learning rate in the hidden to output layer. How to identify this parameter is an open research issue. Including a momentum η^{m} term, the weight update takes place according to the equation shown below:

$$\left(w_{no}^{2}\right)_{k+1}=\left(w_{no}^{2}\right)_{k}+\left(\eta^{o}\left\{-\nabla f\left(w_{no}^{2}\right)\right\}\right)_{k}+\eta^{m}\left[\left(w_{no}^{2}\right)_{k-1}-\left(w_{no}^{2}\right)_{k}\right]. \tag{2.20}$$

2.4.2 First Layer Gradient

The first layer error Gradient ∇w_{in}^{1} can be expressed by the following partial derivatives as shown in Equation 2.21:

$$\nabla w_{in}^{1}=\frac{\partial \varepsilon}{\partial w_{in}^{1}}=\left(\frac{\partial \varepsilon}{\partial \hat{y}_{no}^{p}}\frac{\partial \hat{y}_{o}^{p}}{\partial z_{o}^{p}}\frac{\partial z_{o}^{p}}{\partial \tilde{g}_{n}^{p}}\frac{\partial \tilde{g}_{n}^{p}}{\partial \tilde{h}_{n}^{p}}\frac{\partial \tilde{h}_{n}^{p}}{\partial w_{in}^{1}}\right). \tag{2.21}$$

The individual gradients are shown below through Equations 2.22 to 2.26.

$$\frac{\partial \varepsilon}{\partial \hat{y}_{o}^{p}}=-2\sum\sum\left(y_{o}^{p}-\hat{y}_{o}^{p}\right); \tag{2.22}$$

$$\frac{\partial \hat{y}_{no}^{p}}{\partial z_{o}^{p}}=\hat{y}_{o}^{p}\left(1-\hat{y}_{o}^{p}\right); \tag{2.23}$$

$$\frac{\partial z_{o}^{p}}{\partial \tilde{g}_{n}^{p}}=\frac{\partial\left(\sum_{n=1}^{N} w_{no}^{2}\tilde{g}_{n}^{p}\right)}{\partial \tilde{g}_{n}^{p}}=\sum w_{no}^{2}; \tag{2.24}$$

$$\frac{\partial \tilde{g}_{n}^{p}}{\partial \tilde{h}_{n}^{p}}=\frac{\partial}{\partial \tilde{h}_{n}^{p}}\left(\phi\left(\tilde{h}_{n}^{p}\right)\right)=\nabla\tilde{h}\left(\phi\left(\tilde{h}_{n}^{p}\right)\right)=\tilde{g}_{n}^{p}\left(1-\tilde{g}_{n}^{p}\right); \tag{2.25}$$

$$\frac{\partial \tilde{h}_{n}^{p}}{\partial w_{in}^{1}}=\frac{\partial\left(\sum_{i=1}^{I} w_{in}^{1}x_{i}^{p}\right)}{\partial w_{in}^{1}}=\sum x_{i}^{p}. \tag{2.26}$$

Equation 2.27, with simplified notation, is the descent directional vector $-\nabla f\left(w_{in}^{1}\right)$ that points towards the minimum function value.

$$-\nabla f(w_{in}^2) = 2\sum\sum \left(y_o^p - \hat{y}_o^p\right)\hat{y}_o^p\left(1-\hat{y}_o^p\right)\left(\sum w_{no}^2\right)\bar{x}_n^p\left(1-\bar{x}_n^p\right)\left(\sum x_i^p\right). \tag{2.27}$$

Defining a learning rate η', the ANN input layer weights are updated during training cycle k, according to the Equation 2.28:

$$\left(w_{in}^1\right)_{k+1} = \left(w_{in}^1\right)_k + \eta'\left(-\nabla f(w_{in}^1)_k\right). \tag{2.28}$$

The weight update using a momentum term, η'', is shown in the following equation:

$$\left(w_{in}^1\right)_{k+1} = \left(w_{in}^2\right)_k + \left(\eta'\left|-\nabla f(w_{in}^1)\right|\right)_k + \eta''\left[\left(w_{in}^1\right)_{k-1} - \left(w_{in}^1\right)_k\right]. \tag{2.29}$$

2.4.3 ANN Training

The algorithm for evaluating the derivatives of the error function is known as back propagation (Bishop, 1995) as the errors are propagated backward through the network. However, the term back propagation is used in neural computing literature to mean different things. The multi-layer perceptron is sometimes called a BP network (Bishop, 1995). It also describes a method of reducing the sum of squared error measure of a function. An important concept of BP is to provide an efficient method of computing the derivatives.

Training algorithms involve an iterative procedure for minimization of an error function, by adjusting weights in sequence. The processes of adjusting weights constitute two distinct passes. They are called the forward and the backward pass. The forward pass determines the gradient with respect to the network weights. During the backward pass the errors are propagated backwards through the network and the derivatives are used to adjust the weights using a variety of optimization techniques. Rumelhart et al. (1986) uses gradient descent minimization technique to accomplish this. Each pass through the ANN is called an epoch which processes the input pattern through the network to gradually adjust the weights with the aim of reducing the error measure.

This research views the training an ANN as the minimization of an error function of the form 2.6 or 2.13. A feed forward ANN is trained either using a batch or on line training method (Deco and Obradovic, 1996). In batch training the weights are modified after calculating the error on all training patterns. On-line or incremental training updates the weights at each presentation of the training pattern. To train the ANN, a training set $\{(x^1,y^1),(x^2,y^2),.....,(x^P,y^P)\}$; is applied to the input layer. The notation (x^p,y^p) represents the input vector x and output vector y associated with pattern $p=1,2,....,P$. The output $\hat{y}^p$ produced by the ANN becomes as close as possible to the actual output

y'' due to the minimization of the error function (2.13). Training is accomplished by successively adjusting the weights, w_{io}^{1} and w_{no}^{2}.

2.4.4 The Standard BP Algorithm

The standard BP (Rumelhart et al., 1986) relies on the first derivatives of the error function in neural networks to find descent directions. The adaptation lengths η' and η'' in the first to the hidden layer and the hidden to the output layer are fixed or varied depending on the performance of the BP algorithm. A momentum term η^{m} is also selected to prevent oscillation in training (Rumelhart et al., 1986). The algorithm that trains a 3-layered feed forward ANN is described below. The algorithm uses the gradient information in the hidden and outer layers to determine the direction of move according to Equations 2.18 and 2.27. Depending on the training method, the weights are updated in the two layers accordingly using the Equations 2.19 or 2.20 and 2.27 or 2.29. The back propagation algorithm is described in Table 2.1.

Initialize

Set iteration or epoch counter $k \leftarrow 1$; initialize adaptation length to a small value $\eta_k' \leftarrow \alpha_1$, $\eta_k'' \leftarrow \alpha_2$ and momentum term $\eta^{m} \leftarrow \alpha_3$. Set $\varepsilon_k \leftarrow \infty$ to a large value. Initialize connection weights $(w_1, w_2, \ldots\ldots, w_m)$ in all the layers and perform *main step*.

Main step

Compute outer layer gradient using equation 2.18
Update output layer weight using equation 2.19 or 2.20

Compute first layer gradient using equation 2.27
Update weight in first layer according to equation 2.28 or 2.29.

Stopping Criteria

If error in equation 2.13 is reduced to acceptable limit, then stop, otherwise set $k \leftarrow k+1$ and perform *main step*.

Table 2.1 The back propagation training algorithm

2.5 Notation Simplification

To simplify notation in ANN connection weights, the following convention is now adopted to represent a one-dimensional ANN weight vector instead of a two-dimensional weight vector in a three-layered feed forward ANN. A vector w is defined as one-dimensional vector and is transformed from two dimensional weight vectors w_{in}^1 and w_{no}^2. The vector w represents the entire connection weights in ANN and is defined in Equation 2.30 and 2.31:

$$w_{(n-1)I+i} = w_{in}^1, \forall(i,n); \tag{2.30}$$

$$w_{IN+(l-1)N+n} = w_{no}^2, \forall(n,o). \tag{2.31}$$

Clearly, the total numbers of elements in weight vector w are $(I.N+N.O)$. The notation, $w=(w_j, j=1,.....,(I.N+N.O))$, represents the ANN weights. As an example, in a 6-4-1 ANN, $I=6$, $N=4$, $O=1$ and $w=(w_j, j=1,2,.....,28)$. This indexing method uniquely describes the ANN weights from two dimensions to a one-dimensional weight vector. The two and one-dimensional weights, for example, are related as $w_{12}^1 = w_7$, $w_{22}^1 = w_8$, $w_{34}^1 = w_{23}$, $w_{21}^2 = w_{26}$, and so on.

With this notation, the basic training scheme is expressed in Equation 2.32 and is similar to 2.19 and 2.28. The learning rate parameters, η_k' and η_k'' are replaced by a single learning rate parameter, η_k, in Equation 2.32:

$$w_{k+1} = w_k + \eta_k\{-\nabla f(w_k)\}. \tag{2.32}$$

This simplification is convenient in batch or off-line training implementations. In gradient descent BP, the negative gradient $-\nabla f(w)$ of the function $f: w_k \in E^m$ defines the direction along which the function has the maximum local decrease in value.

2.6 Oscillations in BP Training

The error function forms geometry in the weight space, and the training refers to locating updated weight vector w_k that minimizes the error function. An on-line training scheme (Rumelhart et al., 1986) modifies the weights of the ANN immediately after the presentation of each input pattern with the target pattern but in off-line training, all the pairs of patterns in the training set are presented once to update the ANN weights. The vector w_k is the updated network weight at iteration k. In feed forward ANN the weight vector is adapted according to the recursion:

$$w_{k+1} = w_k + \eta_k d_k \tag{2.33}$$

to find the trained weight vector. This adaptation is performed presenting a set of pairs of input and target vectors to the ANN sequentially. The parameter η_k, which is called the learning rate, is usually positive and is determined by the user arbitrarily. The direction vector d_k is the negative gradient of the error function $f(w_k)$ defined as:

$$d_k = -\nabla f(w_k). \tag{2.34}$$

During training the weight update follows a direction that yields maximum local reduction in the error function. Convergence can be achieved if the learning rate is small enough to provide a steepest descent move (Salomon and Van Hemmen, 1996) and the convergence difficulty is noticed when the problem size increases. In some experiments Jang et al. (1997) observe that as the value of $\eta_k \to 1$, the search accelerates but oscillates at the minimum without locating it precisely. With small value of η_k, the search process is extremely inefficient and requires large iterations to reach minimum. Jang et al. (1997) further claims in a given experiment that the search process is inefficient when η_k is small, $\eta_k \leq 0.2$, and when η_k is moderate, $\eta_k \equiv 0.6$, the search path tends to show oscillatory behavior (Jang et al., 1997). Using $\eta_k > 0.6$, the search path diverges and the method fails (Jang et al., 1997). This behavior points to the difficulty in training.

2.7 Training with Momentum to Prevent Oscillations

Rumelhart and McCleland (1986); Rumelhart, Hinton and Williams (1986) and Plaut et al. (1986) propose to update ANN weights according to the following equation:

$$w_{k+1} = w_k - \eta_k \nabla f(w_k) + \eta_k^{m} (w_k - w_{k-1}) . \tag{2.35}$$

The momentum term η^m regulates the search directions. Chan et al. (1987) and Chan et al. (1990) suggest a heuristic to update η based on the back propagation rule similar to the expression 2.36. The update rule is given by:

$$\eta_k = \eta_{k-1}(1 + \tfrac{1}{2} Cos\phi_k) \tag{2.36}$$

where, the value of $Cos\phi_k = \frac{\nabla f(w_k)^T d_{k-1}}{\|\nabla f(w_k)\| \|d_k\|}$.

This scheme aims at adjusting the step size when the proper descent direction is determined. The momentum parameter measures the effect of past weight change on the current training of the ANN. It acts as a memory term that incorporates the weight change of the previous step and slows down oscillation near the minimum of the error surface (Salomon and Van Hemmen, 1996). Yu, Loh and Miller (1993) propose a similar method to compute d_k using information of the current gradient and the direction from the last iterations.

The parameter η_k^m takes a value between 0 and 1. It smoothes out the oscillation in weight update and tends to resist erratic weight changes due to the non-quadratic error surface along which gradient descent is oscillatory (Bishop, 1995). The use of a momentum term does not always accelerate training and is dependent on application (Widrow and Lehr, 1990). In some cases it plays a role in preventing learning from being trapped into poor local minimum. It may be beneficial to some local features of minimum (Weir, 1991) but its acceleration features are limited, when small step sizes are required near the minimum.

2.8 Heuristic Self Adaptive Training Methods

Hush et al. (1992) point out that the ANN error surface contains many flat surfaces and steep regions. The search in the region of a flat surface causes the algorithm to decrease the error function slowly. The convergence thus becomes slow (Van Ooyen and Nienhuis, 1992; Krzyzak et al., 1990). The BP provides the direction of learning but does not suggest the amount of step to be taken in training.

Self-determination of adaptive length is considered in Weir (1991). The learning rate parameter selection is based on extreme gradient information (Weir, 1991). The extreme gradients with respect to some training weights show the magnitude of the gradients and do not provide the information about the optimal move. The step length determined from this information some times over shoots the minimum. Fahlman (1988) uses a heuristic called quickprop. This method approximates error surface by quadratic polynomial using information during training and the minimum of the polynomial is used to update the training weights. The heuristic of Hush and Salas (1988) suggest a method to re-use the gradient. They use the gradient in as many iterations as possible so long as the decrease in the error function is noticed. A new value of the learning rate is determined depending on the re-use count.

Vogl et al. (1988) attempt to improve the BP by updating ANN weights after each pattern is presented. All weight changes are calculated as usual through back-propagation, but these changes are not immediately applied. Instead, the changes for each weight are summed over all of the input patterns and the sum is applied to modify the weight after each iteration over all the patterns. They vary the adaptation length η_k during iteration,

according to the reduction in total error for all patterns. If an update reduces the error measure, then η_k is multiplied by a factor greater than one for the next iteration. If a step increases the network error by a small percentage of the order 1.5, then η_k is multiplied by a factor less than one and the momentum term η_k^m is dropped. The process is repeated until a successful step is taken. At this stage the momentum term η_k^m is included for the next iteration. The rational behind such maneuver is that when the topography is relatively uniform and the descent direction is in a relatively smooth line, the memory implicit momentum term will aid in convergence. If the topography is such that the descent direction is lost, then the direction is changed so that the search continues towards the minimum direction. The persistence in addition of momentum term does not always lead to improvement (Qian, 1999). To orient the search in descent direction, the memory term is set to zero so that the memory from previous step is lost.

Salomon and Van Hemmen (1996) provide a method to adjust the learning rate using the following expression:

$$\eta_{k+1} = \begin{cases} \eta_k \varsigma & \text{if } f(w_k - \eta_k e_k \varsigma) \le f(w_k - \eta_k e_k / \varsigma) \\ \eta_k / \varsigma & \text{otherwise .} \end{cases} \tag{2.37}$$

The term e_k is the unit vector in the direction $\nabla f(w_k)$ and the suggested value of ς is 1.8. This factor in combination with the learning rate controls the training performance as shown in above equation. The method uses fixed value of learning rate. The suggested parameter in combination with the unit vector e_k, which contains the sign of gradient of error function, changes the learning rate dynamically. Due to the change in gradient, the learning rate shows the effect of dynamic variations. It however does not perform a steepest descent training or search in weight space. The algorithm is claimed to be dynamic but at times the parameter is kept constant and recommends change when severe oscillation is experienced. The method looks after extreme oscillations rather than the sensitive nature of the error surface. The direction is favorable but the step length is not always correct in this training method. There exists a chance of missing the direction towards the minimum. The selection of small step size or learning rate may not overshoot the minimum but may slow the convergence. According to Salomon and Van Hemmen (1996) the algorithm is not restricted in determining the appropriate learning rate.

All ANN weights are dynamically adjusted in the training methods proposed by Cater (1987), Codrington and Mohandes (1994), Mohandes et al. (1994), Nachtsheim (1994), Salomon and Van Hemmen (1996) and Weir (1991) based on heuristically determined self-adaptive parameters.

Individual learning rates for each weight are determined separately in Jacobs (1988), Pirez et al. (1993) and Silva et al. (1990). They use a different heuristic to adapt the individual

learning rates from the information gained in earlier iterations. Jacobs (1988) determines three parameters and needs fair amount of experiment to identify suitable values of these parameters, while Silva and Almedia (1990) need two parameters but show unstable convergence behavior.

According to Jacobs (1988), if the error surface is relatively flat along a weight dimension, the derivative of the weight is small in magnitude and therefore the weight is adjusted by a small amount. Alternately, where the error surface is highly curved along a weight dimension, the derivative of the weight is large in magnitude and consequently the weight is adjusted by a large amount with the risk of overshooting the minimum. If the eignevalues of the Hessian matrix of the error function are far apart, the error function forms a geometry that is skewed. The negative gradient vector in steepest descent training in this case may not point towards the minimum of the error surface. An arbitrary learning rate is not appropriate for all portion of the error surface determined in these methods. The component of the gradient vector is smaller in the direction of the eigenvector corresponding to the minimum eigenvalue as compared to the direction of the eigenvector with reference to the maximum eigenvalue (Jacobs, 1988 and Le Cun et al., 1993). The minor and major axes of the contour surface are related to the minimum and maximum eigenvales of the Hessian matrix. The value of the learning rate that produces moderate steps along major axes of the contour may produce large steps along minor axis in the weight space and the training exhibits oscillations. In computation, the value of learning rate is defined such that the successive steps in weight space do not overshoot the minimum of the error surface. Most often the value of the learning rate is limited by the magnitude of the largest eigenvalue and only small steps in weight space are taken in the direction of major axes of the error surface (Jacobs, 1988). The learning rate is either dynamically adjusted for all weights or separately for each weight. These methods use some local features of the error function to adjust weights during computation of the error function. Johansson et al. (1992) report a line search in the steepest descent BP. However, in this method the user determines the learning rate η_k arbitrarily.

2.8.1 Parameter Dependent Training

Kamarthi and Pittner (1999) propose a regression type training method. They observe that in back propagation training the gradient of the error surface is either positive and monotonically decreasing or negative and monotonically increasing for several iterations. This behavior suggests that the error surface has a smooth variation along the respective axis and therefore extrapolation is possible using the information of weights at the end of a few iterations. The convergence patterns of the weights are examined and the weights for the next iteration are predicted by extrapolation using the trends in weights. This prediction relies on previous iterations and a corresponding extrapolation function. The success of this method depends on the assumption that behavior of this kind must exist during the entire training cycle. Such a phenomenon is not always true, since for some

training cycles the reduction in error function is insignificant to accommodate this behavior.

2.9 Second Order Self-Adaptive Newton Type Training Algorithms

Second order training methods are faster than the first order BP training (Johansson et al., 1992 and Wang and Lin, 1998) and have quadratic convergence properties. Second order training suffers due to ill conditioning of the error function as the ANN grows in size (Wang and Lin, 1998). The major concern in second order Newton based training algorithms is the storage and inversion of the Hessian matrix. Training methods such as variable metric, conjugate gradient and other similar ones have improved convergence properties. These algorithms are concerned with the generation of feasible learning directions. The network weight update has the following form:

$$w_{k+1} = w_k - \eta_k (H^{-1}) \nabla f(w_k). \tag{2.38}$$

The primary requirement in the second order training methods is to maintain at least positive semi definite properties in the Hessian matrix. In the event the inverse Hessian matrix $[H]^{-1}$ is near singular, the algorithm fails to converge and results in premature termination. A common approach is therefore to design a deflected matrix $[D]$, which possesses the required properties of the inverse Hessian matrix $[H]^{-1}$ and does not require to be inverted. The deflected matrix must always posses all the characteristics of the inverse Hessian matrix $[H]^{-1}$ in full second order method and preserving these properties results in new training methods.

Newton's second order method to train ANN is not efficient (Haykin, 1994). The method needs second order derivative information. The approximation of a Hessian matrix by its diagonal term does not maintain the major properties of the true Hessian matrix of a multi-layer perceptron (Wang and Lin, 1998). The inversion of a large Hessian matrix poses computational difficulty (Bazaraa et al., 1993 and Luenberger, 1984). Newton's method requires a good initial estimate for convergence. The full Newton's method in ANN training is not globally convergent and does not consider the special properties of the Hessian matrix. In a recent study, Wang and Lin (1998) attempt to improve Newton's second order method to train ANN training using a block Hessian matrix derived from the order-based-derivatives of multi-layer perceptrons. The following discussion is important to describe the principle behind Newton type second order training.

2.9.1 Levenberg (1944)-Marquardt (1963) Hessian Update

The method due to Levenberg (1944) and Marquardt (1963) is similar to the Newton algorithm. The method corrects the Hessian matrix from possible ill conditioning. To

demonstrate the principle behind such a training scheme, consider the following steps that construct and improve learning directions d_k. In Newton's method d_k is constructed using $[-H^{-1}\nabla f(w_k))$. However, it may not be a descent direction if the Hessian matrix H is not positive definite (Luenberger, 1984). A basic modification is shown below:

$$d_k = -\{[\eta_k \bar{I} + H]^{-1}\nabla f(w_k)\}: \ \eta_k \geq 0. \tag{2.39}$$

The matrix I is the identity matrix. This modification provides the property of positive definiteness to the inverse Hessian matrix $[H]^{-1}$, at all stages of training while maintaining $\eta_k > 0$. The weight update is similar to standard back propagation and the direction vector is defined in Equation 2.39.

Newton type methods work well for small problems (Dennis and Schnabel, 1983; Luenberger, 1984 and Bazaraa et al., 1993) but computationally they are very expensive due to the Hessian matrix update and inversion. The Newton type BP training method is sensitive to initial weights (Kolen and Pollack, 1991). Starting with inappropriate weights causes poor convergence. The eigenvalues of the Hessian matrix are responsible for the geometry of the error surface (Moller, 1997). The Levenberg (1944) and Marquardt (1963) method and all other similar methods force the Hessian matrix to be positive definite. This can be fatal in situation when the Hessian is singular. The approximation matrix is significantly different and convergence is greatly affected (Eisenpress and Greenstadt, 1966).

2.9.2 Quasi Newton Condition

During training the change in gradient can be approximated (Rardin, 1998) as:

$$\nabla f(w_{k+1}) - \nabla f(w_k) \equiv -[\eta_k \bar{I} + H](w_{k+1} - w_k). \tag{2.40}$$

Equivalently this is represented as:

$$(w_{k+1} - w_k) \equiv -[\eta_k \bar{I} + H]^{-1}(\nabla f(w_{k+1}) - \nabla f(w_k)). \tag{2.41}$$

Therefore, the directional vectors in equation 2.39 can be constructed from the modified Hessian matrix and first order gradient information. The update equation is given by:

$$w_{k+1} \equiv w_k - \{[\eta_k \bar{I} + H]^{-1}(\nabla f(w_{k+1}) - \nabla f(w_k))\}. \tag{2.42}$$

Define the quantities d_k and ∇g_k, as the direction vector and gradient vector. They are approximated from the past training cycle and are shown in the following two equations:

$$d_k \equiv w_{k+1} - w_k ; \tag{2.43}$$

$$\nabla g_k \equiv \nabla f(w_{k+1}) - \nabla f(w_k) . \tag{2.44}$$

The quantity, ∇g_k, determines the change in gradients as shown above. The deflection matrix D_k of quasi Newton algorithms approximates the gradient change behavior of inverse Hessian matrices at every training cycle to satisfy the following condition:

$$d_k = D_k \nabla g_k . \tag{2.45}$$

To improve search directions with gradient information, when a move is made along the direction d_k, the training is improved if the condition 2.46 holds:

$$\nabla f(w_t) d_k < 0 . \tag{2.46}$$

Quasi Newton methods are improved by imposing this condition. To see how the training is improved, consider the following search vector d_k as directional derivative:

$$d_k = -[H]^{-1} \nabla f(w_k) . \tag{2.47}$$

Using the information of the deflection matrix the search vector is modified as:

$$d_k = -D_k \nabla f(w_k) . \tag{2.48}$$

Enforcing the inequality in equation 2.46, the descent directions are derived. Consequently, the following condition must hold in quasi Newton training method (Moller, 1997):

$$\nabla f(w_k)(-D_k \nabla f(w_k) = \nabla f(w_k)(-D_k) \nabla f(w_k) < 0 . \tag{2.49}$$

This inequality plays the crucial role in satisfying the second order training condition in quasi Newton type training algorithms.

2.9.3 Method of Broyden (1967), Fletcher (1963), Goldfrab (1969, 1970) & Shanno (1970)

The well-known BFGS update is based on a scheme that maintains a deflected matrix D_k with positive definite properties and the update scheme is shown below:

$$D_{k+1} = D_k + \left(1 + \frac{g_k^T D_k g_k}{d_k^T g_k}\right)\frac{d_k d_k^T}{d_k^T g_k} - \frac{D_k g_k d_k^T + d_k g_k^T D_k}{d_k^T g_k}. \qquad (2.50)$$

The neural network training using this training scheme is found in Battiti (1989). This study implies that the update in Equation 2.50 changes deflection matrix modestly during training.

2.10 Conjugate Gradient

Conjugate gradient methods are somewhere between gradient descent and Newton's method. These methods accelerate the slow convergence of the gradient descent method while avoiding the computation of inversion of the Hessian matrix and its storage. Hestenes and Stiefel (1952) originally proposed the conjugate gradient method, which produces non-interfering directions of search. When the ANN error function is quadratic, the method minimizes the error function over the whole sub-space spanned by all previous search directions. The necessary and sufficient conditions for generating non-interfering search directions are mutually conjugate in relation to the Hessian matrix; this is expressed in Equation 2.51:

$$d_i^T H d_j = 0 : i \neq j;\ (i,j) \in m. \qquad (2.51)$$

This condition implies that the learning directions $d_i, d_j : i \neq j$ are linearly independent. This can be verified from the following argument. Consider the problem: $\eta_k = \arg\{\min f(w_k + \eta_k d_k)\}$ in direction d_k. The minimum is achieved when $\frac{\partial f(w_k + \eta_k d_k)}{\partial \eta} = 0$. This suggests that $\nabla f(w_{k+1})^T d_k = 0$; which is a necessary condition for all minimization directions. Now consider the second order approximation of the ANN error function in the following form:

$$f(w_{k+1}) = f(w_k + \eta_k d_k) = f(w_k) + \eta_k [\nabla f(w_k)] d_k + \frac{\eta_k^2}{2} d_k^T H d_k. \qquad (2.52)$$

Differentiation of the function, after neglecting higher order terms, provides the Equation 2.53:

$$\nabla f(w_{k+1}) = \nabla f(w_k) + [\eta_k H d_k]. \qquad (2.53)$$

At the minimum point $\nabla f(w_{k+1}) = 0$ and pre-multiplying 2.53 by d_k, the important result in Equation 2.54 is obtained, provided that the directions d_k are linearly independent. In other words, the mutual conjugacy condition holds. The expression for η_k is thus given by:

$$\eta_k = -\frac{\nabla f(w_k)^T d_k}{d_k^T H d_k}. \tag{2.54}$$

From 2.53 it is evident, when pre-multiplying by d_k, $[\nabla f(w_{k+1})^T d]=0$ at the minimum point due to the first order condition and consequently $d_k^T H d_k = 0$. This condition is equivalent to 2.51. The iterative expression for the weight update in the conjugate direction is:

$$w_{k+1} = w_k + \eta_k d_k. \tag{2.55}$$

The successive conjugate gradient directions are generated by a linear combination of the current gradient and previous direction with respect to the coefficient $\ddot{\eta}_k$ in equation 2.56:

$$d_{k+1} = -\nabla f(w_{k+1}) + \ddot{\eta}_k d_k. \tag{2.56}$$

The standard BP (Rumelhart et al., 1986) uses this information to include a momentum term (Bishop, 1995). The requirement that successive directions are H conjugate with the coefficients $\ddot{\eta}_k$, is determined from the equation 2.56 by multiplying H to obtain equation 2.57:

$$d_{k+1}^T H d_k = [-\nabla f(w_{k+1}) + \ddot{\eta}_k d_k] H d_k = 0. \tag{2.57}$$

The value of $\ddot{\eta}_k$ which generates conjugate directions can now be found using Equation 2.58:

$$\ddot{\eta}_k = \frac{\nabla f(w_{k+1})^T H d_k}{d_k^T H d_k}. \tag{2.58}$$

This can be interpreted as momentum term according to Bishop (1995).

Conjugate gradient training methods work well on batch training with very careful implementation of line search. The difficulties with line search computations are discussed in (Shanno, 1978 and Luenberger, 1984). Multilayer perceptron training using conjugate gradients can be found in Watrous (1987); Kramer and Sangiovanni-Vincentelli (1989); Makram-Ebeid et al. (1989); Barnard (1992) and Johansson et al. (1992). The studies in Johansson et al. (1992) demonstrate that the conjugate gradient method due to Polak-Ribiere (1969) and Hestenes-Stiefel (1952), perform better than Shanno (1978). The performance of conjugate gradient method deteriorates with poor line search (Oren, 1972). In some cases for badly scaled problems, the matrix D_k may become singular as shown in McCormic and Pearson (1969) due to the rounding off errors (Brad, 1968).

2.11 Source of Difficulties with Standard BP

The direction d_k is computed using the gradient, $\nabla f(w_k)$, information from a single training pattern in standard on line back propagation. In on-line training, the other training patterns compute the gradient components, $\nabla f(w_k)$, which would result in different directions (Kamarthi and Pittner, 1999) for a particular weight. Therefore, a single descent direction is not generated. The fixed value of a learning rate η_k does not always lead to a maximum local decrease of function value. The learning rate depends on the shape of the error function (Jacobs, 1988) as it trains from the current iteration k to the next iteration $k+1$, $k+2$,.... and so on. The iterates therefore, do not produce a convergent sequence in strict sense. The difficulty originates from two different sources (Jacobs, 1988; Weir, 1991 and Vogl, 1988):

a.) the fixed value of learning rate η_k may misdirect the search towards the minimum during iteration k;

b.) the directions d_k generated from $\nabla f(w_k)$ during iteration k are different for a single weight component in standard back propagation training.

Selecting arbitrary learning rates for each parameter the weights are modified, but a steepest descent move is not performed. In this case the parameters are updated based on the partial derivative of the error function with respect to the patterns in the training set. Due to the different gradient information of a weight resulting from different training pattern in the training set, the directions of moves are different. If the valley has twists and turns, large value of η_k will prevent the system from making reasonable progress across a long flat slope (Vogl et al., 1988 and Jang et al., 1997). Choosing a suitable adaptation length in a particular problem involves experimenting with different values of learning rates that reduce the training time (Fahlman, 1988 and Hinton, 1987). Rumelhart et al. (1986) suggests a large value of η_k for rapid learning without oscillations. As a matter of fact, for some step, a large value of η_k may be ideal but there is no guarantee that the same adaptation length would be appropriate for other steps in the learning process (Jacobs, 1988). The optimum value of η_k will depend on the topography of the domain being traversed. If the contours of the error function are circular then the step size will have little influence on the convergence in BP. The convergence difficulties arise when the contours have stiff ridges and the error surface contains local minimums while the contours with elliptical or circular shape will favor convergence (Moller, 1997). The convergence behavior of the BP is dominated by the eigenvectors associated with the largest and least eigenvalues of the corresponding Hessian matrix (Vogl et al., 1988; Bishop, 1995). The fixed learning rate or variable learning rate that do not fully consider the geometry of the error surface would deviate the search from the minimum trajectory of the error surface.

2.11.1 Validation Issues

The issue of validation is closely related to the training methods. Its property in ANN are affected by the following three major factors:

1. The elements y_o^p in error equation 2.6 or 2.13 are considered to be a random sample from a parent population with the expected value $\bar{y}_o^p$ given by the parent distribution. The variations of y_o^p about the expected values $\bar{y}_o^p$ are some time far wider than the expected variance;
2. The error equations 2.6 or 2.13 are assumed to be a smooth functions with respect to all the training weights w_j;
3. The choice of ANN that is an approximation of the true function influences the ANN training problem.

The error contribution due to factor 1 is difficult to control. The only option is to repeat or replace the observations. The weights are determined in BP by gradual adjustment using gradient information. An unfavorable eigenvalue in the Hessian matrix produces complex geometry and causes ill conditioning in the Hessian matrix. In this case the training terminates prematurely. To control the difficulty arising from the second factor, a training method that overcomes this difficulty is desirable. Finally, the different ANN structures can be compared to determine the most suitable functional form of ANN as a representative of the problem being solved (Adya and Collopy, 1997; Zhang et al., 1999 and Atiya et al., 1999).

A relatively new method in statistical analysis proposed by Effron (1982), known as Jackknife or bootstrap method, can be used to test the validity of the ANN learning (Tibshirani, 1996). However, in interval estimate of population parameters, the result is not robust and yields biased values when the population is far from normal distribution (Effron, 1982). The jackknife procedure (Diaconis and Effron, 1983) is robust and does not require the population to have any specific distribution. It is applicable when the normality assumption is not met.

The validation method in ANN applications is an important issue in the context of generalization and decision-making (Zhang et al., 1998). The common approach is to evaluate some statistical error measures (Tibshirani, 1996). A useful validation, which can be regarded as generalization, is to compare performance of ANN in the training and test period using bootstrap sampling method. The data set in the validation period is small and the parameter estimate is not robust. Tibshirani (1996) compares the bootstrap error estimation method with the standard statistical method given in Effron and Tibshirani (1993), Kent (1982) and Effron (1982). The study demonstrates that the bootstrap method gives better estimate.

Another problem in ANN training is the over fitting of a model (Weigend et al., 1990). It occurs when the network has too many free parameters that allow the network to fit the

training data well (Lisi et al., 1999). This leads to poor generalization (Chauvin, 1990, Morgan et al., 1990).

2.12 Summary and Discussions

The standard BP is important from a theoretical viewpoint. It is one of the basic training methods and other improvements are motivated by an attempt to modify the basic BP training method to improve convergence (Sexton et al., 1998; Archer et al., 1993; Hsiung et al., 1990; Lenard at al., 1995; Subraminun et al., 1990; Wang, 1995; Watrous, 1987; White, 1987 and Weigend et al., 1990). Most often it is tried on new problems. The gradient descent BP converges locally but they often become trapped at sub-optimal solutions depending upon the serendipity of the initial random starting point. There have been many attempts to improve the local convergence of the gradient descent BP in Chen and Mars (1990); Franzini (1987); Holt and Semaani (1990); LeCum et al. (1989); Lee and Bien, (1991); Matsuoka and Yi, (1991); Samad (1990), Shoemaker et al. (1991); Sietsma (1991); Solla (1988); Van Ooyen and Nienhuis (1992); and Weigend et al. (1991). The standard BP therefore remains as a standard reference algorithm and the improvement issues are concerned with the acceleration in computations, self-adaptation of learning rate, determination of appropriate momentum term and global convergence.

2.12.1 Comments on the Self-Adaptive First Order Training

The main factor limiting the convergence of gradient descent back propagation is the contours of the error surface that have different shapes in different directions. The contours that are skewed limit the value of the convergence rate; the circular contours favor the convergence rate (Moller, 1997).

The determination of adaptation length and momentum term plays a crucial role in the training process. Mostly the first order gradient descent BP computations (Jacobs, 1988; Weir, 1991; Silva and Almedia, 1990) ignore this by choosing a fixed parameter or its variation. A method that directly uses the local information of the error surface to determine learning rate is more appropriate. The ANN training with line search is difficult since the error function is difficult to bound due to the large number of ANN weights. Fast convergence is realized when a method provides prior information about the optimal parameter setting in training. Therefore, one needs a mechanism that precisely determines this variable learning rate and momentum parameter automatically during training. It suggests that the method generates a suitable direction vector d_k and the correct learning rate η_k in weight space w_k. Dynamically adjusted correct training parameters do not overshoot the minimum. Such a method can be regarded as self-adaptive training.

If computational difficulties are encountered due to ill conditioning in the error function, then the derivative based training algorithms need modifications (Wang and Lin, 1998).

Also, if the derivative information of an ANN error function is not readily available, then the derivative free methods are other alternatives.

Due to the existence of multiple local minimums (Kantz and Schreiber, 1997; Azoff, 1994), it is hard to find the best local minimum. The algorithm that has a global training feature is an important training scheme. Tabu search (Glover and Laguna, 1997) is the recent trend in global optimization and is a potential research issue in ANN computation.

2.12.2 Comments on the Second Order Training

As the ANN grows in size, the structure of the error functions begins to deteriorate. The eigenvalues of the Hessian matrix are wide apart and consequently, the condition number becomes larger (Ahmed and Cross 2000). As a result, stiff ridges and sharp edges begin to form in error surface. The function in several places becomes skewed and the derivative based training methods tend to encounter difficulties. The training algorithms rely on the assumption that the ANN problem has an equivalent error function that is smooth, differentiable and has positive definite Hessian matrix. The training suffers if these conditions are not met. Newton's method converges in the neighborhood of local solution. Newton type second order methods require that the Hessian matrix to be evaluated and inverted or an approximation to its inverse must be evaluated at each iteration. This poses difficulty for large size problem. Second order methods are fast for small and moderate sized problems. The storage and computation requirements for large problems (Wu et al., 1998) prevent Newton type training method to be attractive. Becker and Cun (1989) use variations of a quasi Newton method to speed up back propagation by approximating the second order derivatives. They report that the approximation method show oscillations during convergence.

The second order training algorithms have quadratic convergence. Newton's method becomes impractical because of the size of Hessian matrix, although it has a good quadratic convergence property. Also, there are chances that these training algorithms face ill conditioning. The Marquardt-Levenberg (M-L) algorithm improves such condition by improving the Hessian matrix but also faces difficulty due to the matrix that is far from true Hessian. Quasi Newton, conjugate gradient and other variable metric training algorithms depend on the quadratic and differentiable properties of the error function (Shanno, 1980). Quasi Newton type algorithms improve Newton's method in preserving and updating the Hessian matrix information in a deflected matrix. These methods deflect the gradient in a negative direction using a previous direction of move. This is viewed as an update of fixed symmetric, positive definite matrix in the form of an identity matrix and hence is regarded as fixed metric in contrast with the term variable metric method in quasi Newton methods. Conjugate direction methods converge in at most m iterations for unconstrained quadratic minimization problems in E^m when using exact line search. Davidon (1959), Fletcher and Powell (1963), Kowalik and Osborne (1968), Luenberger

(1984), Pierre (1969), Powell (1964), and Swann (1964) outlines the precautions to be taken when using line search to achieve fast convergence. Conjugate gradient methods provide better choice of directions and also automate the procedure of learning rate or adaptation length by a line search technique. These algorithms do not require adequate storage and are relatively successful in training. They are also considered as some variant of the quasi Newton type algorithms. The worsening condition number in ANN error function put burden in computation even with these algorithms. In general, the second order training algorithms are self-adaptive. McMenamin and Monforte (1998) report results using a Newton type-training algorithm with time series problem. Comparison of the M-L algorithm with other training methods is given in Webb et al. (1988).

Kramer and Sangiovanni-Vincentelli (1989) compare parallel implementations of the BP, steepest descent and conjugate gradient methods using the Polak-Ribiere (1969) method. The experiment suggests that Polak-Ribiere (1969) method performs better than the conventional BP and steepest descent method for small Boolean encored problems and for the parity problem. These algorithms are super-linearly convergent in the neighborhood of a local solution. The algorithm due to Polak-Ribiere (1969) is a variant of the Fletcher-Reeves method while the method of Broydon-Flatcher-Goldfrab-Shanno originates from Devidon's algorithm. These algorithms have been used successfully for simply conducting a descent search (Battiti, 1989). Conjugate gradient methods explicitly construct their searches using linearly independent vectors that span the space.

2.12.3 Research Focus

To resolve some of the difficulties in BP training, we propose to automate the learning rate by using the geometry of the error surface. The ill conditioning put pressure on BP training. To tackle such problems, we propose some derivative free methods. These training methods also have the characteristics of improving solutions in the presence of unfavorable geometry of the error functions and parameter identification problems (Schwefel, 1981). The derivative free training methods can train an ANN with error functions that may be discontinuous. These methods can be developed as parameter free training algorithms. The ill conditioning in training problem also occurs due to scaling of variables as a result of high condition number in the Hessian matrix. It is also noticed in parameter identification problems. These issues are common in ANN computations. Parameter identification is similar to identifying trained weights in ANN (McMenamin and Monforte, 1998; Warner and Misra, 1996). As the smallest eigenvalue approaches a value close to zero, the convergence becomes slow. The performance of the derivative free training is not readily available. Such training methods do not require derivative information, however they adjust ANN weights gradually to approximate the ANN to the actual problem. Hence such training approaches are different than the back propagation training method. In the next chapter these issues are investigated.

Chapter
3

Research Problems and Research Scope

3.1 Introduction

This chapter discusses the research issues and research problems. Theoretical approaches to design the self-adaptive and derivative free ANN training algorithms are briefly discussed. The research design methodologies are addressed.

In Section 3.2, the problems in self-adaptive ANN training algorithms are briefly highlighted. The research issues and research problems are presented in Section 3.3. The test problems are given in Section 3.4. The performance measures of the algorithms are addressed in Section 3.5 and finally some conclusions are provided in Section 3.6.

3.2 Dilemma in Self-Adaptive Training

In the existing self-adaptive training algorithms some preliminary experiments are necessary to determine appropriate learning rate parameters. For example, individual learning rates for each weight are determined separately in Jacobs (1988), Pirez et al. (1993) and Silva et al. (1990). They use different heuristics to adapt the individual learning rates from the information gained in earlier iterations. Jacobs (1988) determines three parameters and the algorithm needs considerable experimentation to identify suitable values of these parameters, while the algorithm proposed by Silva and Almeida (1990) need two parameters but show unstable convergence behavior. According to Jacobs (1988), if the contour surface is relatively flat along a weight dimension, the derivative of the weight is small in magnitude and therefore the weight is adjusted by a small amount. Alternately, where the contour surface is highly curved along a weight dimension, the derivative of the weight is large in magnitude and consequently the weight is adjusted by a large amount with the risk of over shooting the minimum. This happens with the fixed learning rate. If the eignevalues of the Hessian matrix of the error function are far apart, the error function forms contour surfaces that are skewed. In such a case the negative gradient vector in steepest descent training may not point towards the minimum of the contour surface. An arbitrary learning rate is not appropriate for all portion of the contour surface determined in these methods. The component of the gradient vector is smaller in the direction of the eigenvector corresponding to the minimum eigenvalue as compared to the direction of the eigenvector with reference to the maximum eigenvalue (Jacobs, 1988).

The fixed or arbitrary value of the learning rate that produces moderate steps along major axes of the contour surface may produce large steps along minor axes in the weight space and the training exhibits oscillations (Jacobs, 1988 and Bishop, 1995).

Johansson et al. (1992) report a line search in the steepest descent BP ANN training. However, the user determines the learning rate η_k manually. Experimenting with such a method is difficult and arduous. The value of learning rate should be defined such that the successive steps in weight space do not overshoot the minimum of the contour surface. Most often, the value of the learning rate is limited by the magnitude of the largest eigenvalue and only small steps in weight space are taken in the direction of major axis of the contour surface (Jacobs, 1988). The learning rate is either adjusted for all the weights or separately for each weight. This may be a safe but an inefficient approach. These algorithms use some local features of the error function to adjust weight during computation of the error function. As a result the computational efforts are large. In some instances the algorithm cycles and faces difficulty in convergence. In general, slow learning and convergence to a false local minimum is common (Fukuoka et al., 1998).

3.2.1 Research Issues

The aim of the research is to develop ANN training algorithms that automate the selection of the learning rate parameters, the momentum term and the descent directions by monitoring the variations in error surface. Such algorithms provide the basis to adjust dynamically the learning rate parameter in a self-adaptive manner. In contrast, the existing algorithms necessitate the user to undertake experiments with a given problem to identify the suitable training parameters depending on the type of the contour surface under consideration. Hence, the proposed developments relieve the user from pre-optimizing the training parameters.

Consider also the issue of training an ANN that uses a non-smooth transfer function. The research develops training algorithms that train ANN without derivative information in cases where explicit derivative information of an error function is not easily available.

3.3 Research Scope

This research provides a theoretical framework to optimize the learning rate parameter, the momentum term and descent direction in ANN training. The self-adaptive training algorithms that consider the geometry of the error surface in a constrained region are developed. In addition, the development addresses the situations when:

a.) a user need not pre-optimize the learning rate parameter or need not select learning rate parameter at all;

b.) one needs a training algorithm that does not require the derivative information of the error function (Conn et al., 1997);

c.) it is difficult to provide analytical expression of the derivative of the error function;

d.) there is discontinuity in the error function;

e.) the contour surface develops stiff ridges and results in an ill conditioned Hessian matrix which makes ANN training difficult;

f.) the global convergence is an important issue. The proposed derivative free training methods initiates research directions in global search. The recent development in this line is the tabu search first proposed by Glover and Laguna (1997).

3.3.1 Research Problems

The following sections briefly present the self-adaptive, the multi-directional and restart derivative free training algorithms and the methodologies to derive the self-adaptive training methods are addressed.

3.3.1.1 Self-Adaptive Training With Gradient Information

The research aims at determining the learning rate η_k and the descent direction d_k in Equation 3.1 such that an optimized learning rate and the descent direction is identified during training iteration or epoch k. The parameter η_k is determined by an interpolation search in a constrained space where the geometry of the contour surface is examined. The error surface is sampled at discrete intervals and the function values are evaluated to identify the appropriate learning rate. The error function $f(w)$ at iteration k is perturbed by an amount η_k along a chosen direction d_k and the consequence of the change is observed. The method finds η_k such that the error function $f(w_k+\eta_k d_k)$ is minimized. The problem is formulated as:

$$\eta = \arg\left\{\text{minimize} f(w_k+\eta_k d_k)\right\} \qquad (3.1)$$

$$\text{Subject to: } L=(\eta : \eta_k \geq 0)$$

where, L is the constrained interpolation search that determines the learning rate, which is restricted in sign. The search terminates at $\eta=\eta_k$ when the function $f(w_k+\eta_k d_k)$ is minimized with respect to η_k. An interpolation search method is used to find an optimized learning rate. Given the vectors w_k and d_k, the value of η_k is varied such that the value of

the function $f(w_k + \eta_k d_k)$ is changed. As η_k is varied, a function $f(\eta_k)$ of the learning rate is formed. The search involves bracketing a minimum with other two relative minimum points in the neighborhood of the newly transformed error function. An interpolation through the trial points picks up the appropriate learning rate. Since this parameter reduces the error function $f(w_k + \eta_k d_k)$, an optimized learning rate is, therefore, identified. The learning rate parameter along with the direction vector d_k updates the ANN weights. Constantly it provides descent to the error function, during all epochs of training.

3.3.1.1.1 Central Difference Approximation of Descent Direction

The ANN training problem needs information on the descent direction d_k. In standard gradient descent BP, it is the negative gradient of the error function. Here we develop a central difference approximation scheme that implicitly identifies the search vector and hence the direction d_k is computed implicitly. The rate of convergence of the error function and the information of the updated weights are used to choose the central difference step size that controls the accuracy in d_k. This allows the ANN training method to move along the descent direction so that the training converges to a limit point that is a minimizer of the error function.

3.3.1.2 Self-Adaptive Multi-Directional Derivative Free Training Algorithm

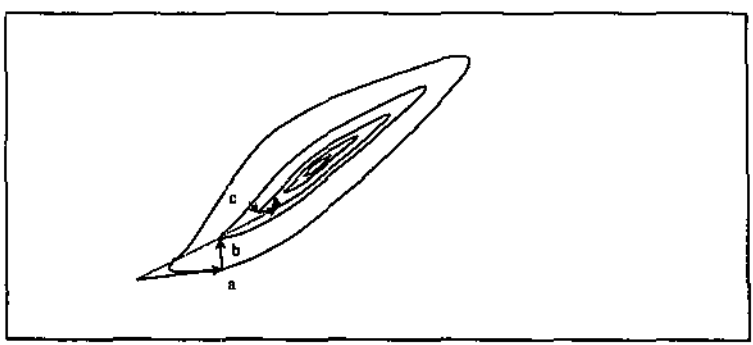


Figure 3.1 Convergence Difficulty and Premature Termination

A multi-directional training algorithm that does not require the derivative information of the error function is proposed. In addition, the training algorithm is designed with an oriented search vector to improve the training performance. The initial trial vector is first

positioned in a suitable location at the beginning of training. This is done at most in m steps, before the multi-directions are explored.

Given a descent error function $f: w \in E^m$, the multi-directional training algorithm reduces the error function $f(w)$ gradually, by changing its weights with respect to the directions, $d_j \equiv (d_1, d_2, \ldots\ldots, d_m)$, where d_j is a vector of zeros, except 1 at the j^{th} position. The value of the weight w_j is therefore changed in the d_j direction, while all other weights are kept fixed. Moving in all m directions the algorithm changes all the value of the weights $(w_{j=1}, w_{j=2}, \ldots\ldots, w_{j=m})$ in m directions. The process is again repeated to obtain change in the function value $f(w_k)$ for the next iteration k. Given an initial weight vector w_k, the algorithm chooses a search direction d_j and the learning rate η_j for a given value of j, where $j = 1,2,\ldots,m$. The selection of η_j can take variety of forms. An interpolation search is performed in a constrained space to determine the magnitude of the learning rate. The interpolation search is defined as:

$$\text{minimize} f(w + \eta_j d_j) \tag{3.2}$$

$$\text{Subject to } \eta_j \in L$$

where, L has the form: $L = E^1$. The defined training problem explores in m different directions to determine the m individual and variable learning rates. The network weights have different learning rates according to this algorithm. To accelerate the training performance, a momentum search, which is similar to the pattern move, is designed. The length of the momentum term is determined in a self-adaptive manner. The dotted line in Figure 3.1 defines a momentum search.

Also, notice the search path when it moves from point a to b in Figure 3.1. If the point b is nearly sharp-edged, the search may cease at b since the derivative is not defined. Making a momentum search along the dotted line, the search is continued to point c and beyond until the search reaches the local minimum. This step is taken in the proposed multi-directional search algorithm. Also observe the shape of the contour surfaces, which is to some extent skewed. To reach a minimum point in such geometry, the training algorithm would require extra effort due to the ridge like or curved structure.

3.3.1.3 Derivative Free Restart Training Algorithm

The simplex method proposed by Spendley, Hext and Himsworth (1962) is improved as a derivative free training algorithm. In factorial design, the number of trials for experimental identification method is about $(m+1)$. These $(m+1)$ equally spaced points are allowed to

form a pattern or geometry that is known as regular simplex. The research improves the method that maintains a set of $(m+1)$ points in m dimensional space to generate non-regular and non-degenerate simplex. When the simplex degenerates, a re-scale phase is performed with a restart vector that forces the simplex to search a wider parameter space.

The proposed algorithm evaluates the error function along all the vertices of the simplex. The search method replaces a vertex with the largest objective function value by a new vertex situated at a *reflection point* midway between other m vertices to find the descent directions. This principle locates a new vertex at best minimum point. The midway point plays the role of a centroid. It provides the descent direction to the error function. This newest vertex can also be reflected to explore the best point in the neighborhood. Three main strategies are defined. They are called *reflection, expansion and contraction*, which generate the directions of search. If the current new vertex improves the function value, then the highest function value is replaced by the improved function value. The proposed algorithm redefines a new simplex with lower function value at the vertex of a degenerated simplex. The edge length of the polyhedron is changed so that the search does not stagnate. The polyhedron is forced to change the size and direction of search. A non-degenerate simplex that has finite volume is reflected with a new search vector to improve the convergence of the algorithm.

3.4 Test Problems and Experimental Set up

The research provides theoretical analysis of convergence of the newly developed algorithms where possible. Some test problems are considered to observe the performance of the algorithms namely:

a.) the XOR problem with 2-2-1 ANN and parity problem with 5-5-1 ANN configurations;

b.) character recognition with the letters L and T with different orientations in 3x3 pixel;

c.) seasonal time series problem: Australian peak electric load forecast and

d.) hotel occupancy rate in Australia as multivariate statistical analysis with small data set.

The purpose of these problems is to test convergence of the algorithms in classification problems and its ability to replicate results as a forecast and regression model. The standard back propagation algorithm is used to compare the results. Some published results from literature are also used to compare the performance of the algorithms. Additionally, the standard statistical regression method (Mendenhall and Sincich, 1996) is used to compare the results of the last two test problems. All the ANN architectures use a

log transfer function in the hidden layer neurons and the output layer uses a constant function.

3.4.1 Parity Problem

The Parity problem is a standard benchmark against which performance of an algorithm can be measured. There are a number of reports on this problem in literature (Johansson et al., 1992; Jacobs, 1988, Kamarthi et al., 1999 and Salomon et al., 1995). In the parity problem we have a number of boolean input variables and one boolean output variable. The input/output rules states that the output should be true in case an odd number of input values are true. If there are just two input variables the problem is known as exclusive OR (XOR). This rule states that either of the inputs can be true but not both.

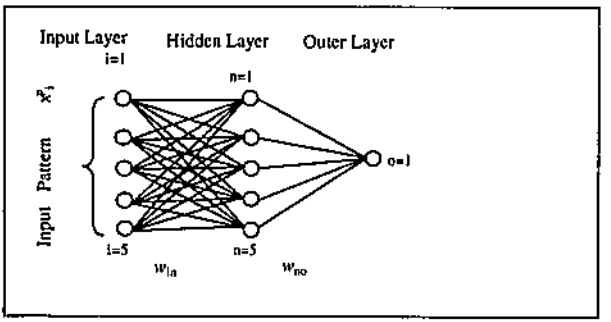


Figure 3.2 Three layer feed forward 5-5-1 ANN

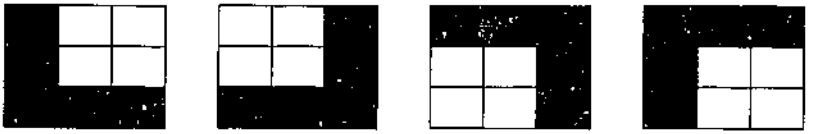
Figure 3.2 shows the ANN that contains 5 neurons in the input layer, 5 neurons in hidden layer and one neuron in output layer. The 5-5-1-configuration network is chosen with full training set for each problem. For example, there are 2^5 training pattern in a 5-5-1 ANN configuration. The training data set is shown in Table A.1 (Appendix A). The pattern of 0 and 1 values are placed on the input layer neurons. The network then produces a 1 if there are an odd number of 1 bits in the input and a 0 if there are even numbers of 1 bits in the input. The capability of the algorithm to solve the parity problems is investigated.

3.4.1.1 Starting Points for the Parity Problem

The starting points for the problem for 10 experiments are shown in Table A.2 (Appendix A). The weights have low as well as high magnitude to observe the performance of the algorithms with different staring vectors. To create starting points in small magnitude, the weights given in Table A.2 (Appendix A) can be factored appropriately to create weights in different magnitudes.

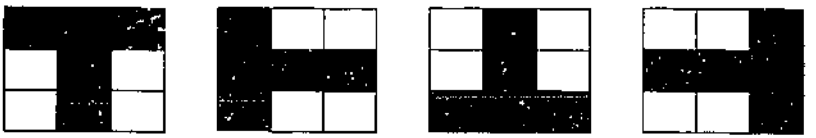
3.4.2 Pattern Recognition Problem

A training experiment is considered to recognize the letters L and T. An ANN with nine inputs, two hidden units and a single output unit is trained to recognize the letters L and T. Each input pattern consists of a 3x3 pixel binary image of the letter. The training set is formed by four orientations of each letter as shown in Figure 3.3 and Figure 3.4. The target values are 0 and 1 respectively at the output unit, which identifies the letters L and T. The training data set is shown in Table A.3 and A.4 (Appendix A). The 9-2-1 ANN configuration is chosen to compare the results given in Kamarthi and Pittner (1999).



(a) (b) (c) (d)

Figure 3.4 Four Orientations of the Letter L



(a) (b) (c) (d)

Figure 3.4 Four Orientations of the Letter T

3.4.3 Seasonal Time Series Problem

A seasonal time series problem is considered as shown in Figure 3.5. In particular, quarterly seasonal peak electric load data in mega watt-hours is collected from Australian Bureau of Statistics time series data. The data set contains the values of peak electric load,

y_s^p, from September 1976 to September 1998 and is shown in Table A.5 (Appendix A). To model the behavior of the seasonal time series data, an ANN with five neurons in the input layer and five neurons in the hidden layer and one neuron in the output layer is proposed. The ANN configuration is similar to the problem discussed in section 3.4.1. The output layer produces the estimated value for pattern p. The input layer neurons receive the following input pattern x_i^p:

1. x_1^p = a bias term for the input layer neuron
2. $x_2^p = p$, the pattern at instance p
3. $x_3^p = \begin{cases} 1 & \text{if quarter 2 falls at instance } p \\ 0 & \text{otherwise} \end{cases}$
4. $x_4^p = \begin{cases} 1 & \text{if quarter 3 falls at instance } p \\ 0 & \text{otherwise} \end{cases}$
5. $x_5^p = \begin{cases} 1 & \text{if quarter 4 falls at instance } p \\ 0 & \text{otherwise} . \end{cases}$

The purpose of this exercise is to notice the performance of the training algorithms in convergence. A 5-5-1 ANN model is chosen to represent and predict Australia's quarterly peak electric load. The algorithm is tested to measure its strength in the forecasting problem. A reasonable procedure in forecasting validation/generalization is to split the available data into two parts, which Snee (1977) calls the estimation data and the prediction data. The estimation data is used to build the ANN model and the prediction data is then used to study the predictive ability of the model. Sometimes data splitting is called cross-validation (Mosteller and Tukey, 1968 and Stone, 1974). In a large data set, the data can be separated into three parts. The third component of the data set is the testing set, which could comprise the entire data set.

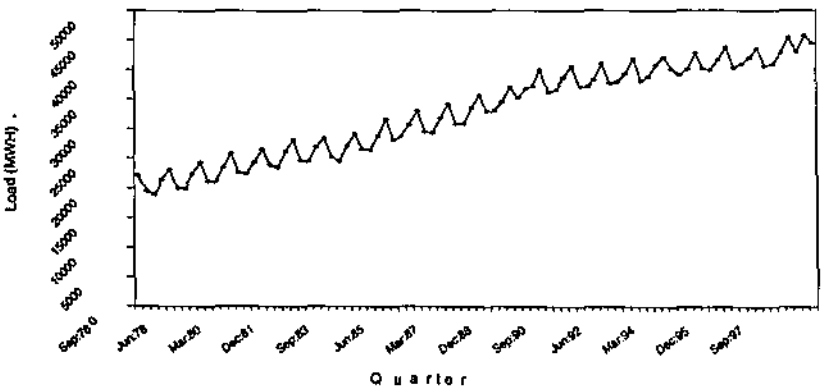


Figure 3.5 Peak Electric Load

The data set from September 1976 to June 1993 is used to train the ANN. Sixteen quarterly data from September 1994 to June 1997 are used to measure the forecasting capability. The standard statistical regression method (Mandenhall and Sincich, 1996) is used as a benchmark to compare the performance of the proposed algorithm.

3.4.4 Hotel Occupancy Rate

Table A.6 in Appendix-A shows the limited tourism data that is available from an ABS report. Eight quarterly data from March 97 through December 1998 is used to train the ANN model. It is possible to work with the small data set in ANN modeling (Law, 1998). Figure 3.6, 3.7 and 3.8 displays the pattern of the data set including room night spent, consumer price index (CPI) and gross domestic product (GDP) respectively. The training method investigates the capability of the algorithm to model the small seasonal data set as the multivariate statistical analysis problem and its potential as an interpolation or calibration model. All the data points are used for training and the trained ANN is used to compare the in-sample extrapolation capability. The multivariate statistical method is used as a benchmark to compare the performance of the algorithm.

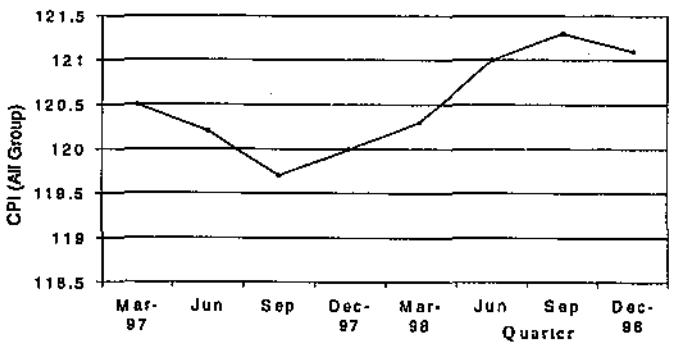


Figure 3.6 CPI All group

3.4.4.1 ANN Model

The room-nights spent constitutes the hotel occupancy rate y_o^p. To develop an ANN model using data set in Table A.6 (Appendix A) the first layer neurons are parameterized and the input to the neurons is described below. A 7-4-1 ANN configuration is chosen to study the performance of the algorithms. The hidden layer transfer functions are similar to the problem in section 4.1. The input layer neurons receive the following input patter x_i^p.

1. x_1^p = a bias term as input for the input layer neuron
2. $x_2^p = p$, the pattern at instance p
3. x_3^p = CPI (Consumer price Index)
4. x_4^p = GDP (Gross Domestic Product)
5. $x_5^p = \begin{cases} 1 & \text{if quarter 2 falls at instance } p \\ 0 & \text{otherwise} \end{cases}$
6. $x_6^p = \begin{cases} 1 & \text{if quarter 3 falls at instance } p \\ 0 & \text{otherwise} \end{cases}$
7. $x_7^p = \begin{cases} 1 & \text{if quarter 4 falls at instance } p \\ 0 & \text{otherwise} \end{cases}$

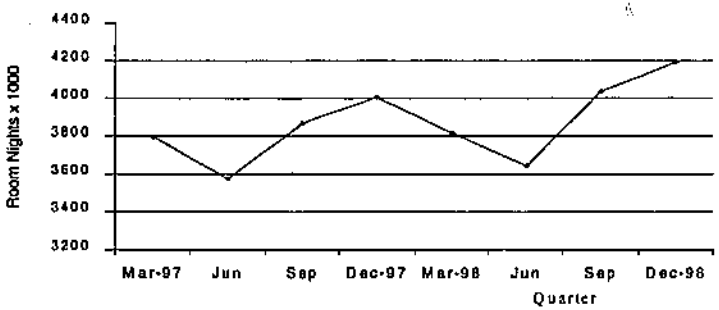


Figure 3.7 Room Nights Spent

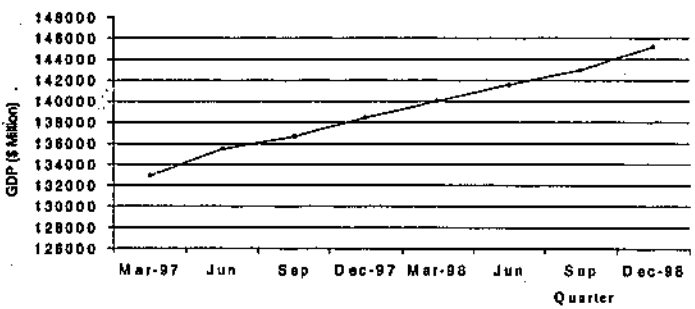


Figure 3.8: GDP $M Illion

3.4.4.2 The Starting Points

The random starting points are according to the weights shown in Table 3.2 in which the last two weights are repeated to complete the total weights for initialization, defined as $w_{31} = w_{20}$ and $w_{32} = w_{30}$.

3.5 Performance Measure of the Algorithms

To compare the performance of the algorithms we require some kind of metric. The number of iterations or cycles or epochs is not a valid metric (Johansson et al., 1992). In BP, the training set is presented through the network once per iteration. In second order BP, the training set is presented several times per iteration and most of the computation time is spent in evaluating the error function and the gradient.

The conventional BP training requires one forward and backward propagation of the signal per iteration. The second order algorithm on the other hand may involve several forward and backward propagations at each iteration. Depending on the line search method, the number of forward and backward propagations may or may not be the same (Johansson et al., 1992). There are a variety of line search methods and some use function evaluation and gradient at each step while some use only function value or gradient. The methods that use only gradient information have the advantage of rapid convergence (Al-Sultan et al., 1997). The number of gradient evaluations is not usually reported when the performance measures are addressed.

An algorithm that starts in the vicinity of a minimum converges rapidly. To eliminate this biased performance, the algorithms are tested on random starting points. We propose to compare the algorithms based on average performance of the algorithms. In particular we address the following measures:

a.) number of iterations or epochs;
b.) number of gradient evaluations;
c.) total number of function evaluations including gradient evaluations;
d.) standard deviations of the performances on the above measures;
e.) maximum, minimum, median and range on the above measures and
f.) terminal error function value at the end of training as squared error measures.

The median value divides an ordered set of values into two halves. The maximum and minimum values indicate the worst-case performance through the measure called the range. The convergence of the algorithm depends on the starting point and as the algorithms are tested on random problems, the standard deviation measure is a good indicator to judge the consistent performance of an algorithm. There would be some instances where the training may not converge. Under such situations, the algorithm is

forced to terminate after it has completed 50,000 numbers of epochs and the statistics are collected within this period of training.

3.5.1 Performance Measure in Forecasting and Multivariate Analysis

To see the performance of the algorithm in seasonal time series forecasting and multivariate analysis, the following measures are considered:

a.) mean squared error;
b.) mean absolute percentage error;
c.) mean percentage error and
d.) R^2 test.

It should be noted that a method to interpret the ANN weight similar to the multivariate statistics is not yet developed. Nevertheless, the ANN computations provide a model in multivariate statistical analysis to represent and replicate the data.

As usual, the performance of the algorithm to this class of problems is first evaluated according to the experimental set up in section 3.5. The experiment that finds the minimum error function value is considered for forecast and multivariate statistical analysis.

3.5.2 Termination Criteria of the Algorithms

The termination of the algorithm is based on the following criteria:

$$f(w_k) - f(w_{k-\hat{k}}) \leq \varepsilon \qquad (3.3)$$

where, $\hat{k}$ is set to 3. The convergence is checked when $k \geq 3$. This implies that the function improvement in three consecutive iterations, $\hat{k}$, is insignificant. The training is terminated when the relative improvement in error function is less that $\varepsilon = 10^{-10}$ in three successive iterations. A maximum limit is also set for the total number of iterations/epoch as 50,000. The algorithm terminates if it does not converge within this allowed iteration.

3.6 Final Remarks in Research Design

The focus of the research is to provide a theoretical framework on the development of the self-adaptive and derivative free training algorithms. Some test problems have been selected only to test the performance of the algorithms. The selected problems address a variety of characteristics of the self-adaptive and derivative free training algorithms. In particular the performance and strength of the algorithms to solve problems in:

a.) classification;
b.) pattern recognition;
c.) forecasting and
d.) multivariate statistical analysis

are evaluated. This will identify the strength and weakness of the training algorithms.

Random starting points in a small and wide range are used to observe the sensitivity of the algorithm to solve different test problems. The objective of the research, however, is not to solve a variety of problems and different ANN structures. The focus is the development of self-adaptive and derivative free ANN training algorithms. The proposed problems are used to test the efficiency of the algorithms in different classes of problems and its convergence performance. The data sets for the problems are used without transformation. Hence, the ANN configuration, initial starting points and data set are standardized. The objective is to study the performance of the algorithms, which require minimum human intervention with the data set or ANN configuration or training parameter selection.

Chapter
4

Self-Adaptive Back Propagation Training

4.1 Introduction

A gradient descent self-adaptive back propagation training method, which dynamically adjusts variable learning rate is developed in this chapter. A specialized interpolation search is developed to determine the optimized learning rates that do not over shoot the minimum. A central difference gradient approximation scheme is developed to provide descent direction in training. The descent directions are controlled by the convergence rate of the error function. The training method is derivative free in the sense that the derivative information of the error function is provided by the central difference gradient approximation scheme implicitly rather than explicitly. The algorithm solves XOR problems and results are reported.

The directional vector and self-adaptive learning rate parameter that provide appropriate descent to the error function are developed in Section 4.2. The interpolation search in constrained space provides the appropriate learning rate. Hence an arbitrary or an ad hoc method to select the learning rate is abandoned. The algorithm is given in Section 4.3. A central difference approximation scheme that uses convergence properties of the error surface is developed in Section 4.4. The convergence of the error function controls the central difference step size to compute suitable descent directions. The algorithms that generate these vectors are presented. In Section 4.5, the first order gradient descent back propagation training algorithm is developed using the descent direction vector and the self-adaptive learning rate parameter. Section 4.6 shows that the algorithm generates a convergent sequence of the network weights and terminates at a point, which is a minimizer of the error function. The XOR problem is used for the analysis of the algorithm in Section 4.7. Section 4.8 provides related discussions.

4.2 Definitions in ANN Computations

To facilitate the development of the self-adaptive gradient based training algorithm some definitions and properties of the ANN error function are introduced in the following sections. The aim is to develop self-adaptive learning rate and implicit descent directions for efficient computations in ANN.

4.2.1 Error Function

The notation $f(w)$ represents the error function, which corresponds to the equation 2.6 or 2.13 in Chapter 2. The representation $f(w_j)$ means that the error function is consisting of $w_j, j=1,2,...,m$ number of weights. Hence the notations $f(w)$ and $f(w_j)$ are equivalent and some times used interchangeably for convenience.

4.2.2 Error Surface

The locus of the ANN error functions with respect to $w_j \equiv (w_1, w_2, ..., w_m)$ forms a geometry that is defined as the *response surface or error surface*. For example in E^2, the intersection of the plane parallel to the w_j plane and the response surface constitutes *contours* with constant function values. If $f(w_j)$ is continuous, the contours will be connected, continuous and smooth curves.

4.2.3 Algorithmic Map and Iterative Process

An algorithm contains a set of instructions that performs a defined task. Applying an instruction on the error function, the current weight vector w_k during iteration or epoch k is changed to a new vector w_{k+1}. The notation w_k represents the vector of weights during iteration k. The instruction generates a sequence of weight vectors with reference to a vector preceding it. The instruction that generates the vector is called an *algorithmic map*. An algorithmic map M can be described by:

$$w_{k+1} \in M(w_k). \tag{4.1}$$

Given an initial weight vector w_k, the *algorithmic map* M generates a sequence of vectors: $w_{k+1}, w_{k+2}, ...$, and the process of generating this sequence is called an *iterative process or iterative algorithm.*

4.2.4 Sequence

A sequence of vectors w_1, w_2, w_3 ...., is said to converge to the limit point w^*, if $\| w_k - w^* \| \to 0$ as $k \to \infty$. Alternately, for any given $\varepsilon > 0$, there exist a positive integer k_1 such that $\| w_k - w^* \| < \varepsilon$ for all $k \geq k_1$. The sequence is denoted by $\{w_k\}$. The limit point w^* is represented by $w_k \to w^*$ as $k \to \infty$ or $\lim_{k \to \infty} w_k = w^*$.

4.2.5 Subsequence

A subsequence is obtained by dropping certain elements of a sequence $\{w_k\}$. A subsequence is represented as $\{w_k\}_{k_1}$, where k_1 is the subset of all positive integers, say k. Suppose that $\{w_4, w_6, w_{12}, w_{16}.....\}$ denotes a subsequence in $\{w_k\}_{k_1}$, then the notation $\{w_{k+1}\}_{k_1}$ also denotes a subsequence represented as $\{w_5, w_7, w_{13}, w_{17}.....\}$ by adding 1 to the indices of all the elements in the sequence $\{w_k\}_{k_1}$.

4.2.6 Neighborhood

Given a relative local minimum $w^* \in E^m$ and an $\varepsilon > 0$, the ball $N_\varepsilon(w^*) = \{w : \|w - w^*\| \leq \varepsilon\}$ is called the ε neighborhood of w^*.

4.2.7 Global Convergence and Closed Map

When an iterative algorithm is applied to an error function with an initial arbitrary weight vector w_k, at the beginning of iteration k, the algorithm generates a sequence of vectors w_{k+1}, w_{k+2},... during iteration $k+1$, $k+2$,....,.., and so on. The iterative algorithm is *globally convergent* if the sequence of vectors converges to a solution set Ω. Consider for example the following training problem, where w is defined over E^m:

minimize $f(w)$ (4.2)

subject to: $w \in E^m$.

Let, $\Omega \in E^m$ be the solution set, and the application of an algorithmic map, A, starting with w_k generates the sequence w_{k+1}, w_{k+2}..... such that $(w_{k+1}, w_{k+2}.....) \in \Omega$, then the algorithm converges globally and the *algorithmic map* is *closed* over Ω.

4.2.8 Descent Function

To define a descent function, consider the minimization problem stated above. Let Ω be a non-empty compact subset of E^m, and if an algorithmic map generates a sequence: $\{w_k\} \in \Omega$ such that $f(w)$ decreases at each iteration while satisfying $f(w_k) > f(w_{k+1}) > f(w_{k+2})$,...., then the function $f(w)$ is said to be a *descent function*. In

ANN computation the error function $f(w)$ is assumed to be a descent function possessing convexity property due to the results given in Hecht-Nielsen (1990). Using this result we can define a descent direction along which the error function can be minimized and state the following proposition to demonstrate this.

Proposition 4.1

Suppose that $f: E^m \to E^1$ *and the gradient,* $\nabla f(w)$, *is defined then there is a vector* d *such that* $\nabla f(w)^T d < 0$, *and* $f(w+\eta d) < f(w): \{\eta \in (0,\delta), \delta > 0\}$, *then the vector* d *is a descent direction of* $f(w)$.

Proof

Expanding the error function $f(w)$ by Taylor series and neglecting higher order terms, the following expression can be obtained:

$$f(w+\eta d) = f(w) + \eta \nabla f(w)^T d \,. \tag{4.3}$$

and therefore:

$$\frac{f(w+\eta d)-f(w)}{\eta} = \nabla f(w)^T d \,. \tag{4.4}$$

Since $\nabla f(w)^T d < 0$, then for $\delta > 0$ and $\eta \in (0,\delta)$ we get:

$\frac{f(w+\eta d)-f(w)}{\eta} < 0$, it follows then $f(w+\eta d) \le f(w)$. This implies that d is the descent direction in ANN computation $\Diamond$.

4.2.9 Directional Derivative

In ANN computation the direction along which the error function decreases is conceptualized by a vector defined as *directional derivative* as shown in the following section.

Let, $f: E^m \to E^1$, $w \in E^m$ and d is a non-zero vector satisfying $(w+\eta d) \in E^m$, $\eta > 0$ and $\eta \to 0^+$. The *directional derivative* at w along the descent direction d is given by:

$$\nabla f(w;d) = \lim_{\eta \to 0^+} \frac{f(w+\eta d)-f(w)}{\eta} \,. \tag{4.5}$$

Assuming the error function is smooth and continuous, we can define its directional derivative from the following proposition.

Proposition 4.2

Let $f: E^m \to E^1$ is a descent function. Consider any point $w \in E^m$ and $d \in E^m$: $d \neq 0$. Then the directional derivative $\nabla f(w;d)$ of the error function $f(w)$ in the direction d always exists.

Proof

Let η_1 and η_2 are two arbitrary quantities that denotes the learning rate such that $\eta_2 > \eta_1 > 0$, since $f(w)$ is a descent function and posses convexity property (Hecht-Nielsen, 1990), we get the following expressions:

$$f(w+\eta_1 d) = f[\frac{\eta_1}{\eta_2}(w+\eta_2 d)+(1-\frac{\eta_1}{\eta_2})w] \qquad (4.6)$$

$$\leq \frac{\eta_1}{\eta_2} f(w+\eta_2 d)+(1-\frac{\eta_1}{\eta_2}) f(w).$$

This inequality implies that: $\frac{f(w_k+\eta_1 d)-f(w)}{\eta_1} \leq \frac{f(w+\eta_2 d)-f(w)}{\eta_2}$.

Thus, in general, $[\frac{f(w+\eta d)-f(w)}{\eta}]$ always decreases as $\eta \to 0^+$. Again due to descent property of the function $f(w_k)$ for $\eta > 0$, we have:

$$f(w) = f[\frac{\eta}{1+\eta}(w-d)+\frac{1}{1+\eta}(w+\eta d)]$$

and hence:

$$\leq \frac{\eta}{1+\eta} f(w-d)+\frac{1}{1+\eta} f(w+\eta d).$$

Simplifying the above expressions we get:

$$\frac{f(w+\eta d)-f(w)}{\eta} \geq f(w)-f(w-d). \qquad (4.7)$$

It implies that we have a sequence generated as: $[\frac{f(w+\eta d)-f(w)}{\eta}]$ that converges as $\eta \to 0^+$ and $k \to \infty$ and this convergence is bounded by $[f(w_k)-f(w_k-d_k)]$ from below. Therefore, the directional derivative is given by $\nabla f(w;d) \equiv \lim_{\eta \to 0^+} \frac{f(w+\eta d)-f(w)}{\eta}$ $\Diamond$.

4.3 The Interpolation Search Map

The central theme behind the BP computation is the computation of the search vector d and the learning rate parameter η in weight space towards the minimum point. Since the exact location of the minimum point is not known, there is an uncertainty in identifying the boundary or region in weight space over which the search may explore. The uncertainty can be reduced if we can eliminate the sections of search boundary (Kiefer, 1953), which do not contain the minimum through an interpolation search in a constrained interval. Therefore, what we need is a search map that explores the constrained region of the error surface. The search map samples the function value on the error surface in discrete length with a given direction. The following definitions are needed to describe the interpolation search map. For convenience we define the notation:

$$u_k \equiv w_{k+1} = w_k + \eta_k d_k \text{ or } u = w + \eta d . \tag{4.8}$$

Now consider a training problem with ANN error function defined by:

$$\eta_k \equiv \arg\{\min f(w+\eta d)\} \ ; \ \text{subject to}: \eta_k \in L \tag{4.9a}$$

in a closed interval $L = \{\eta : \eta \in E^1\}$. The interpolation map is defined as $A : E^m \times E^m \to E^m$ such that:

$$A(w,d) = \{u : u = (w + \eta\ d) | \eta\} \ \in L \tag{4.9b}$$

$$f(u) = \min f(w + \eta\ d) \ ; \ \text{Subject to}: \eta^* \equiv \eta \in L . \tag{4.9c}$$

The map A produces the descent directions. The map should be closed as set value mapping (Luenberger, 1984) such that an appropriate learning rate is obtained. The following proposition shows that the map A is closed over the interpolation search.

Proposition 4.3

Let $f : E^m \to E^1$ be the error function. Then the interpolation search map defined above is closed at $A(w,d)$ if $d \neq 0$ over the defined search interval $L = \{\eta : \eta \in E^1\}$.

Proof

Let k be the iteration counter and suppose that the sequence $\{w_k\}$ and $\{d_k\}$ are such that $w_k \to w^*$, $d_k \to d^*$ and $d^* \neq 0$ so that the search is active where w^* and d^* are considered

as the limit points. Let $u_k \in A(w_k, d_k)$ and $u_k \to u^*$ as limit point. The mapping is closed when $u^* \in A(w_k, d_k)$ and this is what we need to demonstrate. Now consider an error function that has produced a sequence of $\{u_k\}$ as $k \to \infty$. The sequence of iterations is generated as: $\{u_k\} = (w_k + \eta_k d_k)$ provided that $d_k \neq 0$. The learning rate parameter can be expressed as $\eta_k = \frac{w_k - w_k}{d_k}$. We know that $f(w)$ is a descent function in BP, therefore, as $k \to \infty$, $\eta_k \to \eta^*$. The notation η^* is the limit point. Also let $u_k \to u$, $w_k \to u$ and $d_k \to d$ as $k \to \infty$. It follows then, $\eta^* = \frac{u - w}{d}$ or alternately, we have $u = w + \eta^* d$. Suppose during iteration k, $\eta_k \in L$, then $\eta^* \in L$ and $u_k \to u^*$ as $k \to \infty$. Now denote $u^* \bullet u_{k+1}$, for the training function, as $k \bullet \bullet$. The mapping, therefore, satisfies $f(u_{k+1}) \leq f(w_k + \eta_k d_k)$. Consequently we have, $f(u_{k+1}) \leq f(w_k + \eta^* d_k)$ and hence $\{u_k\}$ is a convergent sequence. The sequence $\{u_k\}$ is derived from the map A and therefore, $u_k \in A(w_k, d_k)$. It follows that the map is closed ◊.

4.3.1 Interpolation Search by Sampling Error Surface

The requirement that $d \neq 0$ is important in the search. If $d = 0$, then theoretically algorithm must have converged to minimum point or no search direction is generated. Now consider the following training problem:

$$\text{minimize} f(w), \quad \text{Subject to}: w \in E^m \tag{4.10a}$$

and transform the problem in the form:

$$\eta^* = \eta_j = \arg\left\{ \min f(w_j + \eta_j d_j) \right\} \tag{4.10b}$$

$$\text{subject to}: \eta \in L.$$

The problem is solved; setting the value of the vectors w_j and d_j, while the value of η_j is adjusted such that $f(w_j + \eta_j d_j)$ is minimized. As $\eta \leftarrow \eta_j$ is varied, $f(\eta)$ forms an equivalent error function for the given value of w_j and d_j. To determine the learning rate evaluate the training function at three different points η_1, η_2 and η_3 as $f(\eta_1)$, $f(\eta_2)$ and $f(\eta_3)$ respectively. The search involves bracketing the minimum learning rate with other relative minimum points in a close neighborhood. An interpolation through these points forms a convex function that picks an approximate learning rate parameter as shown in the following proposition.

Proposition 4.4

Let $\dot{\eta}_1, \dot{\eta}_2, \dot{\eta}_3$ *be the three relative learning rate parameters defined over a function* $f(\eta_j)$ *such that* $\dot{\eta}_2 = (1+a_1)\dot{\eta}_1$, $\dot{\eta}_3 = (1+a_2)\dot{\eta}_1$ *and* $\eta^* = (1+a_0)\dot{\eta}_1$. *The quantity* a_0, a_1, a_2 *are such that* $f(\dot{\eta}_2) < f(\dot{\eta}_1)$ *and* $f(\dot{\eta}_3) > f(\dot{\eta}_2)$. *Given a search map at current iterate* k, w_j *and* d_j, *the optimum learning rate to the problem:* $\min f(w_j + \eta_j d_j)$, subject to $\eta_j \in L$ *is given by:*

$$\eta^* = \left(1 + \frac{1}{2}\frac{(a_1^2 - a_2^2)f(\dot{\eta}_1) + a_2^2 f(\dot{\eta}_2) - a_1^2 f(\dot{\eta}_3)}{(a_1 - a_2)f(\dot{\eta}_1) + a_2 f(\dot{\eta}_2) - a_1 f(\dot{\eta}_3)}\right)\dot{\eta}_1 . \tag{4.11}$$

Proof

Select a search direction d_j for a given j and set $\dot{\eta}_1 \leftarrow w_j$ to determine $f(\dot{\eta}_1)$. Now the value of $\dot{\eta}_2$ is expressed as $\dot{\eta}_2 = (1+a_1)\dot{\eta}_1$ with respect to an appropriately defined percentage factor a_1 such that the condition: $f(\dot{\eta}_2) < f(\dot{\eta}_1)$ holds. It implies that the value of $\dot{\eta}_2$ is a_1% greater than $\dot{\eta}_1$. If the condition $f(\dot{\eta}_2) < f(\dot{\eta}_1)$ is not satisfied at this stage, a lower value of $\dot{\eta}_2$ with respect to $\dot{\eta}_1$ is generated. The gradual adjustment of the parameter forces the condition $f(\dot{\eta}_2) < f(\dot{\eta}_1)$ to exist. The search verifies the condition:

$$\dot{\eta}_2 = (1+a_1)\dot{\eta}_1 : f(\dot{\eta}_2) < f(\dot{\eta}_1). \tag{4.12}$$

Next the point $\dot{\eta}_3$ is determined as $\dot{\eta}_3 = (1+a_2)\dot{\eta}_1$, such that $f(\dot{\eta}_3) > f(\dot{\eta}_2)$. This is done gradually increasing the value of a_2 in comparison with the value of a_1. The trial process finally identifies the condition:

$$\dot{\eta}_3 = (1+a_2)\dot{\eta}_1 : f(\dot{\eta}_3) > f(\dot{\eta}_2). \tag{4.13}$$

Now define a quadratic function in η_j along the direction j. Consider that ϕ_0, ϕ_1, ϕ_3 are the parameters of the function expressed as:

$$f(\eta) = \phi_0 + \phi_1(\eta) + \phi_2(\eta^2). \tag{4.14}$$

Let us assume that $f(\eta)$ takes the function values $f(\dot{\eta}_1), f(\dot{\eta}_2), f(\dot{\eta}_3)$ at three different positions $\eta = \dot{\eta}_1, \dot{\eta}_2, \dot{\eta}_3$. Therefore, we can write the following relations:

$$f(\dot{\eta}_1) = \phi_0 + \phi_1(\dot{\eta}_1) + \phi_2(\dot{\eta}_1^2) \tag{4.15}$$

$$f(\dot{\eta}_2) = \phi_0 + \phi_1(\dot{\eta}_2) + \phi_2(\dot{\eta}_2^2) \tag{4.16}$$

$$f(\dot{\eta}_3) = \phi_0 + \phi_1(\dot{\eta}_3) + \phi_2(\dot{\eta}_3^2). \tag{4.17}$$

Using the above relations following results are derived:

$$\phi_0 = \frac{[\dot{\eta}_2\dot{\eta}_3(\dot{\eta}_3 - \dot{\eta}_2)f(\dot{\eta}_1) + \dot{\eta}_3\dot{\eta}_1(\dot{\eta}_1 - \dot{\eta}_3)f(\dot{\eta}_2) + \dot{\eta}_1\dot{\eta}_2(\dot{\eta}_2 - \dot{\eta}_1)f(\dot{\eta}_3)]}{(\dot{\eta}_1 - \dot{\eta}_3)(\dot{\eta}_2 - \dot{\eta}_3)(\dot{\eta}_3 - \dot{\eta}_1)} \tag{4.18}$$

$$\phi_1 = \frac{-[(\dot{\eta}_3^2 - \dot{\eta}_2^2)f(\dot{\eta}_1) + (\dot{\eta}_1^2 - \dot{\eta}_3^2)f(\dot{\eta}_2) + (\dot{\eta}_2^2 - \dot{\eta}_1^2)f(\dot{\eta}_3)]}{(\dot{\eta}_1 - \dot{\eta}_3)(\dot{\eta}_2 - \dot{\eta}_3)(\dot{\eta}_3 - \dot{\eta}_1)} \tag{4.19}$$

$$\phi_2 = \frac{[(\dot{\eta}_3 - \dot{\eta}_2)f(\dot{\eta}_1) + (\dot{\eta}_1 - \dot{\eta}_3)f(\dot{\eta}_2) + (\dot{\eta}_2 - \dot{\eta}_1)f(\dot{\eta}_3)]}{(\dot{\eta}_1 - \dot{\eta}_3)(\dot{\eta}_2 - \dot{\eta}_3)(\dot{\eta}_3 - \dot{\eta}_1)}. \tag{4.20}$$

Suppose that η^* is the minimum point of the equation 4.14. Then the condition that η^* is minimum point requires $\phi_2 > 0$. Therefore it follows that:

$$\eta^* = -\frac{\phi_1}{2\phi_2}. \tag{4.21}$$

Consequently:

$$\eta^* = \frac{1}{2}\frac{[(\dot{\eta}_3^2 - \dot{\eta}_2^2)f(\dot{\eta}_1) + (\dot{\eta}_1^2 - \dot{\eta}_3^2)f(\dot{\eta}_2) + (\dot{\eta}_2^2 - \dot{\eta}_1^2)f(\dot{\eta}_3)]}{(\dot{\eta}_2 - \dot{\eta}_3)f(\dot{\eta}_1) + (\dot{\eta}_3 - \dot{\eta}_1)f(\dot{\eta}_2) + (\dot{\eta}_1 - \dot{\eta}_2)f(\dot{\eta}_3)}. \tag{4.22}$$

The corresponding condition that η^* is minimum, is therefore:

$$\frac{[(\dot{\eta}_3 - \dot{\eta}_2)f(\dot{\eta}_1) + (\dot{\eta}_1 - \dot{\eta}_3)f(\dot{\eta}_2) + (\dot{\eta}_2 - \dot{\eta}_1)f(\dot{\eta}_3)]}{(\dot{\eta}_1 - \dot{\eta}_3)(\dot{\eta}_2 - \dot{\eta}_3)(\dot{\eta}_3 - \dot{\eta}_1)} > 0. \tag{4.23}$$

Further define a quantity a_0 in relation to the minimum point η^* and $\dot{\eta}_1$ such that the following relation exists:

$$\eta^* = (1 + a_0)\dot{\eta}_1. \tag{4.24}$$

The value of a_0 is determined from the above relations. Simplification and some rearrangement results the following expression:

$$a_0 = \frac{1}{2}\frac{(a_1^2 - a_2^2)f(\dot{\eta}_1) + a_2^2 f(\dot{\eta}_2) - a_1^2 f(\dot{\eta}_3)}{(a_1 - a_2)f(\dot{\eta}_1) + a_2 f(\dot{\eta}_2) - a_1 f(\dot{\eta}_3)}. \tag{4.25}$$

The optimized value of η^* is therefore given by:

$$\eta^* = \left(1 + \frac{1}{2} \frac{(a_1^2 - a_2^2) f(\eta_1) + a_2^2 f(\eta_2) - a_1^2 f(\eta_3)}{(a_1 - a_2) f(\eta_1) + a_2 f(\eta_2) - a_1 f(\eta_3)} \right) \eta_1 \quad \Diamond .$$

4.3.2 Difficulty in Interpolation Search and its Correction

The method described in Section 4.3.1 repeatedly monitors the error surface to determine the variable learning rate. Since it takes into account the shape of the contour surface, an appropriate learning rate is therefore possible to obtain. One possible difficulty inherent with this method is that during training the error function will not always behave as a quadratic function and the interpolation method may not produce correct results. Further away from the minimum point the value of $\eta^* = (1 + a_0)\eta_1$ may actually be $f(\eta^*) > f(\eta_1)$. If this happens, it is corrected by re-evaluating the minimum point as $\eta^* = (1 + a_1)\eta_1$ so that $f(\eta_2) < f(\eta_1)$. A proper values of a_0, a_1, a_2 would uniquely bound the value of η^* in relation to η_1 and η_3. The quantities a_0, a_1, a_2 are expressed as relative percentage factors with reference to a trial point.

4.3.3 Algorithm to Determine Self-Adaptive Learning Rate Parameter

The algorithm that determines the self-adaptive learning rate parameter by interpolation search is described next.

	Initialization:
Step: 1a	*Set* $j \leftarrow 0$, *and* $\varepsilon \leftarrow 0^+$ *as termination criteria and* $\wp \leftarrow 0^+$ *as interpolation search precession factor. Set* $\delta_3 \leftarrow .01$, $i \leftarrow 0$, *let* m = *number of ANN connection weights, set learning rate parameter in direction j, is defined by* $\eta_j \equiv (\eta_1, \eta_2, \ldots, \eta_m) \leftarrow 0$. *Set* $\delta_1 \leftarrow 4$, $\delta_2 \leftarrow 2.5$, *Initialize* $w \equiv w_j \equiv (w_1, w_2, \ldots, w_m)$. *set* $d \equiv d_j \equiv (d_1, d_2, \ldots, d_m) \leftarrow 0$.
Step:1b	*1.1 Set,* $j \leftarrow j + 1$
Step:1c	*1.2* $\eta_1 \leftarrow w_j$, $d_j \leftarrow 1$, $f_1 \leftarrow f(w_j + \eta_1 d_j)$ *and perform next step*
Step: 2	*2.1* $a_j \leftarrow \delta_3$
Step:3	*3.1* $\eta_2 \leftarrow (1 + a_j)\eta_j$
Step:4	*4.1* $f_2 \leftarrow f(w_j + \eta_2 d_j)$, *if* $f_2 < f_1$ *perform step 5, else*
	4.2 $a_j \leftarrow \delta_1 (a_j)$, *if* $\lvert a_j \rvert < \wp$, *perform 8, otherwise perform step 3*

Step:5	*5.1* $a_2 \leftarrow a_1$
Step:6	*6.1* $a_2 \leftarrow (\delta_2) a_2$
	6.2 $\dot{\eta}_3 \leftarrow (1+a_2)\dot{\eta}_1$, $f_3 \leftarrow f(w+\dot{\eta}_3 d_j)$, *if* $f_3 > f_2$ *perform step 7, otherwise*
	6.3 $a_1 \leftarrow a_2$, $f_2 \leftarrow f_3$, *repeat step 6*
Step:7	*Find* a_0
	7.1 $a_0 \leftarrow \frac{1}{2}\frac{(a_1^2 - a_2^2)f(\dot{\eta}_1) + a_2^2 f(\dot{\eta}_2) - a_1^2 f(\dot{\eta}_3)}{(a_1 - a_2)f(\dot{\eta}_1) + a_2 f(\dot{\eta}_2) - a_1 f(\dot{\eta}_3)}$
	7.2 $\eta^* \leftarrow (1+a_0)\dot{\eta}_1$, $f_1 \leftarrow f(w_j + \eta_j^* d_j)$, *if* $f_1 < f_2$ *perform step 2, otherwise set* $\eta^* \leftarrow (1+a_1)\dot{\eta}_1$ *and* $f_1 \leftarrow f(w_j + \eta_j^* d_j)$, *so that* $f_2 < f_1$ *and perform step 2.*
	7.3 If $\lvert f_1 - f_2 \rvert \leq \varepsilon$, *perform step 8, otherwise*
	7.4 Set $\hat{f} \leftarrow f(w_j + \eta^* d_j)$; *and monitor error function value*
	7.5 change $\dot{\eta}_1 \leftarrow \frac{(1+a_1)\dot{\eta}_1}{(1+a_0)}$ *and perform step 1c.*
Step:8	*8.1 set* $d_j \equiv (d_1, d_2, \ldots\ldots, d_m) \leftarrow 0$, *If* $j = m$ *perform step 9, else, perform step 1b.*
Step 9.	*9.1 Set* $k \leftarrow k+1$, *for next iteration.* $(d_j \equiv d_1, d_2, \ldots\ldots, d_m) \leftarrow 0$, *set* $j \leftarrow 0$, *perform step 1b.*

Table 4.1 The interpolation search algorithm

4.3.3.1 Interpolation search Algorithm implementation

To begin the interpolation search, first set the equation $\dot{\eta}_2 = (1+a_1)\dot{\eta}_1$ and sample the function value $f(\dot{\eta}_2)$ by a factor $a_1 = \delta_3 = 0.01$ relative to the initial position $\dot{\eta}_1$. If the sampled function value is greater than the initial position, the factor a_1 is adjusted depending on the condition of the error surface by the factor $\pm\delta_1$, which is set as $a_1 = a_1 * \delta_1$, where $\delta_1 = 4$. If the new sampled function value is less than the initial position, the factor a_2 is magnified by $\delta_2 = 2.5$. A new location $\dot{\eta}_3$ is now sampled with reference to the initial position $\dot{\eta}_1$ defined by $\dot{\eta}_3 = (1+a_2)\dot{\eta}_1$ to force the condition $f(\dot{\eta}_3) > f(\dot{\eta}_2)$. Otherwise, reset the value of $a_2 \leftarrow a_1$ and reset the function value

$f(\dot{\eta}_3) \leftarrow f(\dot{\eta}_2)$ and again attempt to evaluate the function in other different location such that the desired condition is found. When the three sampled error surface locations are suitable according to the proposition 4, the best learning rate is determined according to the expression given in proposition 4. Due to the complexity of the error function if we get the condition $f(\dot{\eta}_2) > f(\dot{\eta}_1)$, it is corrected by setting $\eta^* = (1+\alpha_0)\dot{\eta}_1$, provided it satisfies all other conditions.

4.4 Development of Descent Directions in ANN

The objective of this section is to construct normalized descent direction for ANN training. It will be shown that the normalized direction provides descent to the error function. Let d be a m dimensional vector defined as $d = (d_1, d_2, \ldots\ldots, d_m)$ to a function $f(w)$ at w, then there exist a scalar $\delta > 0$ such that $f(w+\eta d) < f(w)$ for all $\eta \in (0, \delta)$. If $[\{f(w+\eta d) - f(w)\}/\eta] < 0$, then d is a decent direction under the limiting condition $\eta \to 0^+$. If the function is differentiable at w with non-zero gradient, then $\frac{-\nabla f(w)}{|\nabla f(w)|}$ is the normalized descent direction. The following proposition supports this.

Proposition 4.5

For a differentiable error function $f(w): E^m \to E^1$ at w; there exist a non-zero gradient $\nabla f(w)$ such that the steepest descent direction $d^* = d_j^* = \frac{-\nabla f(w)}{|\nabla f(w)|}$ is the minimization direction to the problem of the form:

$$\min f(w_j + \eta_j d_j) \tag{4.26}$$

subject to $|d_j| \le 1$, $\eta_j \in (0, \delta)$, $\delta > 0$.

Proof

For a differentiable and continuous descent error function $f(w)$, the gradient at w with a learning rate $\eta \in (0, \delta)$ in direction d (according to proposition 4.1) is expressed as:

$$\lim_{\eta \to 0^+} \frac{f(w+\eta d) - f(w)}{\eta} = \nabla f(w)^T d . \tag{4.27}$$

To prove that d is a descent direction, the condition $\nabla f(w)^T d < 0$ must hold (proposition 4.2).

Using Schwartz inequality with $\|d\| \le 1$, we obtain the following relation:

$$\nabla f(w)^T d \geq -\|\nabla f(w)\|\|d\| \geq -\|\nabla f(w)\|. \tag{4.28}$$

When $d^* = \frac{-\nabla f(w)}{\|\nabla f(w)\|}$; the above inequality is satisfied and the given function is minimized using $\nabla f(w)^T d$ subject to $\|d\| \leq 1$. Therefore the directional vector $d^* = \frac{-\nabla f(w)}{\|\nabla f(w)\|}$ that reduces the function value is the steepest descent direction ◊.

4.4.1 Approximation of Normalized Descent Direction

The normalized gradient of the error function is computed using the central difference approximation scheme. Once the gradient is found, the normalization is a routine procedure. The step size of the difference scheme is controlled by the convergence rate of the function value. The central difference gradient approximation is discussed next.

The property of the error function $f(w)$ depends on the ANN weights in different layers explicitly. The error function $f(w)$ is said to be differentiable at $w \in E^m$ if $\frac{\partial f(w)}{\partial w_j}$ exists for all $j = 1,2,...m$. The gradient of $f(w)$ at a trial point w is defined as $\nabla f(w) = \left[\frac{\partial f(w)}{\partial w_1}, \frac{\partial f(w)}{\partial w_2}, ..., \frac{\partial f(w)}{\partial w_m}\right]^T$. To compute the gradient of the error function $f(w)$ each parameter is varied independently in the neighborhood of the trial point to yield an approximate value of the partial derivatives. The error function is perturbed by an amount π_j and its consequences are determined. Define a quantity ψ_j is relation with the convergence rate ρ and the step size, π_j, according to the following expression:

$$\pi_j = \rho\psi_j \tag{4.29}$$

where,

$$\rho = \frac{f(w_{k+1})}{f(w_k)} \tag{4.30}$$

is the rate of convergence of the error function and ψ_j is an arbitrary vector containing m values to be set at the beginning of the algorithm's execution. The values of this vector are the percentage factors that are related to the weight vector w_j. Hence the value of ψ_j is related with the ANN weights by a factor and can be defined as:

$$\psi_j = (factor_j) * w_j . \tag{4.31}$$

The quantity $(factor_j)$ has a magnitude of the order 10^{-2} to 10^{-4} and depends on the magnitude of the weights. The gradient approximation scheme is explained below.

Proposition 4.6

If $f : w \in E^m$ is continuously differentiable, then for any non-zero perturbation π_j, during training iteration k the directional derivative of $f(w_j)$ at w_j in direction d_j is defined by

$$d_j \equiv -\nabla f(w_j) \equiv -\frac{f(w_j + \pi_j \bar{I}_j) - f(w_j - \pi_j \bar{I}_j)}{2\pi_j}. \tag{4.32}$$

The term π_j is the controlled central difference step size and $\bar{I}_j \equiv (\bar{I}_1, \bar{I}_2, \ldots, \bar{I}_m)^T$ is the column vector of 1 in the $\bar{I}^{th}$ element and zero elsewhere.

Proof

Consider the Taylor series expansion of an ANN error function $f(w_j + \pi_j \bar{I}_j)$ and $f(w_j - \pi_j \bar{I}_j)$ around w_j with small perturbation π_j as shown in 4.33 and 4.34:

$$f(w_j + \pi_j \bar{I}_j) = f(w_j) + \nabla f(w_j) \pi_j \bar{I}_j + \nabla^2 f(w_j) \frac{\pi_j^2 \bar{I}_j}{2!}; \tag{4.33}$$

$$f(w_j - \pi_j \bar{I}_j) = f(w_j) - \nabla f(w_j) \pi_j \bar{I}_j + \nabla^2 f(w_j) \frac{\pi_j^2 \bar{I}_j}{2!}. \tag{4.34}$$

Subtracting 4.33 from 4.34 we obtain the following result:

$$\nabla f(w_j) = \frac{f(w_j + \pi_j \bar{I}_j) - f(w_j - \pi_j \bar{I}_j)}{2\pi_j \bar{I}_j} \tag{4.35}$$

and this proves the proposition $\Diamond$.

4.4.2 Issues in Central Difference Approximations

The appropriate value of ψ_j plays a significant role in approximating the gradient information. Large value of ψ_j would cause instability in the approximation process and very small value would slow the convergence. Hence an appropriate selection of ψ_j

would be necessary. This may vary from one problem to another. A simulation experiment determines the appropriate values of ψ_j. The finite difference step size is also controlled by the convergence rate of the error function. Depending on the shape of the contour surface the convergence ratio changes and therefore the central difference step size is dynamically adjusted. A simple approach is to use the value recommended in Section 4.4.1.

4.5 The Self-Adaptive Back Propagation Algorithm

The error function describes a hyper surface in m dimensional space. The training aims at searching the space to locate a minimum weight vector. Usually, an error function contains more that one local minimum (Bishop, 1995). If the search space is restricted to a region in which it is known that there is a local minimum then a deterministic or stochastic search method (Ma et al., 2000) can be employed to locate the minimum. The proposed gradient descent training generates search directions from the proposition 4.6. The value of the central difference step size $\pi_j = \rho\psi_j$ is chosen such that the minimum is reached in few steps. As the search converges to minimum, the magnitude of π_j is modified to adjust the direction. All the weights w_j are adjusted according to the magnitude of the learning rate determined by constrained search so that the resultant direction of travel in parameter space is along the descent direction of the error surface. The results of the previous section provide us the direction vector whose components are the rates at which the error surface decreases most rapidly.

To determine the gradient, the variation of $f(w)$ in the neighborhood of a trial weight vector is sampled independently for each parameter. To find an approximate value of the partial derivatives, the amount by which w_j is changed in order to determine the derivative is kept smaller than the step size π_j. During the progress of training the value of ρ is set to $\frac{f(w_{k+1})}{f(w_k)}$. The gradient components are sampled according to the convergence rate of the error function. This provides a mechanism to approximate the gradient according to the local condition of the error surface. The gradient has both magnitude and direction. If the dimensions of the parameters w_j are different, the components of the partial derivatives are also different in dimensions. The direction, which the proposed gradient descent back propagation follows, is the normalized gradient of the steepest descent according to the proposition 4.5 and 4.6.

The search begins by incrementing all the parameters by an amount π_j in direction $d_j = \frac{-\nabla f(w_j)}{\|\nabla f(w_j)\|}$. If the sampled gradients are fairly approximate, the method is reliable. The advantage of this method comes mainly from the proposition 4.6 and 4.4. The proposition

4.4 and 4.6 choose its own learning rate and descent direction. Thus the user is spared form the problem of trying to optimize the learning rate for convergence and precision. The interpolation search over the error surface improves the precision of locating the minimum. The self-adaptive gradient descent BP algorithm is described in Table 4.2.

Initialization:

a.) Set a termination criteria $\varepsilon \leftarrow 0^+$, *iteration counter* $k \leftarrow 1$, $\delta \leftarrow 0^+$ *(as a percentage factor),* $j \leftarrow 1$ *, set limit to the iteration number as 50,000 and let* $w_{k=1}$ *be the initial vector, execute step 1.*

Step 1 *Determine direction of search by Central difference*

a.) Set, $\pi_j = \rho\psi_j$, $\rho = \frac{f(w_{k+1})}{f(w_k)}$ *and* $\psi_j = (factor_j) * w_j$

b.) $\nabla f(w_j) \leftarrow \dfrac{f(w_j + \pi_j \bar{I}_j) - f(w_j - \pi_j \bar{I}_j)}{2\pi_j \bar{I}_j}$

c.) If $j = n$, *perform step 2 else* $j \leftarrow j+1$ *and repeat step 1*

Step 2: *a.) Set,* $j \leftarrow 1$

b.) $\nabla f(w_j) \leftarrow \dfrac{\nabla f(w_j)}{\left\|\nabla f(w)\right\|}$

c.) $d_j \leftarrow -\nabla f(w_j)$

d.) If $\left\|\nabla f(w_k)\right\| < \varepsilon$, *stop and report* w_k *as the solution, otherwise perform interpolation search.*

Interpolation Search: Perform interpolations search according to Table 4.1 and select adaptation length η_k *at iteration* k. *Solve the following minimization problem:*

a.) $\eta \equiv \min_{\eta \geq 0} f(w_k + \eta_k d_k)$

b.) Set $w_{k+1} \leftarrow w_k + \eta_k d_k$

Step 3: *a.)* $j \leftarrow 1$, $k = k+1$ *and perform step 1.*

Table 4.2 Self-Adaptive Back Propagation Training Algorithm

4.6 Convergence of the Algorithm

We need to consider a composite mapping for further analysis in back propagation convergence. It is demonstrated in this section that the algorithm terminates at zero gradients and hence the algorithm converges. We note from Chapter 2 that the error function of interest is a complex geometry. The following section shows that the proposed self-adaptive algorithm converges to a local minimum point while satisfying the first order condition. This is done through a composite algorithmic mapping. It exploits the property of a closed mapping in order to generate a convergent sequence $\{f(w_k)\}$. The convergence characteristics of the proposed training method are discussed next.

Given a current weight w_k at iteration k the next weight at iteration $k+1$ is determined by the algorithm as w_{k+1}, the subsequent iterations generate the sequence w_{k+2}, w_{k+3}... and so on. Moving in direction $d_k = -\nabla f(w_k)$ and determining a learning rate η_k the interpolation search produce this sequence. The error function $f(w)$, therefore, generates a sequence according to the following iterate:

$$w_{k+1} = w_k + \eta_k d_k = w_k - \eta_k \nabla f(w_k); (k = 1,2,.....,\infty). \tag{4.36}$$

As usual d_k is the negative gradient vector at the point w_k. The sequence generated by the equation 4.36 converges to a local minimum point in the proposed training. The following proposition provides sufficient condition for convergence.

Proposition 4.7

The proposed self-adaptive back propagation training algorithm generates convergent sequence and terminates at a point with zero gradients.

Proof

Let Ω be a solution set such that $\Omega = \{w^* : \nabla f(w^*) = 0\}$. Define the algorithmic map $M = LA$, where $A[w, \nabla f(w)]$ is the map that determines the descent direction and L is the interpolation search map. Assume that the derivative of $f(w)$ is available, then A is continuous. Furthermore, L is closed by the property of interpolation search due to proposition 4.3. Since A is continuous and L is closed, the map $A[w, \nabla f(w)]$ is closed at w. The overall map M is therefore, closed at w. Using these properties and with the help of proposition 4.3 and 4.5 we have $\lim_{\eta \to 0^+} \frac{f(w_k + \eta_k d_k) - f(w_k)}{\eta_k} = \nabla f(w)^T d < 0$. Hence we have a

vector which points towards the minimum trajectory. Since d is a descent direction, we have $f(w_k+\eta_k d_k)<f(w_k)$ for $\eta_k\in(0,\delta)$ and $\delta>0$. Since the map $M(w_k)$ is closed, the set value mapping is finite. Hence, we get the limit points $w_k\to w^*$ and $(w_k+\eta_k d_k)\in M(w_k)$. The map M therefore generates a convergent sequence defined as $\{w_k+\eta_k d_k\}$ or simply $\{w_k\}$. Therefore, as $f(w_k+\eta_k d_k)\to f(w^*)$ for some value of $k\to\infty$, and the convergent sequence $\{w_k\}$ generated by the self-adaptive BP algorithm converges to a limit point w^* with zero gradients 0.

4.6.1 Convergence to Local Minimum

The result in previous section shows that the proposed self-adaptive back propagation algorithm converges to zero gradients. It implies that the error function $f(w)$ is minimized sequentially. Consequently, the proposed algorithm terminates at a local minimum. The following proposition is put forward to demonstrate this.

Proposition 4.8

If the limit of sequence w_k converges to w^* and if the first order partial derivatives of $f(w)$ exists with respect to all network weight parameters in the neighborhood of $w^*\in w$ then $f(w)$ has a local minimum at $w=w^*$.

Proof

By contradiction, assume that $f(w^*)$ is not a local minimum of $f(w)$, then $[\nabla f(w^*)]^2>0$. Since the first derivative $\nabla f(w)$ of the error function is available, there exist the numbers $\mu>0$ and $\eta>0$ such that:

i). $[\nabla f(w)]^T\nabla f(u)\geq\mu>0$ for all w and u in some neighborhood of w^*;

ii). If w_k is a point in this neighborhood, then w_{k+1} is also a neighborhood point where $w_{k+1}=w_k-\eta_k\nabla f(w_k)$.

Suppose that the error function is a descent function and the sequence w_k to w_{k+1} generated by the algorithm changes the function value. Then by the mean value theorem, the following condition holds:

$$f(w_{k+1})=f(w_k)-\eta_k[\nabla f(w_k)]^T[\nabla f[w_k-\theta\eta_k\nabla f(w_k)]],0\leq\theta\leq1. \qquad (4.37)$$

In equation 4.37, the coefficient $-\eta_k$ is at least θ from (i) and (ii). Thus, the sequence generated by the algorithm w_k to w_{k+1}, decreases the function value $f(w)$ by at least $\eta_k \mu$. There are infinite sequences of $\{w_k\}$ in any neighborhood of w^*. Since $\lim_{k \to \infty} w_k = w^*$, hence, by applying the algorithm for successive values of k, the error function value $f(w^*) \to -\infty$, which contradicts the fact $\nabla f(w^*)$ exists. The assumption that $f(w^*)$ is not local minimum of $f(w)$ is therefore false, and the proposition is proved $\Diamond$.

4.6.2 The Rate of Convergence of the Algorithm

For any $w \in E^m$ the self-adaptive BP in quadratic case converges to the unique minimum point w^* of $f(w)$. Since the proposed self-adaptive BP is similar to the first order gradient descent algorithm, the following inequality in 4.38 exists (Luenberger, 1984):

$$\frac{\|f(w_{k+1})\|}{\|f(w_k)\|} \leq \left(\frac{A-a}{A+a}\right)^2 \tag{4.38}$$

where, A and a are the largest and lowest eigenvalues of the Hessian matrix H which is assumed to be positive definite. The convergence of the self-adaptive BP algorithm will slow if the contours are more eccentric. If $A = a$, the contours are circular and the convergence is achieved with less efforts. Convergence will slow, if the eigenvalues are at a greater distance (Luenberger, 1984). The above relation suggests that the self-adaptive BP converge linearly with a convergence ratio no greater than $[(A-a)/(A+a)]^2$. The ratio A/a of the largest and lowest eigenvalue determines the condition number, which influence the convergence rate. The convergence ratio can be represented by $\left(\frac{A-a}{A+a}\right)^2 = \left(\frac{(A/a)-1}{(A/a)+1}\right)^2$. It is this factor by which the error function is reduced per iterations. The ratio A/a governs the convergence in the self-adaptive BP algorithm. The convergence becomes slow if the condition number is high in magnitude. It is investigated in Chapter 7 with a seasonal time series problem.

4.7 Performance of the Self-Adaptive BP Algorithm

The computational experience with the proposed algorithm and its convergence behavior is discussed next. The XOR benchmark problem is chosen to demonstrate the various aspect of the convergence of the algorithm. A trained 2-2-1 XOR is displayed in Figure 4.1. The zero input to the hidden layer is approximated as 10^{-5}. Sample calculations are shown in Table 4.3.

4.7.1 Learning Rate for Standard Back Propagation Method

To select a suitable learning rate, an experiment with XOR problem is carried out. The initial starting weight is the same but the learning rate is changed from one experiment to another. The simulated results are given in Table 4.4 with learning rate and its effect on function value, number of epoch and number of function evaluations. This experiment suggests a reasonable value of the learning rate is 0.1, which is a bargain between epoch number, function evaluations and terminal function value. The value marked with (*) sign indicates that the training is not converging with the predefined convergence criteria as discussed in Section 3.5.2 in Chapter 3.

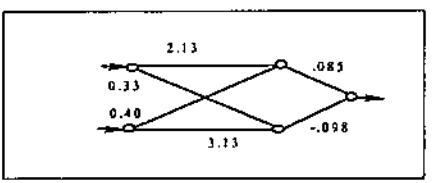


Figure 4.1. A trained 2-2-1 ANN XOR

Trained XOR

$x_1^{p=1}=0$, $x_2^{p=1}=0$, $h_1^{p=1}=(2.13)*0+(.40)*0=0\Rightarrow 10^{-5}$, $h_2^{p=1}=(.33)*0+(3.13)*0=0\Rightarrow 10^{-5}$

$g_1^{p=1}=\frac{1}{1+\ln|10^{-5}|}=-0.078$, $g_2^{p=1}=\frac{1}{1+\ln|10^{-5}|}=-0.078$,

$z_1^{p=1}=(.085)*(-0.078)+(-0.098)*(-0.078)\cong 0.001$

$x_1^{p=2}=1$, $x_2^{p=2}=0$, $h_1^{p=2}=(2.13)*1+(.40)*0=2.13$, $h_2^{p=2}=(.33)*1+(3.13)*0=.33$

$g_1^{p=2}=\frac{1}{1+\ln|2.13|}=0.569$, $g_2^{p=2}=\frac{1}{1+\ln|.33|}=-9.2027$,

$z_1^{p=2}=(.085)*(.569)+(-.098)*(-9.2027)\cong .95$

$x_1^{p=3}=0$, $x_2^{p=3}=1$, $h_1^{p=3}=(2.13)*0+(.40)*1=.40$, $h_2^{p=3}=(.33)*0+(3.13)*1=3.13$

$g_1^{p=3}=\frac{1}{1+\ln|.40|}=11.95$, $g_2^{p=3}=\frac{1}{1+\ln|3.13|}=0.467$, $z_1^{p=3}=(.085)*(11.95)+(-.098)*(.467)\cong .97$

$x_1^{p=4}=1$, $x_2^{p=4}=1$, $h_1^{p=4}=(2.13)*1+(.40)*1=2.53$, $h_2^{p=4}=(.33)*1+(3.13)*1=3.46$

$g_1^{p=4}=\frac{1}{1+\ln|2.53|}=.5186$, $g_2^{p=4}=\frac{1}{1+\ln|3.46|}=.4461$, $z_1^{p=4}=(.085)*(.5186)+(-.098)*(.4461)\cong 0.00036$

Table 4.3 Sample calculations with XOR (2-2-1 ANN) problem

	Experiment Number											
	1	2	3	4	5	6	7	8	9	10	11	12
L. Rate	0.01	0.05	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
Function Value	0.0001	0.0005	0.0006	0.05000	0.01600	0.09100	0.06000	0.18000	0.34000	0.29000	0.3400	0.4300
Iterations	50000*	5864	706	9	388	5	181	6	73	4	21	25
Function Evaluation	50000*	6866*	4950	71	2724	43	1275	50	519	36	155	183

Table 4.4 Learning rate and training performance in standard back propagation method [(*) indicates training is not converging]

Epoch	Function Value	Self-Adaptive Parameters						
k	$f(w_k)$	η_k	$d_{k,1}$	$d_{k,2}$	$d_{k,3}$	$d_{k,4}$	$d_{k,5}$	$d_{k,6}$
1	1.02E-01	0.960231	0.00762	-0.01923	-0.03524	0.22718	-0.33601	-0.91514
2	9.53E-02	0.361391	-0.07835	0.00459	-0.51824	-0.84664	-0.03533	-0.08495
3	9.22E-02	0.022106	-0.01671	0.00155	0.74838	0.66273	-0.00828	-0.01915
4	8.99E-02	0.000457	-0.00441	0.00052	-0.65271	-0.75757	-0.00232	-0.00523
5	8.87E-02	0.000037	-0.00068	0.00009	0.92184	0.38756	-0.00037	-0.00083
6	8.71E-02	0.000002	-0.00071	0.00011	-0.80886	0.588	-0.0004	-0.00087
7	8.58E-02	0	-0.00032	0.00005	-0.70311	0.71108	-0.00018	-0.0004

Table 4.5 Convergence with 2-2-1 ANN XOR problem with starting vector $(-.1,.2,.2,-.3,.1,.2)^T$ using self-adaptive BP training

4.7.2 Analysis with XOR Problem

The self-adaptive and parameter free training algorithm is tested with 2-2-1 ANN configuration XOR problem. Figure 4.2 displays the self-adaptive parameters generated by the algorithm, while Figure 4.3 shows the convergence of the algorithm with reference to the number of epoch/iterations. The input data for the XOR problem corresponds to the subset of the Table A.1 in Appendix A in rows 1, 2, 3, 4 and column 4, 5. The random starting weights $w_j = w_1, w_2, \ldots, w_6$ are taken from the Table A.2 in Appendix A. The learning rate is 0.1 for the standard back propagation training.

The self-adaptive parameters and descent directions are computed according to the method discussed in Section 4.3 and 4.4 with starting vector $(-.1,.2,.2,-.3,.1,.2)^T$. Sample calculations are shown in Table 4.5. Figure 4.2 shows the self-adaptive learning parameter chosen by the algorithm. It reduces the function value monotonically as shown in Figure 4.3.

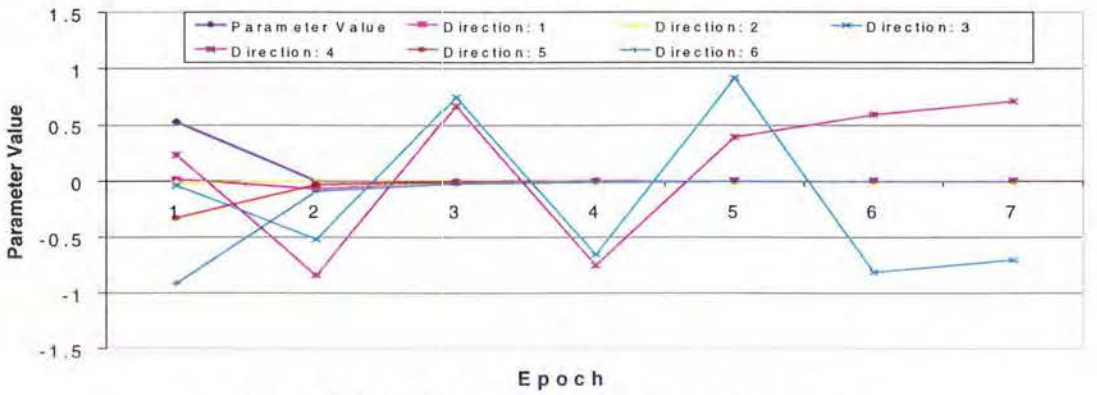


Figure 4.2 Self-Adaptive Parameter Generation

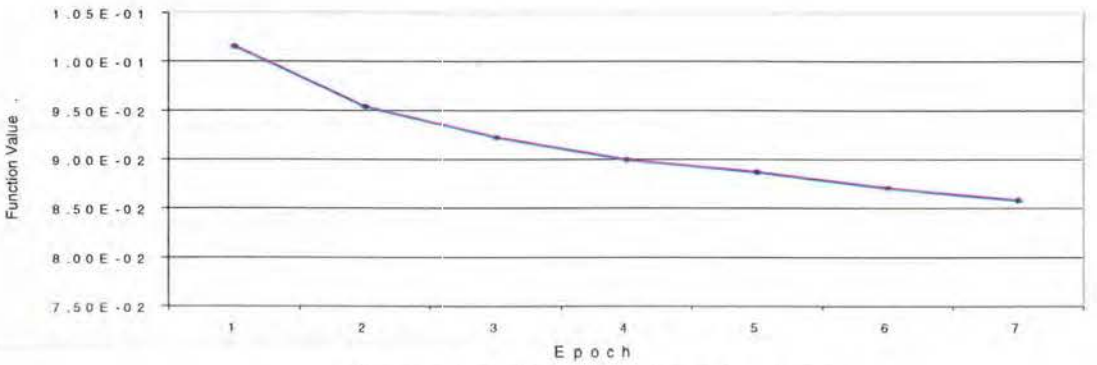


Figure 4.3 Function Convergence

4.7.3 Random Starting Weights in the Wide Range

Table 4.6 shows the results with random starting points that are according to the Table A.2 in Appendix A. The average epoch, function evaluations and gradient evaluations are 151.8, 916.8 and 2601.1 respectively. The corresponding values with the standard back propagation method are 5142.3, 5144.3 and 30861.8 respectively. The median performance of the proposed algorithm with epoch size, function evaluations and gradient evaluations corresponds to the values 17, 108 and 206, while with the standard back propagation method these counts are 3579, 3581 and 21480 respectively. The standard deviations are higher in magnitude and suggest that some experiments show worst performance. The experiment number 2, for example, with the proposed method shows bad performance. It is due to the fact that the starting points are not in favorable location in the error surface or the algorithm face geometry where the convergence is difficult.

The maximum and minimum epoch sizes are 1349 and 6 respectively with the proposed method. The maximum and minimum numbers of total function evaluations including gradient evaluations are 31896 and 126 respectively. The related value of the range

statistics is also wide and hence it suggests that there are considerable variations in the experimental results.

The simulations with the standard BP method show that the maximum and minimum numbers of epoch sizes are 17434 and 15 respectively. The corresponding figures for the total number of function evaluations are 122046 and 133 respectively.

	Proposed Method					Standard Back Propagation Method				
Experiment #	Epoch	Function evaluations	Gradient evaluations	Total Function evaluations	Terminal function value	Epoch	Function evaluations	Gradient evaluations	Total Function evaluations	Terminal function value
1	16	102	183	285	1.47E-9	22	24	138	162	6.2E-04
2	1349	8100	23796	31896	2E-8	5867	5869	35208	41077	8.2E-04
3	14	90	194	289	1.5E-8	29	31	180	211	1.5E-02
4	10	66	147	213	1.3E-10	17434	17436	104610	122046	5.1E-03
5	18	114	218	332	1.6E-10	4474	4476	26850	31326	2.5E-03
6	47	288	713	1001	8.1E-5	5737	5739	34428	40167	4.6E-03
7	29	180	314	494	1.3E-8	2137	2139	12828	14967	3.0E-03
8	22	138	272	410	2.4E-7	13024	13026	78150	91176	8.9E-04
9	7	48	78	126	8.58E-2	15	17	116	133	2.9E-03
10	6	42	96	138	4E-8	2684	2686	16110	18796	1.0E-02
Mean	151.8	916.8	2601.1	3518.4	0.00859	5142.3	5144.3	30861.8	36006.1	0.00454
Median	17	108	206	310.5	1.8E-08	3579	3581	21480	25061	0.00295
Standard Deviation	420.828	2524.97	7449.30	9974.04	0.02713	5852.45	5852.45	35112.7	40965.2	0.00461
Range	1343	8058	23718	31770	0.0858	17419	17419	104494	121913	0.01438
Minimum	6	42	78	126	1.3E-10	15	17	116	133	0.00062
Maximum	1349	8100	23796	31896	0.0858	17434	17436	104610	122046	0.015

Table 4.6 Comparison with standard back propagation method (2-2-1:ANN XOR problem: starting point in wide range)

4.7.4 Random Starting Weights in the Small Range

It is relevant to point out that the random starting points in a small range of magnitude influence the average performance of the algorithm. Ten simulations are carried out to test the proposed training method and the standard BP algorithm with small random starting weights. The corresponding simulation results are shown in Table 4.7.

The average epoch, function evaluations and gradient evaluations are 10.8, 124.5 and 70.8 respectively. The corresponding values with the standard back propagation training are 463.6, 465.6 and 2787.6 respectively. The median performance of the proposed algorithm with epoch size, function evaluations and gradient evaluations corresponds to the values 9.5, 108.5 and 63, while with the standard back propagation training these counts are 25.5,

27.5 and 159 respectively. The related standard deviations are low with the proposed method and suggest consistent performance. The standard back propagation show high magnitude in standard deviations and therefore inconsistent behavior is expected.

	Proposed Method					Standard Back Propagation Method				
Experiment #	Epoch	Gradient evaluations	Function evaluations	Total #. of function evaluations	Terminal function value	Epoch	Gradient evaluations	Function evaluations	Total #. of function evaluations	Terminal function value
1	17	108	163	271	3.00E-07	25	156	27	183	6.18E-04
2	19	120	222	342	9.70E-06	27	168	29	197	8.21E-03
3	20	126	221	337	1.40E-06	26	162	28	190	1.50E-02
4	3	24	54	78	9.90E-06	24	150	26	176	1.00E-02
5	10	66	124	190	4.90E-07	1751	10512	1753	12265	1.30E-04
6	5	36	70	106	2.30E-06	30	186	32	218	5.60E-03
7	12	78	115	193	2.60E-07	2687	16128	2689	18817	3.95E-03
8	9	60	102	162	5.60E-07	20	126	22	148	8.97E-03
9	7	48	78	126	2.60E-06	23	144	25	169	1.40E-03
10	6	42	96	138	3.90E-08	23	144	25	169	1.80E-03
Mean	10.8	70.8	124.5	194.3	2.75E-06	463.6	2787.6	465.6	3253.2	0.005568
Median	9.5	63	108.5	176	9.8E-07	25.5	159	27.5	186.5	0.004775
Standard Deviation	6.033	36.199	59.454	93.263	3.81E-06	951.121	5706.727	951.121	6657.849	0.004892
Range	17	102	168	264	9.86E-06	2667	16002	2667	18669	0.01487
Minimum	3	24	54	78	3.9E-08	20	126	22	148	0.00013
Maximum	20	126	222	342	9.9E-06	2687	16128	2689	18817	0.015

Table 4.7 Comparison with standard back propagation method (2-2-1 ANN: XOR problem: small rage weight)

Refer to the Table 4.7 and note the maximum and minimum numbers of epoch are 20 and 3 respectively. The algorithm in some experiment trains the XOR with as low as 3 numbers of epoch. The maximum and minimum numbers of total function evaluations including gradient evaluations are 342 and 78 respectively. The algorithm that purely operates on gradient information should take into account the efforts of gradient evaluations including function evaluations.

In standard back propagation training, the maximum and minimum numbers of epoch are 2687 and 20 respectively. The corresponding figures for the total number of function evaluations are 18817 and 148 respectively.

4.7.5 Comparison with Standard BP Method

The simulation results show that the convergence of the algorithm is influenced by the small magnitude of the starting points. The average number of epoch is 10.8 when experimented with small magnitude of starting weights. The simulation with wide range of

starting points results average number of epoch as 151.8. The proposed method trains the 2-2-1 XOR problem efficiently with small magnitude of random starting points.

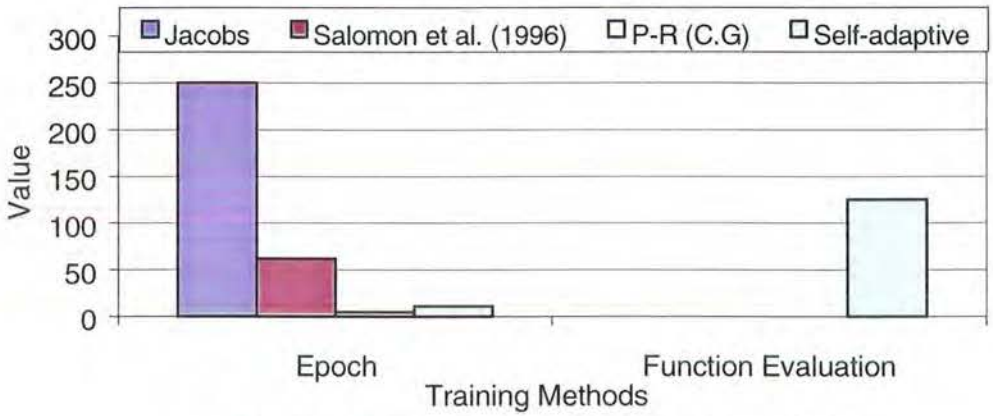


Figure 4.4 Comparison with different training methods

The results suggest that the proposed method is faster than the standard back propagation method when simulating in the small rage of starting weights. The efficiency of the proposed method over the standard back propagation method in average epoch, gradient evaluations and function evaluations are (463.6/10.8) 42.9, (2787.6/70.8) 39.4 and (465.6/124.5) 3.74 respectively. The magnitude of the random starting points affects the performance of the proposed and the standard back propagation algorithms. The relative efficiency in total number of function evaluation is (3253.2/194.3) 16.74

The convergence behavior of the standard back propagation training method is shown in Figure 4.5. For a small size problem the convergence is pattern not oscillatory, however later it will be seen in Chapter 7 that the standard BP training suffers from oscillations with the large size problem or where the convergence becomes difficult. In a recent paper Ampaziz et al. (1999) study the convergence characteristics with eigenvalue analysis.

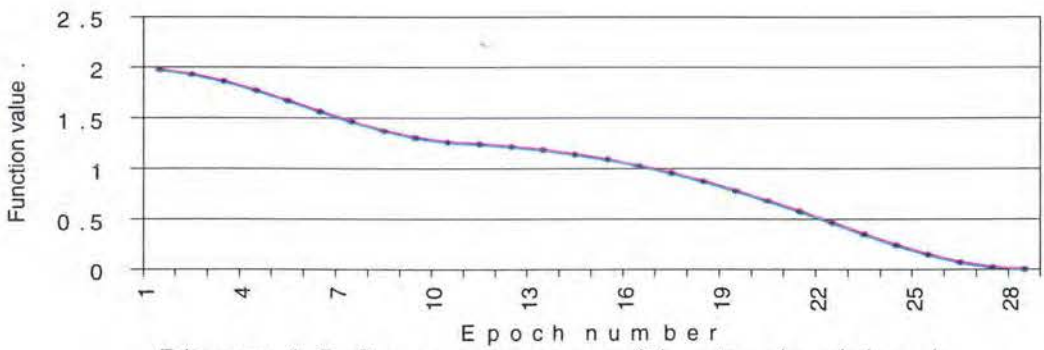


Figure 4.5 Convergence with standard back propagation method (XOR problem)

	Proposed Method					Standard Back Propagation Method					Jacobs (1988)	Salomon (1996)
Experiment #	Epoch	Gradient evaluations	Function evaluations	Total Function Evaluations	Terminal function value	Epoch	Gradient evaluations	Function evaluations	Total Function Evaluations	Terminal function value	Epoch	Epoch
Mean	10.8	70.8	124.5	194.3	2.75E-6	463.6	2787.6	465.6	3253.2	0.00557	250	62
Median	9.5	63	108.5	176	9.8E-7	25.5	159	27.5	186.5	0.00477		
Standard Deviation	6.0332	36.199	59.454	93.263	3.81E-6	951.12	5706.73	951.12	6657.85	0.0049	60	
Range	17	102	168	264	9.86E-6	2667	16002	2667	18669	0.01487		
Minimum	3	24	54	78	3.9E-8	20	126	22	148	0.00013		
Maximum	20	126	222	342	9.9E-6	2687	16128	2689	18817	0.015		

Table 4.8 Comparison with other method (2-2-1 ANN XOR Problem)

4.7.6 Comparison With Results in Literature

The simulation results are compared in Table 4.8 using the reports available in Jacobs (1988) and Salomon et al. (1996) with XOR problem. We compare the performance of the proposed algorithm that is initiated with random starting weights in small range. The simulation results are taken from the Table 4.7. The proposed method needs 10.8 number of epoch on average to train the XOR problem. The back propagation method due to Salomon (1996) takes 62 number of epoch, while the delta bar delta method reported in Jacobs (1988) takes 250 epochs to train the XOR problem. The standard back propagation needs on average 463.6 numbers of epochs to train the 2-2-1 ANN XOR problems. It is observed that the proposed method improves over the method due to Jacobs (1988), Salomon (1996) and the standard back propagation method. The reasons for less number of epoch with the proposed method are due to the constrained interpolation search algorithm and the controlled step size computations in central difference gradient approximation scheme. The efficiency is gained, however, at the expense of function evaluations. The corresponding total numbers of function evaluations are 194.3. Kamarthi et al. (1999) report that the Polak and Ribiere conjugate gradient takes 14 epochs to train XOR problem.

4.8 Discussions

A self-adaptive gradient descent BP training method is developed. The convergence of the algorithm is proved using descent and convexity properties of the error function. A constrained interpolation search automates the selection of variable learning rates. The descent directions are computed by central difference approximation scheme. The step size of the difference scheme is controlled by the convergence behavior of the error function. It computes suitable descent directions. The simulation with the small random

starting weights show that the average number of epoch, gradient evaluations and function evaluations are 10.8, 70.8 and 124.5 respectively with 2-2-1 ANN XOR problem. The mean terminal function value is low and therefore, better training results are obtained. The proposed algorithm improves over the standard BP algorithm in number of epoch, function evaluations and terminal function value.

More comprehensive analysis and results are reported in Chapter 7 with several benchmark problems in higher dimensions. Comparisons with different algorithms in training the standard benchmark problems are also reported in Chapter 7.

Chapter
5

Self-Adaptive Multi-Directional Training Without Derivatives

5.1 Introduction

An efficient multi-directional self-adaptive derivative free ANN training method, which evaluates only error function, is developed in this chapter. The error function is reduced to a sub-problem in a constrained search space and the search directions follow rectilinear and Euclidean moves. An interpolation search determines the self-adaptive parameters. To accelerate the training algorithm, a momentum search is designed. An algorithm determines the self-adaptive momentum term and hence the training method is parameter free. To improve the convergence of the algorithm an *oriented search vector* first positions the initial starting weight in a descent location. The proposed method is useful when the function $f(w_k)$ is ill conditioned, the derivatives $\frac{\partial f(w)}{\partial w_j}$ are discontinuous, or the derivative evaluation is difficult.

The multi-directional search method is discussed in Section 5.2. The new self-adaptive parameter free training algorithm is developed in Section 5.3. A restricted momentum search is designed and this step is taken only when there is improvement in function value. The momentum term is determined dynamically. It prevents the training method from overshooting the minimum. The convergence of the algorithm is discussed in Section 5.4. In Section 5.5, simulation results with the XOR problem are presented and finally Section 5.6 provides some discussions.

5.2 Multi-Directional Search

Figures 5.1 and 5.2 show the all likelihood outcomes of search in rectilinear direction when $m = 2$. Given an ANN error function $f : w \in E^m$, the rectilinear search minimizes the function $f(w)$ cyclically changing its weights moving along one direction at a time. The directions of moves are $d_1, d_2, \ldots\ldots, d_m$, where the direction vector d_j is defined as:

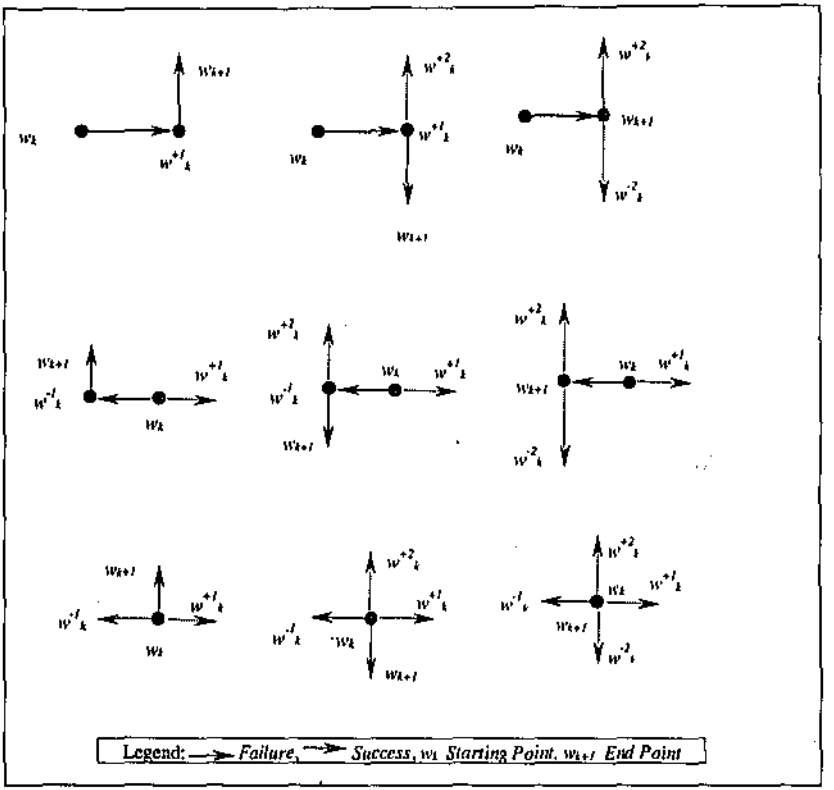


Figure 5.1 All possible rectilinear moves *in* E^2

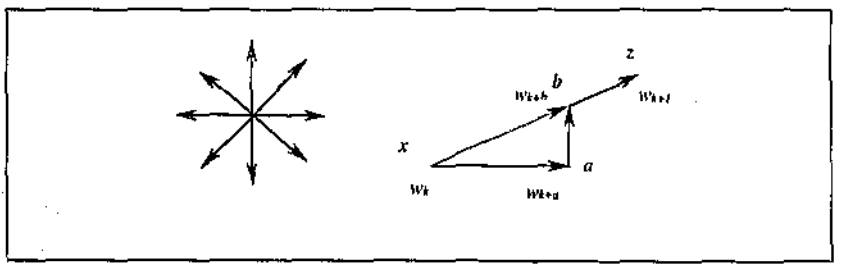


Figure 5.2 Momentum update in E^2

$$d_j = \vec{I}_{ij}, \text{ where } \vec{I}_{ij} = \begin{cases} 1 & \text{for } j = i \\ 0 & \text{for } j \neq i. \end{cases}, \; i = 1,2,\ldots,m, \; j = 1,2,\ldots,m \; . \tag{5.1}$$

Fixing a value of j, the ANN weight w_j is changed in d_j direction in small amount say, Δw_j, while all other weights are kept fixed. Moving in all m directions the algorithm changes the value of all the weights $w_j \equiv (w_1, w_2, \ldots, w_m)$. The process is again repeated to obtain change in the function value $f(w)$. The weight update recursion is given in 5.2:

$$w_{k+1} = w_k + \eta_k d_k \; . \tag{5.2}$$

5.2.1 Rectilinear Direction Search

The training algorithm starts with a current trial vector marked as w_k and the next successful trial vector in direction j is marked w_{k+1} at a distance Δw_j. A successful trial improves training from iteration k to the next iteration $k+1$. The dotted lines along the rectilinear directions are the intermediate trial steps. The dotted lines indicate that the directions along which the error function $f(w)$ is evaluated but fails to decrease the function value. This direction is abandoned and the vector w_k remains the same. The notation w_k^1 and w_k^{-1} designate that the search directions are along the line $j = 1$ but in two opposite directions with respect to a base point. Similarly, The notation w_k^2 and w_k^{-2} indicate that the rectilinear search are along the line $j = 2$ in two opposite directions. In the worst case, there are $2m$ evaluations, which do not decrease the function value. This is shown in Figure 5.1 along third row and third column. The figure implies that $w_{k+1} = w_k$. In such a case the step size Δw_j can be changed for next phase of iteration. We define this strategy as rectilinear search. At the end of the rectilinear search the trial step w_k is redefined as w_{k+1} and it is accepted when the condition: $f(w_{k+1}) < f(w_k)$ is true.

The algorithm updates weights moving in a suitable direction d_j and identifying a learning rate parameter $\eta_j = \Delta w_j$. The simplest form of rectilinear training is to have a constant learning rate η_k. To make the algorithm dynamically self-adaptive the learning rate parameter η_k is identified by a suitable method that provides descent to the error function. The learning rate parameters are unrestricted in sign and have different magnitudes. Depending on the geometry of the error surface these parameters are calculated. The search in the same direction is continued when the chosen direction is successful, until the function value fails to improve further. The direction is then changed and the search repeats to improve the error function value.

An algorithm that describes rectilinear search is given in Table 5.1a. The algorithm selects trial steps depending on the success or failure of previous trial steps. There are 3^m possible trial steps in similar coordinate search but in practice the evaluations are in the range m to $2m$ at any given intervals (Torczon, 1997).

Step 1:	*Choose a starting vector of weight at iteration k. Define $w_k \equiv (w_j) \equiv (w_1, w_2, \ldots\ldots, w_m)$. Let η^m be the momentum term. Set $\delta_1 \leftarrow 1$, $\delta_2 \leftarrow 0.8$, $\wp \leftarrow 0.0001$, $k = j = 1$, set w_k to an initial value. Set an initial weight $w_{k-1} \leftarrow \lceil \bullet \rceil$ by large integer $> w_k$ such that $f(w_{k-1}) > f(w_k)$.*
Step 2:	*Let $(d_1, d_2, \ldots\ldots, d_m) \leftarrow 0$, set $d_j \leftarrow 0$ for the current value of j.*
Step 3	*a.) if $f(w_j + \Delta w_j d_j) < f(w_j)$, the trial is success, let, $w_j \leftarrow (w_j + \Delta w_j d_j)$ and perform step 4.* *b.) If $f(w_j + \Delta w_j d_j) \geq f(w_j)$, the trial is failure, then* *c.) if $f(w_j - \Delta w_j d_j) < f(w_j)$, the trial is a success and let, $w_j \leftarrow (w_j - \Delta w_j d_j)$, and perform step 4.* *d.) if $f(w_j - \Delta w_j d_j) \geq f(w_j)$, then trial is failure and let, $w_j \leftarrow w_j$ and perform step 4.*
Step 4	*a.) if $j < m$, set $j \leftarrow j+1$ and repeat step 3, otherwise, set $w_k \leftarrow w_j \equiv (w_1, w_2, \ldots\ldots, w_m)$ at iteration k.* *b.) if $f(w_k) < f(w_{k-1})$ perform step 5, else* *c.) if $f(w_k) \geq f(w_{k-1})$ perform step 6*
Step 5	*Momentum type or Pattern type search* *a.) $\eta_k^m \leftarrow \eta_k^m * \lceil \delta_1 \rceil$ Determine η_k^m according to the Table 5.1b.* *b.) $w_{k+1} \leftarrow w_k + \eta_k^m (w_k - w_{k-1})$. Let, $k \leftarrow k+1$, $j \leftarrow 1$, and perform step 3.*
Step 6:	*a.) If $\Delta w_j \leq \wp$ stop ($\wp$ = stopping criteria), and report w_k as the solution* *b.) Otherwise, replace $\Delta w_j \leftarrow \Delta w_j * \lceil \delta_2 \rceil$, Let, $w_{k+1} \leftarrow w_k$, Let, $k \leftarrow k+1$, $j \leftarrow 1$, and perform step 3.*

Table 5.1a Multi-directional training algorithm with fixed step size

Step 1 and 2 initializes the algorithm. In step 3, the value of the weight is changed according to the magnitude of $\Delta w_j d_j$ such that the error function is reduced. If the function value does not improve, the weight vector is left unchanged. Step 4 determines if a momentum search should be attempted. If the current iteration is successful, a new step size η'' is adjusted by the factor [δ_1] in step 5. A momentum search is designed at this stage to move the search into a new but descent location. The choice of the factor is important in successful implementation of this method. A method that updates the step size η'' is developed in Ahmed and Cross (1999) and is shown in Table 5.1b. The suggested method varies the step size with the objective of identifying trial points that brackets the minimum with other relative minimums in a neighborhood. The method given in Table 5.1b determines the step size by repeated evaluation of the error function. The step 6 describes the method of updating step length, when the current phase of evaluation, k, is not improving the function value. The step is decreased by a factor [δ_2]. The training algorithm successfully models seasonal load forecasting problem (Ahmed and Cross, 1999; 2000).

1. *One weight parameter w, in a specified direction is incremented at a time by an amount Δw, where the magnitude of the quantity Δw is determined and the sign is chosen such that the ANN error function is decreased.*

2. *The parameter w is repeatedly incremented by some amount until the error surface begins to increase in chosen direction.*

3. *The function is evaluated at regular interval during this repeated trial and the evaluated function values determine the minimum of a quadratic function. Consider w', w'' and w''' as the points in specified intervals that defines a function in E^1. These points are defined as: $w'' = w' + \Delta w$, such that $f(w') > f(w'')$ and $w''' = w' + 2\Delta w$, such that $f(w''') > f(w'')$*

4. *The minimum of the quadratic function is determined by (Mathews, 1992)*

$$\eta''' = w' - \left[\Delta w \frac{4f(w'') - 3f(w') - f(w''')}{4f(w'') - 2f(w') - 2f(w''')}\right]$$

5. *Repeat the process until precision is reached in finding η''' and obtain finally $\eta'''_{best} = \eta'''$.*

Table 5.1b The interpolation method to determine self-adaptive parameter

5.2.2 Self-Adaptive Training With Momentum Search

Refer to the Figure 5.2 and consider a direction from x to b in order to move to a position z without performing rectilinear search at point b. The method attempts to accelerate the training using the previous successful training steps. The current evaluation begins by performing a search along the direction $(w_k - w_{k-1})$, if the trial $f(w_k) < f(w_{k-1})$ is successful. A momentum search similar to the pattern move (Hooke and Jeeves, 1961) along the direction $\vec{d}_k = (w_k - w_{k-1})$ from w_k is performed only when the condition $f(w_k) < f(w_{k-1})$ is true. Figure 5.3 shows the two solid arrows as the two successive rectilinear moves that are success. The dotted line is the move by the momentum search. The decision about the length of move is an important issue. If we take a large step, the minimum will be missed. In the following section we propose a method that takes a step just close to the neighborhood of the minimum. The function is evaluated at this trial step and the method again searches from the current position in directions $\vec{d}_k = \eta_k^m (w_k - w_{k-1})$. The recommended weight update rule according to this method is shown below:

$$w_{k+1} = w_k + \vec{d}_k \eta_k^m = w_k + \eta_k^m (w_k - w_{k-1}). \tag{5.3}$$

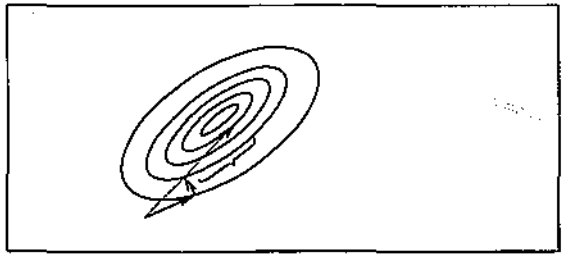


Figure 5.3 Determination of momentum term as self-adaptive parameter

The momentum weight update step is only performed when the algorithm iterated more than one epoch and when the current trial step is successful with reference to the previous move. It is a restricted step and executed only when there is success. This step forces the training algorithm to reach the neighborhood of the minimum. The direction of successive momentum search tends to become aligned with the ridges in error surface as long as the method successfully follows the path according to the Figure 5.2 and Figure 3.1 in Chapter 3.

The term η_k^* is the momentum term and its approximate magnitude determines the convergence of the algorithm. If the rectilinear search about this position is successful then the vectors generated by rectilinear search is accepted as w_{k+1} according to the Figure 5.2 and 5.3. The momentum term is determined from the scheme given in Table 5.1b. If the search fails to improve the function value, then the method reduces to rectilinear search at w_k. The search along the direction x to b at z is the momentum search, and can be considered as a variant of the multidirectional search (Torczon, 1997). The algorithm that implements this training method is shown in Table 5.1a.

5.3 Dynamic Self-Adaptive Training

The method described in Section 5.2.1 through 5.2.4, when combined together constitutes the multi-directional search. Now we develop a new training algorithm that is dynamically self-adaptive. Fixing a search direction, the rectilinear training finds a learning rate, η_j for the error function. To determine the suitable magnitude of learning rate η_j, an interpolation search is performed in a given direction $j = 1,2,\ldots,m$. The search in a constrained space is formulated as:

$$\text{minimize } f(w_j + \eta_j d_j) \tag{5.4a}$$

$$\text{subject to: } \eta_j \in L \ . \tag{5.4b}$$

where, L is expressed by $L = E^1$. The search along any rectilinear direction either must yield a decrease or by the assumption, it cannot change position. Since the error function is a descent function (Hecht-Nielsen, 1990), we must have decrease in function value in at least one direction. If at a point the gradient $\nabla f(w) \neq 0$, then at least one component of $\nabla f(w) \neq 0$ does not vanish and hence a search along the corresponding rectilinear direction must yield decrease. Fixing the value of the weight w_j and the direction d_j in all rectilinear direction except in the current search direction j, the problem 5.4 is solved according to the proposition 4.4 in Chapter 4 to find the learning rate. Step 4 in Table 5.2 implements this algorithm. The proposed algorithm dynamically self-adapts the training parameters.

5.3.1 Automatic Determination of Momentum Term

It is noted earlier that there exists a descent direction of the form $\vec{d}_k = (w_k - w_{k-1})$. The appropriate magnitude of the momentum term η_k^* determines the step length in multi-direction search and does not destroy the search direction. This is done by the interpolation search according to the problem:

$$\text{minimize } f(w_k + \eta_k^m \bar{d}_k) \tag{5.5a}$$

$$\text{subject to: } \eta_k^m \in L. \tag{5.5b}$$

The value of η_k^m is obtained from the appropriate interpolation search method shown in proposition 4.4 in Chapter 4 and L is of the form $L = E^1$. Step 5 in Table 5.2 determines this parameter using the interpolation search algorithm developed in Chapter 4. It takes regulated steps near the minimum and convergence is not at risk. Equation 5.5 determines single momentum term, while in Equation 5.4 there are m different variable learning rates.

5.3.2 Self-Adaptive Derivative Free Multi-Directional Training Method

The algorithm that automates the learning rate parameter and momentum term is shown next in Table 5.2. The algorithm is initialized in step 1.

5.3.2.1 Oriented Search

The training is initiated by an oriented search to accelerate convergence. Initially, an oriented search is performed in step 2 (Table 5.2), where the directions vector d_j in -1 and $+1$ are generated randomly using uniform distribution according to the Equation 5.6:

$$d_j \leftarrow \begin{cases} -1 & \text{if } 0 \le u(0,1) \le 0.5 \\ 1 & \text{if } 0.5 < u(0,1) \le 1. \end{cases} \tag{5.6}$$

The uniform distribution generates a number between 0 and 1. A value between 0 and 0.5 assigns the direction vector a value -1 and $+1$ otherwise. The purpose is to locate the initial descent position such that there are minimum efforts to reach the local minimum. Care should be taken to prevent ties in generating the direction vector in a given set of m or multiple of m trials. Evaluate the error function with the generated direction vector and retain the minimum weight vector along with the direction vector in a separate vector denoted as $w^{(ini)} \leftarrow d_j w_j$, which minimizes the error function. Perform this step m times or multiples of m times depending on the parameter setting. In step 3 (Table 5.2) the preparation for the main search begins with the oriented search vector $w^{(ini)}$.

5.3.2.2 Self-Adaptive and Momentum Parameter Determination

An interpolation search is performed in step 4 (Table 5.2) to determine the learning rate. This makes the training method fully self-adaptive. At the end of step 4, the momentum

Step: 1 *Initialization*

a.) Let $f_{min} \leftarrow \infty$, *set* $j \leftarrow 0$, *set* $d_j \equiv (d_1, d_2, \ldots\ldots, d_m) \leftarrow 1$

Step:2 *Position the Search Vector*

Generate an index $j \in m$ *from uniform distribution and* d_j *as the direction of search in the following manner.*

a.) $j \leftarrow j+1$

b.) $d_j \leftarrow \begin{cases} -1 & \text{if } 0 \le u(0,1) \le 0.5 \\ 1 & \text{if } 0.5 < u(0,1) \le 1 \end{cases}$ *(generate direction using uniform distribution)*

c.) set $\eta_j d_j \leftarrow d_j$

d.) evaluate $f \leftarrow f(w_j + \eta_j d_j)$, *if* $f < f_{min}$ *set* $f_{min} \leftarrow f$ *and* $w^{int} \leftarrow w_j d_j$

e.) if $j < m$ *repeat the step 2.*

f.) If $j = m$, *perform next step.*

Step:3 *Prepare for Main Phase*

a.) Set $k = j = 1$, *set* $w_j \leftarrow w^{ini}$ *and perform step 4.*

Step:4

a.) Set $d_1 = d_2 = d_3, \ldots\ldots, d_m = 0$ *and Let* η_j *be an optimal solution to the problem*

b.) $\min_{\eta_j \in E^1} f(w_j + \eta_j d_j)$ *and set* $d_j \leftarrow 1$ *at current index* j.

c.) Assign $w_j \leftarrow (w_j + \eta_j d_j)$.

d.) If $j < m$, *set* $j \leftarrow j+1$, *set and repeat step 4.*

e.) Else if $j = m$, *set* $w_{k+1} \leftarrow w_j$.

f.) If $(w_{k+1} - w_k) < \wp$ *then stop* ($\wp$ = *stopping criteria)*

g.) Otherwise, perform step 5.

Step:5 *Pattern or Momentum Search:*

a.) let $d_k \leftarrow (w_{k-1} - w_k)$ *and let* η_k *be the optimal solution to the problem*

b.) $\min_{\eta^* \in E^1} f(w_k + \eta_k^* d_k)$

c.) Set $\eta_k \leftarrow \eta_k^*$, $w_{k+1} \leftarrow w_k + \eta_k d_k$, *let* $j \leftarrow 1$, $k \leftarrow k+1$ *and perform step 4*

Table 5.2 Multi-directional training algorithm with interpolation search

search is performed. The interpolation search method developed in Chapter 4 Section 4.3 is used to identify the momentum term. Therefore, there is no user intervention to pre-optimize the learning rate and momentum term. The derivative information of the error function is not required in this training method. The error functions that are discontinuous can be trained with the proposed method. In general, the training method is derivative free.

5.4 Convergence of the Training Method

To address the convergence of the algorithm, we assume that the error function possess descent property and satisfy first order necessary condition. It will be shown next in a proposition that the algorithm converges to zero gradients under the hypothesis that the error function $f(w)$ is differentiable.

Proposition 5.1

Define the multi-directional training algorithm as a map M according to the definitions in Chapter 4 Section 4.1. The algorithmic map M generates the sequence of vector $\{w\}$ according to $w \in M(w)$ moving along the directions $d_1, d_2, \ldots\ldots, d_m$. The search starts at iteration k with the vector w_k and each direction $d_1, d_2, \ldots\ldots, d_m$ has a norm 1. Suppose that the following properties are true:

a.) There exist a $\wp > 0^+$ such that $\det[d(w)] \geq \wp$ for each $w \in E^m$ and $[d]$, is a $m \times m$ matrix whose columns are the search directions generated by the algorithm and the determinant of $[d]$ is represented by $\det[d]$. Further assume that the search directions are linearly independent.

b.) The minimization of $f(w)$ along any direction in E^m is possible.

Given a $w \in E^m$, suppose that the algorithm generates the sequence $\{w_k\}$ such that if $\nabla f(w_k) = 0$, then the algorithm stops with w_k otherwise, $w_{k+1} \in M(w_k)$ and replace k by $k+1$ and repeat the process. Consider $\Omega \in \{w_k\}$ is a solution set. If the sequence $\{w_1, w_2, \ldots\ldots, w_k, \ldots\}$ is contained in a compact subset of E^m, then each accumulation point $w^* \in \Omega$ of the sequence $\{w_k\}$ must satisfy $\nabla f(w^*) = 0$.

Proof

If the sequence $\{w_k\}$ is finite, then the algorithm converges to a solution and the gradient approaches to a small value, due to the descent property of $f(w)$. Suppose that the

algorithm generates the sequence $\{w_k\}$, which is infinite. Let k_1 be an infinite sequence of positive integers and assume that the sequence $\{w_k\}_{k_1}$ converges to a limit point w^*. Suppose by contradiction that $\nabla f(w^*) \neq 0$ and take the sequence $\{w_{k+1}\}_{k_1}$ that is contained in a compact subset of E^m. It implies that $\{w_{k+1}\}_{k_1} \subseteq E^m$ and there exist a subsequence $k_2 \subseteq k_1$ such that $\{w_{k+1}\}_{k_2}$ converges to some point defined as $w^* \in \{w_k\}$. The vector w^* is obtained from minimizing the function $f(w)$ along a set of m linearly independent directions $(d_1, d_2, \ldots\ldots, d_m)$. Let d_k be the $n \times m$ matrix and the search directions $(d_1, d_2, \ldots\ldots, d_m)$ are its column at iteration k. Thus, the algorithm generates the sequence of training weights according to the expression: $w_{k+1} = w_k + [d_j \eta_j]$ or equivalently: $w_{k+1} = w_k + \sum_{j=1}^{m} [d_j \eta_j]_k$. The quantity η_j is the learning rate, which is equivalent to a distance moved along d_j at iteration k. The vector d_k is the descent direction and also from proposition 4.3 in Chapter 4, we have $f(w_{k+1}) \leq f(w_k + [\eta_j d_j]_k) : \eta_j \in E^1, j = 1,2,\ldots,m$. We also have $\det[d_j] \geq \wp > 0^+$ and $[d_j]$ is invertible, hence $\eta_j = [d_j]^{-1}(w_{k+1} - w_k)$ for all j at iteration k. Since each column of $[d_j]$ has norm 1, there exist a quantity $k_3 \subseteq k_2$ for a subsequence such that we have the limit $[d_j] \to [d]$. Since $\det[d_j] \geq \wp$ for each k, $\det[d] \geq \wp$, so that $[d]$ is invertible. Now for $k \in k_3$ and as $k \to \infty$, we have: $w_{k+1} \to w^*$, $w_k \to w^*$, $[d_j]_{k+1} \to [d]$ so that the learning rate converges to a limit point $\eta_j \to \eta$, where $\eta = [d]^{-1}(w^* - w^*)$. Therefore, $w^* = w^* + [d\eta] = w^* + \sum_{j=1}^{m} [d_j \eta_j] ; \eta_j \in E^1, j = 1,2,\ldots\ldots,m$. The vector, w^*, is obtained from minimizing $f(w)$ sequentially along the directions $(d_1, d_2, \ldots\ldots, d_m)$, it follows then $f(w^*) \leq f(w)$. Now consider the case $f(w^*) < f(w)$, since $\{f(w_k)\}$ is a decreasing sequence and since $f(w_k) \to f(w)$ as $k \in k_1$ approaches to ∞, we can also write $\lim_{k \to \infty} f(w_k) = f(w)$. This is not possible, since $w_{k+1} \to w^*$ as $k \in k_2$ approaches to ∞ and by assumption $f(w^*) < f(w)$. Again consider the case, $f(w^*) = f(w)$. We obtain w^* from the vector w, minimizing $f(w)$ in direction $(d_1, d_2, \ldots\ldots, d_m)$ and by the property b of the proposition we get $w^* = w$. This also implies that $\nabla f(w)^T d_j = 0, j = 1,2,\ldots,m$. Since $(d_1, d_2, \ldots\ldots, d_m)$ are linearly independent, $\nabla f(w^*) = 0$, contradicting our assumption. This proves the proposition $\Diamond$.

5.4.1 Characteristics of the Multi-directional Self-Adaptive Training

Two types of training parameter search characterize the proposed training method. There is self-adaptive parameter search followed by a momentum term search. Based on the assumptions that the line joining the first and last points of the rectilinear move represents an especially favorable direction. In Figure 5.2, the direction from x to z is the momentum search, while the search at c and b are the rectilinear search. The rectilinear and momentum search do not always lead to an improvement in the error function value. The success of the iteration is only checked after the rectilinear move has taken place.

The magnitude of the momentum term η_k^* is determined from the interpolation search method according to the proposition 4.4 in Chapter 4. The important feature of this method consists of moving along the ridges and valleys. The momentum move takes long step in the direction of valleys and the rectilinear move finds the path back to these valleys when a momentum move has climbed out of them. The method is successful in solving a forecasting problem, which is ill conditioned (Ahmed and Cross, 1999d; Ahmed and Cross, 2000e). Additional results are given in Chapter 7.

5.4.2 Convergence of the Algorithm to Local Minimum

The training algorithm only evaluates the error function value. Moving along the rectilinear and momentum direction the algorithm provides descent to the error function. The search along the momentum direction is assumed similar to the coordinate search (Torczon, 1997). The successive move along the rectilinear directions with the application of the interpolation search identifies a local minimum. It is shown in the following proposition.

Proposition 5.2

Let $w^* \in \Omega$ be the point in a solution set to the error function $f : w \in E^m$. The sequence $\{w_1, w_2, \ldots\ldots, w_k, \ldots\}$ generated by the algorithm converges to the local minimum w^*.

Proof

Suppose that $\{w_k\}$ is an infinite sequence generated by the algorithm defined as $w_{k+1} = M(w_k)$ and if at any iteration $w_k \in \Omega$, then the algorithm stops. Consider that $\{w_k\}_K$ is a convergent subsequence with limit w^*. Further assume that the value of k is larger than the integer number k_1 and K denotes the subset of all positive integers. Since $f(w_k)$ is descent function $f(w_k) \to f(w^*)$ for some $k \in K$ of the subsequence. Then for a given $\varepsilon^+ > 0$ there exists a relation $f(w_k) - f(w^*) < \varepsilon$ for $k \geq k_1$ with $k \in K$. If we let, $k = k_1$,

then $f(w_{k_1})-f(w^*)<\wp$. With $k>k_1$ and $k\in K$ the expression $f(w_k)<f(w_{k_1})$ exists, since $f(w_k)$ is a descent function. Therefore, with the above results the following expression is derived for every $k>k_1$.

$$f(w_k)-f(w^*)=f(w_k)-f(w_{k_1})+|f(w_{k_1})-f(w^*)|<0+\wp=\wp. \tag{5.7a}$$

Since $\wp^*>0$ is arbitrary, it follows then

$$\lim_{k\to\infty} f(w_k)=f(w^*). \tag{5.7b}$$

Also M is closed at w^* by definition (Section 4.3 and proposition 4.3 in Chapter 4) for $k\in K$, $w_k\to w^*$. Again consider the error function in a small neighborhood at iteration k at a distance $\eta>0$ and define $\nabla f(w_k)=\frac{f(w_k)-f(w^*)}{\eta}$ according to the proposition 4.1 and 4.2 in Chapter 4. Since $f(w_k)$ is a descent function and $\lim_{k\to\infty} f(w_k)=f(w^*)$. The overall algorithmic map M is closed. Therefore, the training converges to the zero gradients at termination and hence the limit point w^* is the local minimum $\Diamond$.

5.5 Analysis with the XOR Problem

Figure 5.4 shows the convergence of the algorithm against the number of epoch/iterations and momentum term. The self-adaptive parameters that are generated during training are shown in Figure 5.5. The training initiates with starting vector $(0.22, 0.34, 0.97, 0.73, 0.1, 0.2)^T$ and the sample calculations are shown in Table 5.3. It shows that the self-adaptive learning rate and momentum parameters determined by the algorithm approaches to a small value as the training progress. The self-adaptive parameters reduce the function value monotonically, which is shown in Figure 5.4.

Epoch no.	Self-Adaptive Parameter: 1	Self-Adaptive Parameter: 2	Self-Adaptive Parameter: 3	Self-Adaptive Parameter: 4	Self-Adaptive Parameter: 5	Self-Adaptive Parameter: 6	Momentum Term	Function Value
1	-0.05442	-0.0641	-0.53011	-0.60881	-0.24987	-0.03935	-0.00469	0.591136
2	0.05427	0.047	0.018047	-0.01618	0.002517	0.000146	0.002194	3.25E-05
3	0.00027	-1.4E-05	0.000255	-0.00192	0.000311	-1.4E-05	0.145158	1.92E-07
4	3.45E-06	-7.8E-07	6.87E-07	-6.1E-06	-3.4E-07	-3.5E-06	-0.00414	1.01E-010

Table 5.3 Self-adaptive parameters, momentum term and function value with starting vector (.22, .34, .97, .73, .1, .2)^T

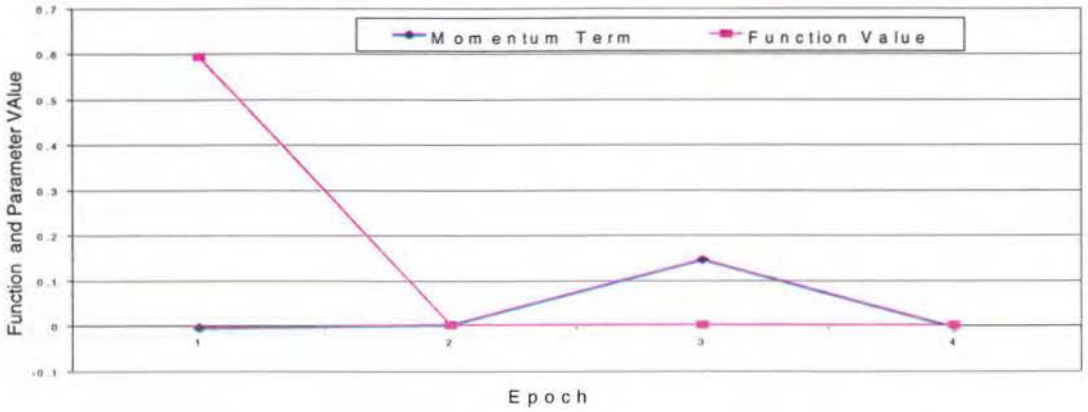


Figure 5.4 Momentum Term and Function Value

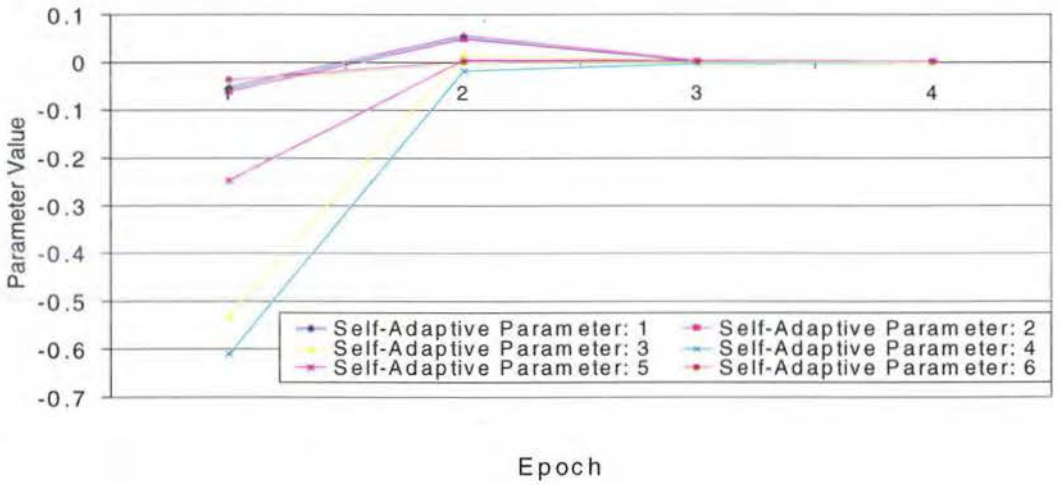


Figure 5.5 Self-Adpative Parameters

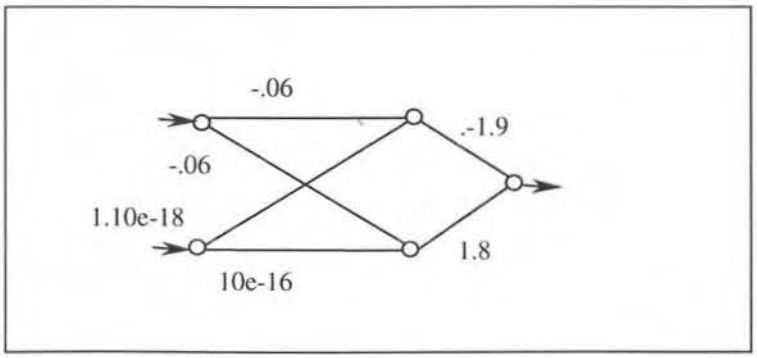


Figure 5.6. A trained 2-2-1 ANN XOR

The proposed training method is tested with 2-2-1 ANN configuration XOR problems. The input data for the problem corresponds to the subset of the Table A.1 in Appendix A in rows 1, 2, 3, 4 and column 4, 5. The random starting points $w_j = (w_1, w_2, \ldots\ldots, w_6)$ are taken from the Table A.2 in Appendix A. The learning rate is 0.1 for the standard back propagation training. The proposed method is self-adaptive, parameter and derivative free.

5.5.1 Sample Calculations with 2-2-1 ANN XOR Problem

Figure 5.6 shows a trained 2-2-1 ANN configuration. Sample calculations are show in Table 5.4 with the trained ANN configuration. The instance of zero input to the hidden layer neuron is approximated to the value 1×10^{-5} for computational convenience.

$x_1^{p=1} = 0$, $x_2^{p=1} = 0$, $h_1^{p=1} = (-.06)*0 + (.06)*0 = 0$, $h_2^{p=1} = (10^{-18})*0 + (10^{-16})*0 = 0$

$g_1^{p=1} = \frac{1}{1+|\ln 10^{-5}|} = -0.078$, $g_2^{p=1} = \frac{1}{1+|\ln 10^{-5}|} = -0.078$,

$z_1^{p=1} = (-1.9)*(-0.078) + (1.8)*(-0.078) \cong 0.0078$

$x_1^{p=2} = 1$, $x_2^{p=2} = 0$, $h_1^{p=2} = (-.06)*1 + (.06)*0 = .06$, $h_2^{p=2} = (10^{-18})*1 + (10^{-16})*0 = 10^{-18}$

$g_1^{p=2} = \frac{1}{1+|\ln 1.06|} = -0.5514$, $g_2^{p=2} = \frac{1}{1+|\ln 10^{-18}|} = -0.0247$,

$z_1^{p=2} = (-1.9)*(-0.5514) + (1.8)*(-0.0247) \cong 1.0032$

$x_1^{p=3} = 0$, $x_2^{p=3} = 1$, $h_1^{p=3} = (-.06)*0 + (.06)*1 = .06$, $h_2^{p=3} = (10^{-18})*0 + (10^{-16})*1 = 10^{-16}$

$g_1^{p=3} = \frac{1}{1+|\ln 1.06|} = -0.5514$, $g_2^{p=3} = \frac{1}{1+|\ln 10^{-16}|} = -0.0279$,

$z_1^{p=3} = (-1.9)*(-0.5514) + (1.8)*(-0.0279) \cong 0.9974$

$x_1^{p=4} = 1$, $x_2^{p=4} = 1$, $h_1^{p=4} = (-.06)*1 + (.06)*1 = 0$, $h_2^{p=4} = (10^{-18})*1 + (10^{-16})*1 = 1.01 \times 10^{-16}$

$g_1^{p=4} = \frac{1}{1+|\ln 10^{-5}|} = -0.078$, $g_2^{p=4} = \frac{1}{1+|\ln 0.01 \times 10^{-16}|} = -0.0279$,

$z_1^{p=4} = (-1.9)*(-0.078) + (1.8)*(-0.0279) \cong 0.098$

Table 5.4 Sample calculations with 2-2-1 trained ANN XOR

5.5.2 Random Starting Vector in Wide Range

The self-adaptive training parameters and momentum parameters are computed according to the method discussed in Chapter 4 Section 4.3. Table 5.5 shows the results with random starting points that are according to the Table A.2 in Appendix A. The average epoch and function evaluations are 6.3 and 1122.6 respectively. The training method is not sensitive to the magnitude of random starting points. Compare the experiment numbers 1, 2, 9 and

10 with different magnitudes of starting weights. The experimental results in Table 5.5 are fairly consistent. The average values of the epoch and total number of function evaluations with the standard BP method are 5142.3 and 36006 respectively. The relative efficiency of the proposed method over the standard BP method is (5142.3/6.3) 816 in number of epoch and it is (36006/11226) 32.07 in number of function evaluations. The comparison with the number of epoch is not appropriate since, the proposed method uses a specialized interpolation search for convergence and consequently, the number of epoch is reduced. The total number of function evaluation is a valid metric.

	Multi-Directional Training			Standard Back Propagation Method				
Experiment #	Epoch	Function evaluations	Terminal function value	Epoch	Gradient evaluations	Function evaluations	Total No. of function evaluations	Terminal function value
			x1.0E-010					
1	4	586	1.65	22	138	24	162	6.20E-04
2	4	546	1	5867	35208	5869	41077	8.20E-04
3	4	684	1	29	180	31	211	1.50E-02
4	21	4110	6.1	17434	104610	17436	122046	5.10E-03
5	7	1303	0.003	4474	26850	4476	31326	2.50E-03
6	5	844	0.095	5737	34428	5739	40167	4.60E-03
7	5	949	1.1	2137	12828	2139	14967	3.00E-03
8	5	869	0.5	13024	78150	13026	91176	8.90E-04
9	4	723	1.01	15	116	17	133	2.90E-03
10	4	612	2.3	2684	16110	2686	18796	1.00E-02
Mean	6.3	1122.6	1.4758	5142.3	30861.8	5144.3	36006.1	0.004543
Median	4.5	783.5	1.005	3579	21480	3581	25061	0.00295
Standard Deviation	5.25	1073.02	1.762	5852.45	35112.73	5852.45	40965.18	0.00461
Range	17	3564	6.09	17419	104494	17419	121913	0.01438
Minimum	4	546	0.003	15	116	17	133	0.00062
Maximum	21	4110	6.1	17434	104610	17436	122046	0.015

Table 5.5 Comparison with BP Method (2-2-1 ANN XOR Problem) with wide range of starting points

The median performance of the proposed algorithm with the number of epoch and total number of function evaluation corresponds to the values 4.5 and 783.5, while with the standard back propagation method these counts are 3579 and 25061 respectively. The standard deviation in number of epoch is 5.2 with the multi-directional training method and is low in magnitude. It suggests that the training show consistent performance.

The maximum and minimum numbers of epoch are 21 and 4 respectively with the proposed method, while the maximum and minimum numbers of total function evaluations including gradient evaluations correspond to the values 4110 and 546 respectively.

The simulations with the standard BP method show that the maximum and minimum numbers of epoch are 17434 and 15 respectively. The corresponding figures for the total number of function evaluations are 122046 and 133 respectively.

It is noticed in Chapter 4 that the standard back propagation method performs well with small random starting points. The proposed multi-directional training method is not sensitive to the initial starting vector. Experiments number 1 and 2 have small starting vectors while experiment number 9 and 10 have large starting vectors. In most of the cases the experiments suggest that the magnitude of the weights do not influence convergence of the algorithm. The simulation results are compared with the standard back propagation method in small range of weights next.

5.5.3 Performance of Training with Random Weight in Small Range

Table 5.6 compares the performance of the proposed algorithm with standard back propagation method. The small random starting vectors are used to train the 2-2-1 ANN XOR problems.

	Multi-Directional Training			Standard Back Propagation Method				
Experiment #	Epoch	Function evaluations	Terminal function value	Epoch	Gradient evaluations	Function evaluations	Total No. of function evaluations	Terminal function value
1	4	586	1.66E-10	25	156	27	183	6.18E-04
2	4	577	5.87E-09	27	168	29	197	8.21E-03
3	4	684	1.00E-10	26	162	28	190	1.50E-02
4	5	842	2.60E-07	24	150	26	176	1.00E-02
5	4	510	1.00E-10	1751	10512	1753	12265	1.30E-04
6	4	652	1.02E-10	30	186	32	218	5.60E-03
7	5	831	5.10E-08	2687	16128	2689	18817	3.95E-03
8	5	793	3.30E-10	20	126	22	148	8.97E-03
9	4	588	3.50E-09	23	144	25	169	1.40E-03
10	4	579	1.20E-09	23	144	25	169	1.80E-03
Mean	4.3	664.2	3.22E-08	463.6	2787.6	465.6	3253.2	0.005568
Median	4	620	7.65E-10	25.5	159	27.5	186.5	0.004775
Standard Deviation	0.483	118.85	8.16E-08	951.12	5706.737	951.12	6657.849	0.004892
Range	1	332	2.6E-07	2667	16002	2667	18669	0.01487
Minimum	4	510	1E-10	20	126	22	148	0.00013
Maximum	5	842	2.6E-07	2687	16128	2689	18817	0.015

Table 5.6 Comparison with BP method (2-2-1 ANN XOR Problem) with small random starting points

The experimental results suggest that the proposed method converges faster than the standard back propagation method. The relative efficiencies of the proposed method over the standard back propagation methods in average epoch size and function evaluations are (463.6/4.3) 107.8 and (3253.2/664.2) 4.9 respectively. The maximum and minimum numbers of epochs are 5 and 4 respectively and the corresponding numbers of function evaluations are 842 and 510 respectively. The median performance of the standard back propagation is however better than the proposed training method in total number of function evaluations. The median values of the total number of function evaluations with the proposed method and the standard back propagation method are 620 and 186.5 respectively. The mean terminal function value with the proposed method is 3.22×10^{-8} and is comparatively less than the standard back propagation method. The corresponding value with the back propagation training method is 0.005568.

5.5.4 Comparison with Results in Literature

Table 5.7 compares the results with the proposed method, standard back propagation method and the results found in Jacobs (1988) and Salomon (1996) with the XOR problem. The training initiates with small random starting points.

	Multi-Directional Training			Standard Back Propagation Method					Jacobs (1988)	Salomon (1996)	C.G (P.R)
Experiment #	Epoch	Function evaluations	Terminal function value	Epoch	Gradient evaluations	Function evaluations	Total Function Evaluations	Terminal function value	Epoch	Epoch	Kamarthi et al. (1999), Epoch
Mean	4.3	664.2	3.22E-08	463.6	2787.6	465.6	3253.2	0.00557	250	62	14
Median	4	620	7.65E-10	25.5	159	27.5	186.5	0.00478			
Standard Deviation	0.483	118.85	8.16E-08	951.12	5706.73	951.12	6657.85	0.0049	60		
Range	1	332	2.6E-07	2667	16002	2667	18669	0.01487			
Minimum	4	510	1E-10	20	126	22	148	0.00013			
Maximum	5	842	2.6E-07	2687	16128	2689	18817	0.015			

Table 5.7 Comparison with other method (2-2-1 ANN XOR problem)

The proposed method needs 4.3 epochs on average to train the 2-2-1 ANN XOR problem. The back propagation method due to Salomon (1996) takes 62 epochs, while the delta bar delta method reported in Jacobs (1988) takes 250 epochs to train the XOR problem. The standard back propagation method needs on average 463.6 number of epochs to train the 2-2-1 ANN XOR problem. The proposed method out performs the delta bar delta method reported in Jacobs (1988), the method suggested in Salomon (1996) and the standards back propagation method in average epoch. The reason for less number of epochs with the proposed method is due to the interpolation search in addition to the momentum search that has been made dynamically self-adaptive. In addition, the proposed multi-directional

search method performs better when the error function is ill conditioned and has stiff ridges (Ahmed and Cross, 2000). The efficiency in average epoch size is gained at the cost of function evaluations. The corresponding total numbers of function evaluations are 664.2 and are still significantly less compared to the number 3253.2 with the standard back propagation method. The relative efficiency of the method against the standard back propagation method is (463.6/4.3) 107.8. The corresponding figures against the method due to Jacobs (1988), Salomon (1996) and Polak and Ribiere (reported in Kamarthi et al., 1999) are (250/4.3) 58.14, (62/4.3) 14.4 and (14/4.3) 3.26 respectively.

Finally, the mean terminal function value is of the order 3.22×10^{-8}. Clearly the proposed method improves the error function value with high precision. This is achieved with relatively less number of function evaluations and number of epoch. Figure 5.7 and 5.8 compare the function evaluations and epoch with different training methods.

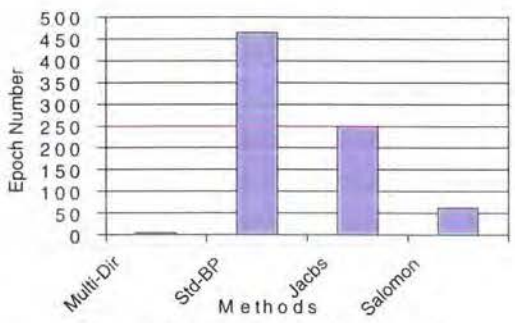


Figure 5.7 Epoch comparison

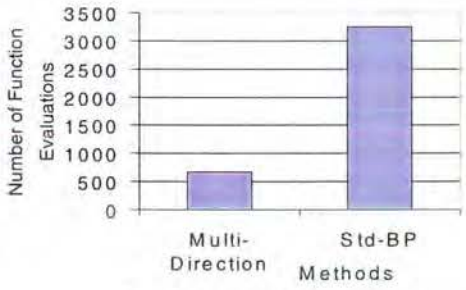


Figure 5.8 Total number of function evaluations

5.6 Discussions

In practice, the Multi-directional search methods without derivative information is reduced into a constrained one-dimensional problem and the search is continued in the direction j varying the learning rate parameter η_j with the given value of w_j and d_j. The error function is sampled in the weight parameter space with an order of magnitude $\eta_j d_j$. The network weight is increased or decreased depending on the shape of the hyper surface. The sampled function values are used to locate the minimum points according to an interpolation search method shown in Chapter 4, Section 4.3. The search is repeated to achieve high precision.

The rectilinear search strategy as discussed in Section 5.2 always converges according to the proposition 5.2 under the hypothesis if $f(w_k)$ has partial derivatives and when interpolation search is performed. Rapid convergence is achieved if the contour surface of

$f(w_k)$ is approximately concentric or if the principal axes of the elliptic contours coincide with the coordinate axes (Schwefel, 1981). When the numbers of ANN weights are large, they influence each other during convergence. This causes the interpolation search to move a small distance and it affects the convergence. The method needs small storage of the order m. It is difficult to compare the convergence rate of this algorithm with the standard back propagation algorithm such as suggested by Rumelhart et al., 1986. The algorithm belongs to a different class and does not require gradient information of the function.

The multi-directional search can be reduced to coordinate search and hence the convergence of the algorithm approximates to $\frac{|f(w_{k+1})-f(w^*)|}{|f(w_k)-f(w^*)|} \leq \left(1-\frac{a}{A(m-1)}\right)^{m-1}$ (Luenberger, 1984). The quantity, (A,a) are the largest and smallest eignevalues of the Hessian matrix with dimension $m \times m$ and $f(w^*)$ is the local minimum. The bound in multi-directional search algorithm depends on the largest and smallest eigenvalue of the Hessian matrix. In fact, the convergence is damped by a factor $(m-1)$. The momentum search makes the method effective and efficient. The specialized interpolation search also improves the convergence. If the Hessian matrix of the ANN error function is nearly diagonal, the steepest descent back propagation would suffer convergence due to stiff ridges and under such circumstances the proposed multi-directional search method tend to improve performance.

It is also noticed that the multi-directional training algorithm is not much sensitive to the initial starting weights in comparison with the gradient-based methods. Therefore, it can locate a relative local minimum from anywhere of the error surface. This is certainly an important property of the developed training method. In first order and second order training method, however, the initial starting vector affects convergence. Further analyses are reported in Chapter 7.

Chapter

6

Restart Training Algorithm

6.1 Introduction

This chapter develops a new derivative free simplex restart training method, which improves the geometry of the degenerate simplex by a restart and re-scale operation. The new restart search is forced to continue into parameter space so that the local search is improved (Ahmed, 1999a; 1999b). Classification and statistical time series problems are solved using this method. The improved search method solves problems in higher dimensions. It is an unconstrained non-linear training method and does not require the derivative information of the error function.

Briefly, the properties of the simplex method are discussed in Section 6.2 and 6.3. The improved method is presented in Section 6.4. Section 6.5 compares the simplex method of Nelder and Mead (1965) with some test problems and Section 6.6 presents some experimental results with XOR problems. Finally Section 6.7 provides some discussions.

6.2 Background of Restart Training

Given a collection of points in Euclidean space, a pattern is formed connecting all the points. For example, a tetrahedron is formed in E^3 and a triangle in E^2. This pattern or geometry is called simplex. It must always enclose finite volume in m dimensional space. To reduce the number of simultaneous trials in the experimental identification procedure of factorial design, the minimum number of starting points is suggested as $m+1$ in E^m (Spendley, Hext and Himsworth, 1962). Therefore, in simplex $m+1$ solutions are maintained to define enough vertices of a polytope surrounding a point. It starts with an initial simplex and by repeatedly replacing its vertices with reflection points; the method generates lower function values. Nelder and Mead (1965) improved the simplex method by maintaining irregular simplex. It repeatedly reflects a vertex along the centroid such that the function values at the vertices satisfy some form of descent condition with reference to the previous simplex. In the following sections a new search with simplex, which confines finite volume, is developed. The proposed method prevents the simplex to stagnate during search so that the training continues into parameter space.

6.2.1 Derivative Free Simplex Search

The simplex optimization method is multi-directional search and belongs to the derivative free class. It is found to be robust for problems with discontinuity (McKinnon, 1998) or where the function values are noisy (McKinnon, 1998). The multi-directional search method (Dennis and Torczon, 1991; Torczon, 1989) maintain uniform linear independence of the simplex edges and require only simple decrease in the function value at each iteration (Lagarias et al., 1998). Recently, Kelley (1999) propose a simplex search method for sufficient decrease which, guarantee convergence of the Nelder and Mead (1965) iteration to a stationary point, if the objective function is smooth and the diameters of the Nelder and Mead (1965) simplex converges to zero. He proposes a new step, which re-initializes the simplex to a smaller size with orthogonal edge whose orientation is determined by an approximate descent direction from the current best point. Results in higher dimensions are not reported in any of these studies.

The computational operation of the simplex search is of the order $O(m^2)$ and according to Nelder and Mead (1965); the number of function calls increases approximately as $O(m^{2.11})$ based on results up to ten variables (Schwefel, 1981). The method is efficient in finding better function value even though it faces difficulty with large size problem. Lagarias et al. (1998) indicate the following reasons for its popularity:

a. In many applications, for example in industrial process control, the interest is to find parameter values that improve some performance measures. The method improves significantly the function value in early stage of iteration;
b. In some applications, function evaluation is expensive or time consuming and the derivative cannot be calculated. The Nelder and Mead (1965) method when succeeds, tend to require fewer function evaluations;
c. The search is simple, derivative free and robust.

The method is reliable when working with few variables (Lagarias et al., 1998; McKinnon, 1998; and Tseng, 1995). McKinnon (1998) observes that the Nelder and Mead (1965) method fails when the search direction defined by the method becomes orthogonal to the gradient directions. McKinnon (1998) presents a family of functions of two variables, which cause the Nelder and Mead (1965) method to converge to a non-stationary point. These examples show that the method performs inside contraction step repeatedly with the best vertex remaining fixed. These results demonstrate the need to improve the Nelder and Mead (1965) method, which in general have convergence properties (McKinnon, 1998).

An improved simplex training algorithm that solves problems containing 36 variables is developed. The proposed algorithm performs a re-scale operation when the simplex degenerates. It defines a favorable simplex using the vertex where the function value is

minimum. The simplex then performs a restart phase to change the direction along which the simplex initiates a new search. The proposed method converges earlier than the Nelder and Mead (1965) method and improves the error function value. The numbers of function evaluations are reduced significantly.

6.3 Simplex Method

Simplex in E^m consists of $m+1$ points. All convex combination of these points does not rest on hyper plane. Consider the problem of minimizing an ANN error function $f(w_j)$, where $w = (w_1, w_2, ..., w_m)$ is the vector containing ANN connection weights as variables. The variables w_j are stored in a vertex of the simplex. Let v^1 be an initial estimate of the simplex at vertex 1, which is set initially to an arbitrary value stored in w_j. The other vertices of the simplex are formed according to the Equation (6.1):

$$v^{t+1} = v^t (1 + \lambda^t) d^t . \tag{6.1}$$

The values of $\lambda^t d^t$ should be such that the quantities $\delta^t = | f(v^t + \lambda^t d^t) - f(v^t) |$ are different. The index t is defined as $t = 1,2,.....,(m+1)$.

All the vertices $(v^1, v^2, ..., v^{m+1})$ of the initial simplex are, therefore, defined. The error function value is evaluated using all the vertices to determine the *lowest, highest and an intermediate or second highest* function value. The corresponding vertex positions are marked *l*, *h*, *s* and these pointers are determined in the following expressions:

$$l = \arg\, [\min_t \{ f(v^t) \}] \tag{6.2}$$

$$h = \arg\, [\max_t \{ f(v^t) \}] \tag{6.3}$$

$$s = \arg [\max_t \{ f(v^t) \} \mid t \neq h]] \geq f(v^l) . \tag{6.4}$$

The average value of the simplex vertices corresponds to the centroid which is a midway point given by:

$$\textbf{\textit{Centroid:}}\ c = \frac{1}{m} \sum_{\substack{i=1 \\ i \neq h}}^{m+1} (v^t) \quad . \tag{6.5}$$

An iteration of the algorithm corresponds to the evaluation of the Equation 6.5. The simplex search replaces the vertex with the highest error function value by a new vertex

situated at a *reflection point* along the midway of other m vertices. Figure 6.1 shows the interpretation of this strategy in two dimensions. This principle locates the new vertex at a minimum point. The newest vertex is reflected according to the Equation 6.6 to explore other minimum in the neighborhood.

Reflection: $r = c + \alpha (c - v^h)$. (6.6)

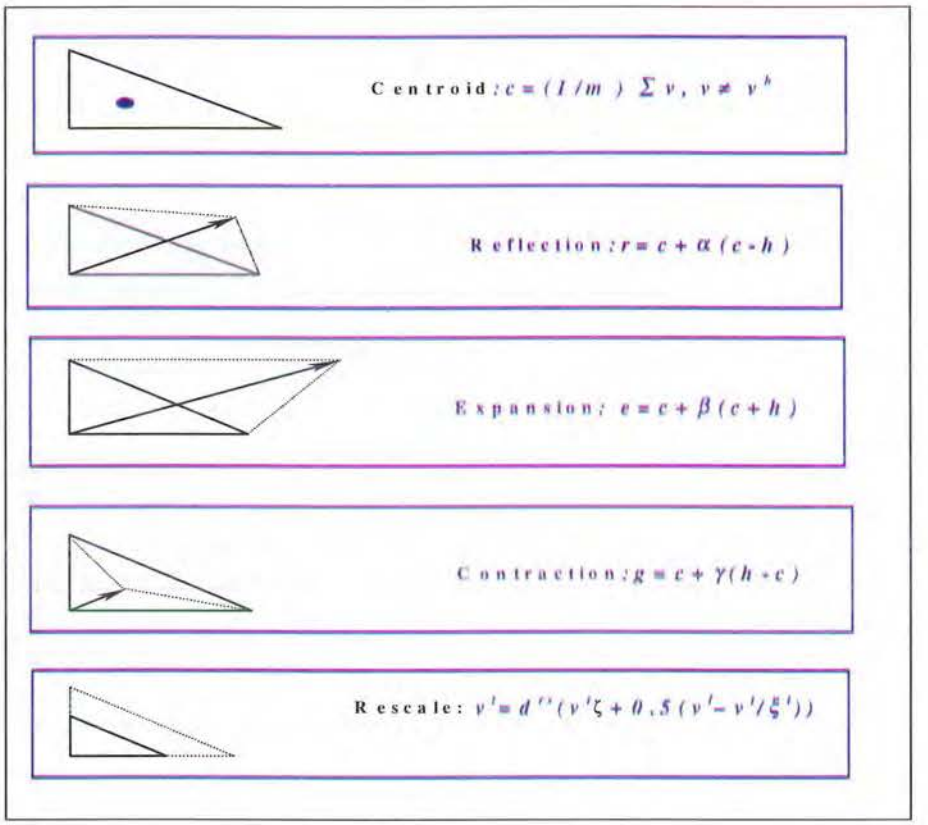


Figure 6.1 Simplex formation strategy in two dimensions

The refection point r is the line joining v^h and c on the extreme side of c form v^h. The function value $f(r)$ is evaluated and three main strategies are followed to generate a new exploration point. They are defined as *expansion, contraction and re-scale* and are shown in Figure 6.1 in E^2. The decision to follow a specific strategy is based on the function value $f(r)$ with reference to four search intervals. These intervals are defined below through the expressions 6.7, 6.8, 6.9 and 6.10:

Interval 1: $f(r) \leq f(v^l)$ (6.7)

Interval 2: $f(v^l) < f(r) \leq \max_{1 \leq l \leq m+1} \{f(v^l); l \neq h\}$ (6.8)

Interval 3: $\max_{1 \leq l \leq m+1} \{f(v^l); l \neq h\} < f(r) \leq v^h$ (6.9)

Interval 4: $f(v^h) < f(r)$. (6.10)

If $f(r)$ is in interval 1, an expansion of the simplex according to the expression 6.11 is recommended:

Expansion: $e \leftarrow c + \beta (r - c)$. (6.11)

If $f(e)$ is in interval 1, the simplex search replaces v^h with e, otherwise v^h replaces r. When the function value $f(r)$ due to reflection falls in interval 2, it is only recommended to replace v^h with r. A contraction step is followed if $f(r)$ is in interval 3 and the simplex with highest function value v^h is replaced with r. The simplex is then contracted according to the Equation 6.12:

Contraction: $g \leftarrow c + \gamma (v^h - c)$. (6.12)

If the function value $f(g)$ is in interval 1, 2 or 3, the vertex with highest function value v^h is replaced by e. If $f(g)$ is in interval 4, Nelder and Mead (1965) at this stage scale or reduce the current simplex with reference to the low vertex of the simplex given by:

$$v^l = \left(v^l + (v^l - v^l)/2\right). \tag{6.13}$$

The evaluation of this step in Equation 6.13 is defined as restart effort. Algorithm is said to have converged when the relative improvement in function value is insignificant. The standard deviation, σ, of the function value along all the vertices is calculated. The algorithm is terminated if the standard deviation is significantly small. To terminate the algorithm, the following termination criterion is used:

Convergence: If $\sigma = \sqrt{\dfrac{\sum_{l=1}^{m+1} \left(f(v^l) - \bar{f}\right)^2}{m}} < \varepsilon$. (6.14)

where, $\bar{f} = \dfrac{\sum_{l=1}^{m+1} f(v^l)}{m+1}$ and set $v^* \leftarrow v^l$ as the best point found so far.

The performance of the search method is measured in total number of function evaluations needed by the algorithm. The Nelder and Mead (1965) simplex search algorithm is shown in Figure 6.2.

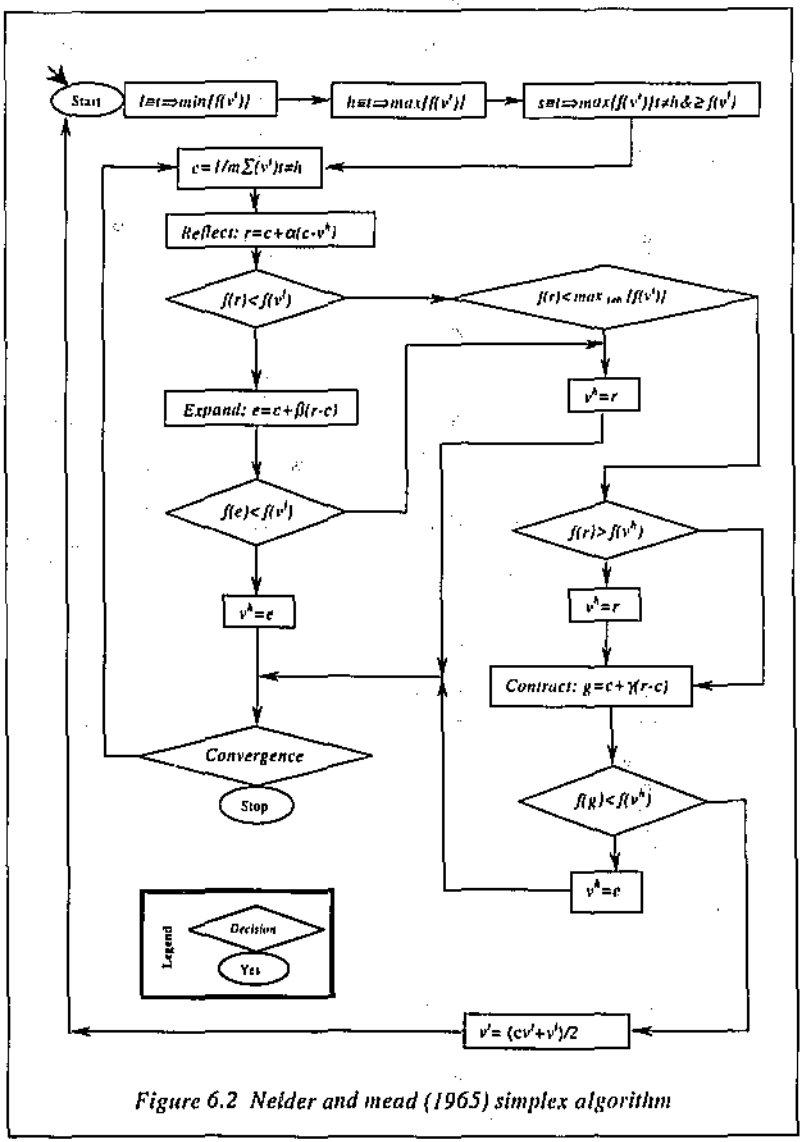


Figure 6.2 Nelder and mead (1965) simplex algorithm

6.4 Improved Restart Training Method

The modified method is shown in Figure 6.3. It replaces r by v^h if the condition $f(v^l) \le f(r) < f(v^h)$ is true and another reflection is attempted. If the reflection is highly successful and generates a new minimum such that $f(r) < f(v^l)$, then the direction chosen is a good direction and expansion is followed. Based on the magnitude of $f(r)$ and $f(e)$ either r or e replaces v^l. A new centroid is again is calculated and reflection is attempted. However, if the reflection is not successful and the condition $f(r) \ge f(v^h)$ is true, then the search is taking place in wrong direction and a contraction is performed. The search begins with expansion, if the contraction is successful and $f(g) < f(v^l)$. In case, we have $f(g) \ge f(v^h)$ a *re-scale* is attempted and the direction of simplex search is changed with the user defined search vector d^{rs} to *restart* the search. The *re-scale* phase constructs simplex according to the Equation 6.15:

$$v^i = d^{rs}\left(v^i \zeta + (v^i - \frac{v^i}{\zeta^n})\right). \tag{6.15}$$

The direction vector d^{rs} is changed to generate new centroid when a particular direction is not favorable. The setting of this vector is shown in Section 6.4.2. The search begins from initial stage while retaining the best function value found so far. This step is defined as *restart* phase. The reflection is attempted when $f(r) < f(v^h)$ and another reflection is suggested when $f(r) > f(v^s)$. In this case v^h is adjusted with reflection point. A new centroid is calculated when $f(r) \le f(v^s)$ and the index h is replace by s. These steps are taken since the repeated reflection is successful with these adjustments.

The edge length of the polyhedron is changed to continue new search. These changes allow the simplex to span in search space. At the end of the routine evaluation, simplex size is changed through the *re-scale* scheme and a new search direction is initialized to follow the *restart* phase. The search begins with a simplex of different edge lengths so that the *reflections*, *expansions* or *contractions* steps improve the descent direction. This step is suggested to maintain non-degenerate simplex. The non-degeneracy of the initial simplex implies non-degeneracy of all other subsequent simplex. Recently Lagarias et al. (1998) proved this assumption. If a non-contraction phase occurs, the trial points replace the worst vertex. If a contraction phase occurs, Equation 6.15 replaces the current simplex such that the geometry of the new simplex is non-degenerate. This is also equivalent to starting search with different starting points in parameter space and a refined local minimum can be discovered by this method. According to Lagarias et al. (1998) the reflection, expansion or contraction step produce a convergent sequence in $\{f(v_k^l)\}$ when the search starts with a non-degenerate simplex. The notation k is embedded in $f(v_k^l)$ to

represent the lowest error function value at iteration k. It is due to this property the *restart* phase is reduced considerably in higher dimensions. The algorithm is shown in Figure 6.3 and the pseudo program is shown in Table 6.1. The Nelder and Mead method require one function evaluation when it terminates in reflection step. It requires two function evaluations when it terminates in expansion or contraction step. There are $m+2$ function evaluations, if a contraction step occurs (Lagarias et al., 1998). The improved method performs a *resale* phase instead of contraction step. The complexity of the restart with *resale* is the same. Therefore, no additional function evaluations are needed. The experimental results are discussed in Section 6.5.

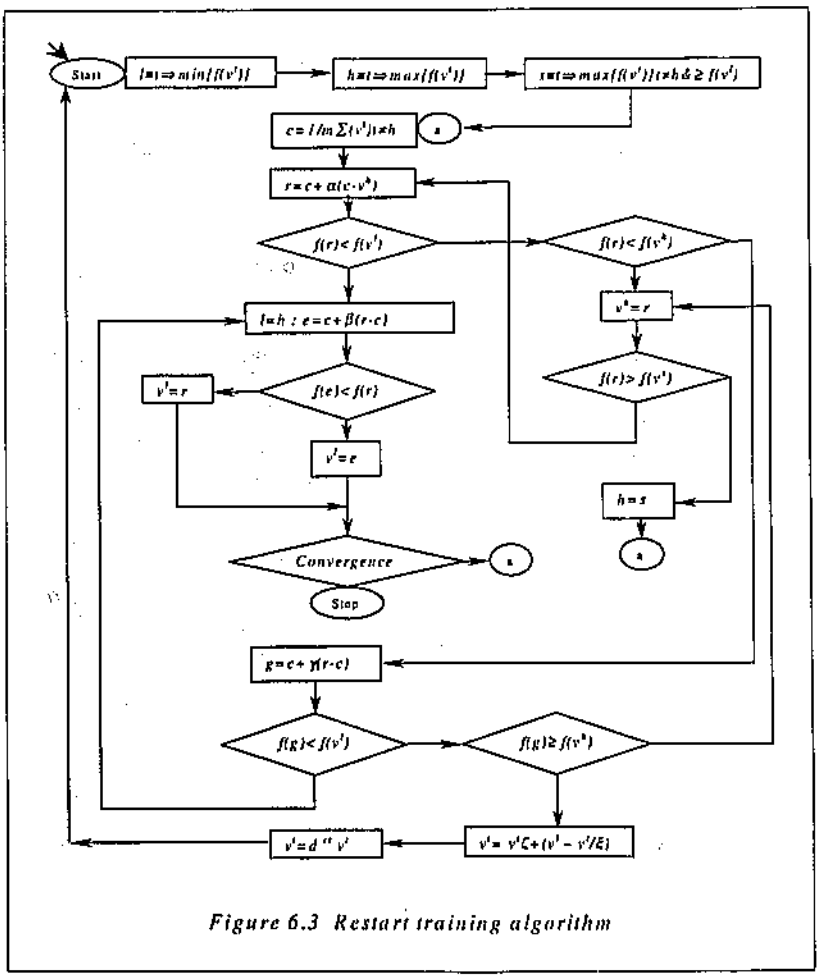


Figure 6.3 Restart training algorithm

Initialization:

1. *1.1 Construct a simplex in* E^{m+1} *choosing the starting points on vertex* $(v^1, v^2, \ldots\ldots, v^{m+1})$.

   1.2 The first and second layer weights are redefined as $w_{(m-1)l+i} \leftarrow w^1_{in}, \forall i,n$ *and* $w_{sa+(a-1)S+n} \leftarrow w^2_{na}, \forall n,a$.

   1.3 Vertex number 1 initializes the initial simplex for ANN error function, $v^{t+1} \equiv \leftarrow (w_1, w_2, \ldots\ldots, w_m)$.

   1.4 The other points of the simplex vertices are generated by appropriately selecting λ^t *and* d^t *defined by :* $v^{t+1} = v^t(1+\lambda^t)d^t$, $1 \le t \le m+1$, *and* $v^1 \equiv (v^1, v^2, \ldots\ldots, v^{m+1})$ *and execute main step 2*

Main Step:

2. *2.1 let,* $v^l, v^h \in \{v^1, v^2, \ldots\ldots, v^{m+1}\}$ *be such that* $l \Rightarrow f(v^l) = \min_{1 \le t \le m+1} f(v^t)$

   2.2 $h \Rightarrow f(v^h) = \max_{1 \le t \le m+1} f(v^t)$ *, and perform Step 3*

3. 3.1 *Let* $v^s \in \{v^1, v^2, \ldots\ldots, v^{m+1}\}$ *be such that* $s \Rightarrow f(v^s) = \max_{1 \le t \le m+1} \{f(v^t) : t \ne h\} \ge f(v^t)$

   3.2 *Centroid: let,* $c \leftarrow \frac{1}{m} \sum_{\substack{t=1 \\ t \ne h}}^{m+1} (v^t)$ *, set* $k \leftarrow k+1$ *and perform Step 4*

4. *4.1 Reflection:* $r \leftarrow c + \alpha\ (c - v^h)$

   4.2 If $f(r) < f(v^l)$ *perform step 6*

   4.3 If $f(r) \ge f(v^h)$ *perform step 8*

   4.4 Otherwise perform step 5

5. *5.1 Set* $v^h \leftarrow r$ *and* $f(v^h) \leftarrow f(r)$

   5.2 If $f(r) > f(v^s)$ *perform step 4*

   5.3 Otherwise, $h \leftarrow s$ *and perform step 3*

6. *6.1 Expansion: Set* $l \leftarrow h$ *and* $e \leftarrow c + \beta\ (r - c)$

   6.2 if $f(e) < f(r)$ *set* $v^l \leftarrow e$ *and* $f(v^l) \leftarrow f(e)$, *and perform step 7*

   6.3 Otherwise, set $v^l \leftarrow r$, $f(v^l) \leftarrow f(r)$ *and perform step 7*

7. *7.1 Convergence: If* $\sigma = \sqrt{\frac{\sum_{i=1}^{m+1} \left(f(v^t) - \bar{f}\right)^2}{m}} < \varphi$ *, where,* $\bar{f} = \frac{\sum_{i=1}^{m+1} f(v^t)}{m+1}$, *Report* $w^* \leftarrow v^l$ *as the minimum point at* k^{th} *iteration and stop.*

   7.2 Otherwise perform step 3

8.	*8.1 Restart:* $g \leftarrow c + \gamma\ (v^h - c\ )$ *8.2 If* $f(g\ ) < f(v^l)$ *perform step 6* *8.3 If* $f(g\ ) < f(v^h)$ *perform step 5, Otherwise perform step 9*
9.	*9.1 Re-scale:* $v^i \leftarrow d^m\left(v^l\zeta + (v^i - \frac{v^l}{\xi^l})\right)$, *and perform step 2*

Table 6.1 Restart training method

6.4.1 Parameter Selection

Initially the factor ς and $\xi^t, t = 1,2,....,n+1$ is set to one. When the simplex degenerates, this factor is set to 3 and the current simplex is modified. At the end of simplex *re-scale* phase the factor is set to one again and attempts a *restart* phase. The direction vector d^m is initially set to the unit vector $[1,1,...,1]^T$. If the attempted *re-scale* does not improve function value, the direction vector is changed to $[-1,-1,.....,-1]^T$ and the parameter ξ^t is set to a value 0.856. It can be any other user defined variable value based on some experiments. This can be tried when a constant value is not appropriate for an application. The restart vector d^m is again reset to vector $[1,1,...,1]^T$ after the *re-scale* phase.

The parameters, α, β and γ influence the convergence. One approach to determine these parameters is to do a full line search according to the method given in Section 4.3 in Chapter 4. This is a crude method. It does not necessarily converge to minimum. Consequently, it will result in large number of function evaluations. Alternately, fixing the values of α, β and γ the directions of search are generated by the simplex from its centroid using the reflection, expansion and contraction steps. The appropriate values of α, β and γ influence the simplex search. These values are fixed based on few experiments. Several starting points are also necessary to arrive at good estimate of the threshold parameter values. These parameters, α, β and γ, are experimentally determined in such a way that the simplex search reaches the minimum trajectory without putting extra burden on function evaluation. The experiments suggest that the values of α, β and γ; are 0.7675, 1.8755 and 0.4615 respectively.

6.5 Experimental Set up

To test the performance of the proposed algorithm in Table 6.1 against the Nelder and Mead (1965) algorithm an ANN seasonal time series problem and the Rosenbrock function (Al-Sultan et al., 1997) are evaluated. The performance of the algorithm against standard back propagation algorithm with XOR problem is also evaluated.

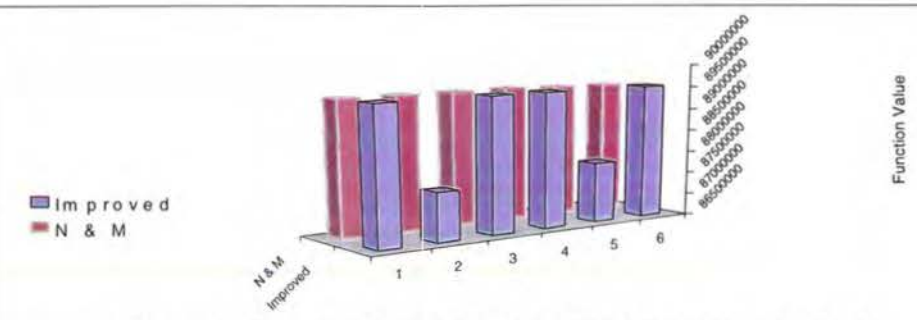


Figure 6.4 Function Value with Improved Algorithm and Nelder & Mead's Method (5-5-1: 30 variables)

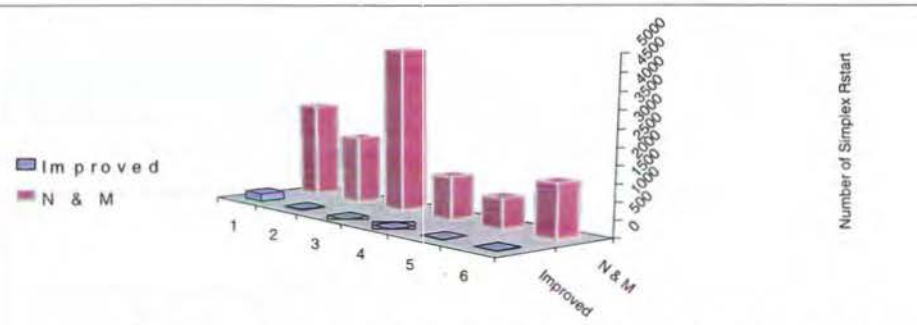


Figure 6.5 Number of Restart with Improved and Nelder & Mead's Method (5-5-1: 30 variables)

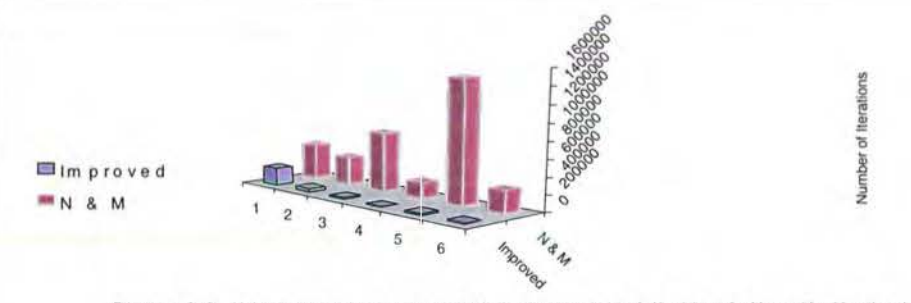


Figure 6.6 Number of Iterations with Improved and Nelder & Mead's Method (5-5-1: 30 Variables)

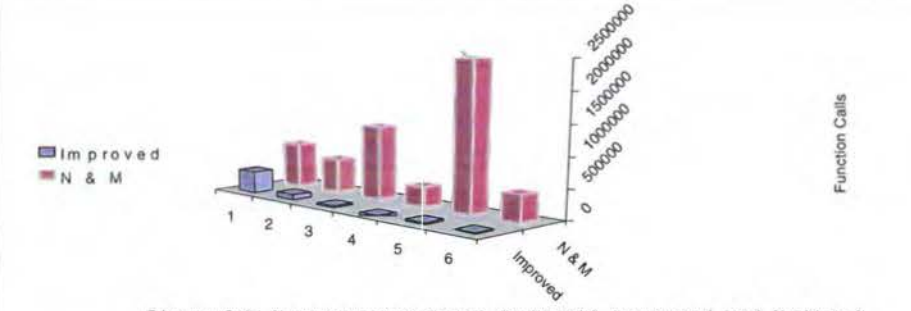


Figure 6.7 Number of Function Calls with Improved and Nelder & Mead's Method (5-5-1: 30 variables)

	Suggested Method				5-6-1 ANN	Nelder and Mead (1965) method			
Experiment number	Function Value SE	No. of Restart	No. of Iterations	No. of Function Evaluations	Starting Points (w_j , j=1,2,....,m)	Function Value SE	No. of Restart	No. of Iterations	No. of Function Evaluations
1.	(^{5})91803810	25	48363	108203	200,100,900,700,800,331,342,310,312,212,421,422,413,344,435,55,54,53, 52,54,4323,7463,6358,6252,9348,67,68,69,60,61,1,2,2,1,1,1	91884980(**)	16812	3440385	5603406
2.	(**)91252060	11	25823	51592	200,100,900,700,800,331,342,310,312,212,421,422,413,344,435,550,541, 532,523,544,4323,7463,6358,6252,9348,67,68,69,60,61,1,2,2,1,1,.5	92038620	4561	879069	1449916
3.	92044390	12	28698	55893	200,100,900,700,800,331,342,310,312,212,421,422,413,344,435,550,541, 532,523,544,4323,7463,6358,6252,9348,675,686,697,608,619,1,2,2,1,1,1	(^{5})92040880	4201	823881	1349421
4.	(^{5})91868300	12	22186	49193	200,100,900,700,800,331,342,310,312,212,421,422,413,344,435,550,541, 532,523,544,432,746,635,625,934,675,686,697,608,619,1,2,2,1,1,.1	92021810(**)	7495	1352406	2275498
5.	92040180	29	72889	135693	8200,8100,8900,8700,8800,331,342,310,312,212,421,422,413,344,435.55 0,541,532,523,544,432,746,635,625,934,675,686,697,608,619,1,2,2,1,1,1	(^{5})91872210	6178	1309033	2109075
6.	92044320	87	95924	247075	8200,8100,8900,8700,8800,331,342,310,312,212,421,422,413,344,435.55 0,541,532,523,544,4323,7463,6358,6252,9348,675,686,697,608,619,1,2,2, 1,1,.4	(^{5})92039900(**)	1732	376366	602270

Table 6.2.a. Performance of algorithm using 5-6-1 ANN configuration with 36 variables

	Suggested Method				5-5-1 ANN	Nelder and Mead (1965) method			
Experiment number	Function Value SE	No. of Restart	No. of Iterations	No. of Function Evaluations	Starting Points (w_j , j=1,2,....,m)	Function Value SE	No. of Restart	No. of Iterations	No. of Function Evaluations
1.	(^{5})89524570	235	211691	362493	12,11,14,13,15,31,31,31,31,31,0.4,0.4,0.4,0.4,0.4,5,5,5,5,5,10,13,12, 11,51,1,2,1,3,1	89526290	2833	444885	745022
2.	(**)87587950	28	39941	88288	12,11,14,13,15,31,31,31,31,31,4,4,4,4,4,5,5,5,5,5,10,13,12,11,51,1,2 ,1,3,1	89517880	2013	355715	580580
3.	89526770	3	20570	29050	1,2,1,1,1,4,1,3,1,5,33,34,31,31,21,4,4,4,4,4,5,5,5,5,4,0.10,0.13,0.12, 0.11,0.51,1,2,1,3,1	(^{5})89524440(**)	4796	723516	1234446
4.	89526740	1	4636	5993	0.12,0.11,0.14,0.13,0.15,33,34,31,31,21,4,4,4,4,3,5,5,5,5,4,10,13,12, 11,51,1,2,1,3,1	(^{5})89525200	1279	205503	342882
5.	(^{5})87855250	17	19875	33326	834,100,900,700,800,4330,343,3515,316,212,4821,4283,4184,3453, 4345,645,545,545,765,765,323,403,6358,524,9348,32,22,61,93,12	89495270	907	1467849	2454559
6.	89525370	6	13224	18430	1,1,1,1,1,3,3,3,3,2,4,4,4,4,3,5,5,5,5,4,1,1,1,1,5,2,2,2,2,2	(^{5})89486860	1574	280637	457517

Table 6.2.b. Performance of algorithm using 5-5-1 ANN configuration with 30 variables

6.5.1 Seasonal Time Series Problem

Two test functions are constructed using a three-layer feed forward artificial neural network with 5-6-1 and 5-5-1 configurations. There are 36 and 30 variables with these two corresponding error functions. The time series data from September 1976 to Jun 1996 is used to test the performance of the algorithms, which use the same starting weights. The two methods are compared with the numbers of function evaluations, number of restart and number of iterations. Six sets of experiments are performed with 5-6-1 and 5-5-1 ANN configurations (Ahmed, 1999b). The termination criterion is set to 10^{-10}. The starting weight vectors have low as well as high magnitudes.

The restart phase is related with contraction phase in Nelder and Mead method (1965). Table 6.2.a lists the results of the improved restart and the Nelder and Mead (1965) method using 5-6-1 ANN configuration. Table 6.2.b lists the similar results for the 5-5-1 ANN configuration. The algorithm when fails to terminate is marked with asterisk (*) sign and forced to terminate if 20,000 total restarts do not improve function value. The convergence difficulty occurs due to extremely small and almost equal edge lengths of the simplex. The method, which identifies better function value, is marked with dollar ($) sign and the best function value for the entire set of experiment is marked with percent (%) sign. Figure 6.4 through 6.7 show the performance of the improved and Nelder and Mead (1965) algorithms in function value, number of restart, number of iterations and number of function calls.

The suggested method improves function value (see % sign in Table 6.2.a and 6.2.b) with 5-5-1 and 5-6-1 ANN configurations against the Nelder and Mead (1965) method in entire set of experiment. The proposed algorithm shows improvement in restart efforts both in 5-5-1 and 5-6-1 ANN problems as compared to Nelder and Mead (1965) method. The number of iterations and function evaluations are less with the proposed method. In 5-5-1 ANN configuration, the Nelder and Mead (1965) method suffers convergence in three experiments with 5-6-1 configuration. The minimum and maximum numbers of function evaluations are 49,193 and 247,075 respectively with the improved method in 5-6-1 ANN configuration. The corresponding numbers of function evaluations in Nelder and Mead (1965) method are 602,270 and 5,603,406 respectively. The minimum function value identified by the improved method is 91,252,060, while with the Nelder and Mead (1965) method it is 91,872,210. The algorithms converge to different points due to excessive number of local minimum in ANN error function (Partridge, 1997). Figure 6.5 compares the number of restart efforts, which is less with the proposed method. In lower dimensions this phenomena, however, is insignificant according to Lagaris et al. (1998), Kelly (1999) and Torczon (1989) for strictly convex functions.

The restart method terminates with improved function value, which is 87,587,950 in 5-5-1 ANN configuration. It is marked with percentage (%) sign. In some experiment the Nelder

and Mead (1965) method finds better function value. This is observed in Table 6.2.a and 6.2.b, marked with dollar ($) sign in experiments with the same starting points.

6.5.2 Comparison with Rosenbrock Function

To compare the performance of the method with reported results in Al-Sultan et al. (1997) and Corana et al. (1987), we list the results in Table 6.3 on Rosenbrock function as shown in Equation 6.22 in four dimensions:

$$f(x_1, x_2, x_3, x_4) = \sum_{j=1}^{3} 100(x_{j+1} - x_j^2)^2 - (1 - x_j)^2 . \qquad (6.22)$$

The proposed method show improved converges in 7/10 experiments with less number of function evaluations. This can be observed in column 3 and 4 of Table 6.3. The experiment number 1, 2 and 3 with the specified starting points do not show significant improvements but converge to an acceptable limit of function value. The experiments in all remaining experiments, 4 through 10 show significant improvements in function evaluations. The Nelder and Mead (1965) method on the other hand faces convergence difficulty in experiment number 1 and 8. This demonstrates the efficiency of the proposed method with respect to number of efforts needed to converge to a solution.

Experiment No.	Starting Points (Al-Sultan et al., 1997)	Number of Function Evaluation (Al-Sultan et al., 1997)		Function Value (Al-Sultan et al., 1997)	
		Nelder & Mead (1965) Method	Proposed Method	Nelder & Mead (1965)	Proposed Method
1	101,101,101,101	1869	1914	3.7	3.42E-6
2	101,101,101,-99	784	1929	5.46E-17	4.97E-6
3	101,101,-99,-99	937	1922	9.8E-18	2.31E-9
4	101,-99,-99,-99	1079	723	3.4E-17	1.43E-8
5	-99,-99,-99,-99	859	400	8.3E-18	2.68E-9
6	-99,101,-99,101	967	913	1.2E-17	6.32E-8
7	101,-99,101,-99	870	723	6.0E-18	1.54E-8
8	201,0,0,0	1419	163	3.7	7.65E-7
9	1,201,1,1	1077	320	9.4-E-18	2.58E-10
10	1,1,1,201	1265	27	3.9E-17	3.81E-8

Table 6.3. Performance of the Algorithm With Rosenbrock Function in 4 Dimensions

The average number of function evaluations with the restart method is 903.4. Results reported in Al-Sultan et al. (1997) and Corna et al. (1987) are reproduced in Table 6.3 and it suggests that the average numbers of function evaluations are 1112.8 with the Nelder and Mead (1965) method. The stating vectors are the same as shown in Table 6.3. The maximum and minimum number of iterations are 1929 and 27 respectively with the

proposed method, while the corresponding values with the Nelder and Mead (1965) method are 1869 and 784 respectively.

6.5.3 Discussions on the Performance of the Restart Algorithm

Two ANN model with 5-6-1 and 5-5-1 configurations, which model seasonal time series with the first 84 data points in Table A.5 as listed in Appendix A are trained with Nelder and Mead (1965) and the improved algorithm. The equivalent numbers of variables are thirty-six and thirty respectively. In a six set of experiments, the proposed method converges with less number of function evaluations, restart attempts and number of iterations. The experiment with 36 variables show that the improved method evaluates functions 51,592 numbers of times and converges to a local minimum. The corresponding number of function evaluation is 2,109,075 with the Nelder and Mead (1965) method. The minimum and maximum numbers of restart attempts are 11 and 87 respectively with the proposed method in 5-6-1 ANN configuration. The corresponding numbers are 4201 and 6178 with the Nelder and Mead method. To find improved function value in all six set of experiment with 30 variables, the proposed method needs 88,288 number of function evaluations and it is 457,517 with the Nelder and Mead method. The quality of solution in Nelder and Mead method is not as good as with the proposed method. The Nelder and Mead method face convergence difficulty as the number of variable increases. However, it identifies better function value in some experiments. This study demonstrates that the improved method can solve problems in higher dimensions.

6.6 Sample Calculations with 2-2-1 ANN XOR Problem

The Figure 6.8 shows a trained XOR 2-2-1 ANN configuration. The training initiates with a starting vector $(1, 5, 3, 1, 1, 2)^T$ and the sample calculations are shown in Table 6.4. When there is zero input to the hidden layer neurons, the input to the hidden layer neuron is approximated to the value $1x10^{-5}$ for computational convenience.

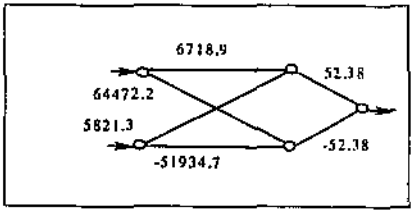


Figure 6.8 A trained 2-2-1 ANN XOR

The sample calculations are shown next in the Table 6.4.

$x_1^{p=1} = 0$, $x_2^{p=1} = 0$, $h_1^{p=1} = (6718.9)*0 + (5821.28)*0 = 0$, $h_2^{p=1} = (64472.2)*0 + (-51934.73)*0 = 0$

$g_1^{p=1} = \frac{1}{1+|\inf|0^{-}|} = -0.078$, $g_2^{p=1} = \frac{1}{1+|\inf|0^{-}|} = -0.078$,

$z_1^{p=1} = (52.38)*(-0.078) + (-52.38)*(-0.078) \cong 0.0$

$x_1^{p=2} = 1$, $x_2^{p=2} = 0$,

$h_1^{p=2} = (6718.9)*1 + (5821.28)*0 = 6718.9$, $h_2^{p=2} = (64472.2)*1 + (-51934.7)*0 = 64472.2$

$g_1^{p=2} = \frac{1}{1+|\inf 6718.9|} = 0.102$, $g_2^{p=2} = \frac{1}{1+|\inf 64472.2|} = .083$,

$z_1^{p=2} = (52.38)*(.102) + (-52.38)*(.083) \cong 1.0045$

$x_1^{p=3} = 0$, $x_2^{p=3} = 1$,

$h_1^{p=3} = (6718.9)*0 + (5821.28)*1 = 5821.28$, $h_2^{p=3} = (64472.2)*0 + (-51934.7)*1 = -51934.7$

$g_1^{p=3} = \frac{1}{1+|\inf 5821.28|} = .1034$, $g_2^{p=3} = \frac{1}{1+|\inf -51934.7|} = 0.0843$,

$z_1^{p=3} = (.1034)*(52.38) + (.0843)*(-52.38) \cong 1.0015$

$x_1^{p=4} = 1$, $x_2^{p=4} = 1$,

$h_1^{p=4} = (6718.9)*1 + (5821.28)*1 = 12540.18$, $h_2^{p=4} = (64472.2)*1 + (-51934.73)*1 = 12537.47$

$g_1^{p=4} = \frac{1}{1+|\inf 12540.18|} = .0958$, $g_2^{p=4} = \frac{1}{1+|\inf 12537.47|} = .09582$,

$z_1^{p=4} = (.0958)*(52.38) + (.09582)*(-52.38) \cong 0.0010$

Table 6.4 Sample calculations with 2-2-1 trained ANN XOR

6.6.1 Analysis of the Training Method With XOR Problem

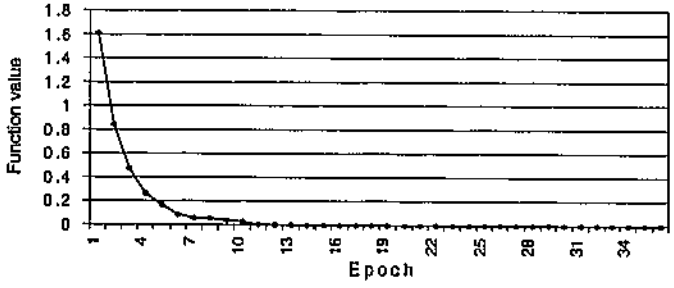


Figure 6.9 Convergence with re-start simplex training

Figure 6.9 shows that error function convergence to almost zero value within 36 epochs. More epochs are taken to achieve higher precision. The input data for the training problem corresponds to the subset of the Table A.1 in Appendix A in rows 1, 2, 3, 4 and column 4, 5. The random starting points $w_j = (w_1, w_2, \ldots\ldots, w_n)$ are taken from the Table A.2 in Appendix A. The proposed training method is self-adaptive and derivative free.

6.6.2 Random Starting Vector in the Wide Range

Table 6.5 shows the training results with random starting weights. The experiment number 8, 9 and 10 failed to converge. The large weighs affect its convergence. The average performance is reported for the cases where the method converged. It takes 268.4 epoch and 649.3 number of function evaluations to train a 2-2-1 XOR ANN. The corresponding figures for the standard back propagation method are 5142.3 and 30,006 respectively. The comparison is biased since the method suffers in three experiments. The experiment will be repeated with random starting weights in small magnitude for detail analysis.

The median performance of the proposed algorithm with epoch size and total function evaluations corresponds to the values 147 and 361, while with the standard back propagation method these counts are 3579 and 25061 respectively.

	The Restart Training Method			Standard Back Propagation Method				
Experiment #	Epoch	Function evaluations	Terminal function value	Epoch	Gradient evaluations	Function evaluations	Total No. of function evaluations	Terminal function value
1	147	361	2.9E-05	22	24	138	162	6.20E-04
2	354	737	3.4E-08	5867	5869	35208	41077	8.20E-04
3	181	416	1.1E-06	29	31	180	211	1.50E-02
4	85	246	3.0E-07	17434	17436	104610	122046	5.10E-03
5	124	300	1.4E-08	4474	4476	26850	31326	2.50E-03
6	132	316	7.0E-09	5737	5739	34428	40167	4.60E-03
7	856	2169	8.6E-02	2137	2139	12828	14967	3.00E-03
8	Failed	Failed	Failed	13024	13026	78150	91176	8.90E-04
9	Failed	Failed	Failed	15	17	116	133	2.90E-03
10	Failed	Failed	Failed	2684	2686	16110	18796	1.00E-02
Mean	268.429	649.286	0.01229	5142.3	5144.3	30861.8	36006.1	0.004543
Median	147	361	3E-07	3579	3581	21480	25061	0.00295
Standard Deviation	273.273	689.26	0.0325	5852.446	5852.446	35112.73	40965.18	0.00461
Range	771	1923	0.086	17419	17419	104494	121913	0.01438
Minimum	85	246	7E-09	15	17	116	133	0.00062
Maximum	856	2169	0.086	17434	17436	104610	122046	0.015

Table 6.5 Comparison with back propagation method (2-2-1 ANN XOR Problem) with wide range of starting points

The maximum and minimum numbers of epoch are 856 and 85 respectively with the proposed method. The corresponding numbers are 17434 and 15 respectively with the standard back propagation method. The maximum and minimum numbers of total function evaluations are 2169 and 246 respectively with the proposed method. The related values are 122046 and 133 respectively with the standard back propagation method. The standard back propagation successfully converges in all the experiments.

The simulation results with the standard back propagation and the proposed method in the small range of weights are compared in the following section.

6.6.3 Comparison with BP Method

Table 6.6 compares the performance of the proposed algorithm with standard back propagation method. The small magnitude random initial weights are used to initiate training. The experimental results suggest that the proposed method converge faster than the standard back propagation method. The relative efficiency of the proposed method over the standard back propagation training in average number of epoch and function evaluations is (463.6/278.2) 1.67 and (3253.2/614.9) 5.3 respectively.

	The Restart Training Method			Standard Back Propagation Method				
Experiment #	Epoch	Function evaluations	Terminal function value	Epoch	Gradient evaluations	Function evaluations	Total No. of function evaluations	Terminal function value
1	147	361	3.00E-05	25	156	27	183	6.18E-04
2	720	1505	8.50E-09	27	168	29	197	8.21E-03
3	181	416	1.10E-06	26	162	28	190	1.50E-02
4	228	474	7.40E-08	24	150	26	176	1.00E-02
5	315	653	9.30E-09	1751	10512	1753	12265	1.30E-04
6	416	868	1.40E-07	30	186	32	218	5.60E-03
7	216	581	3.40E-08	2687	16128	2689	18817	3.95E-03
8	187	436	1.30E-08	20	126	22	148	8.97E-03
9	215	482	6.00E-09	23	144	25	169	1.40E-03
10	157	373	2.40E-08	23	144	25	169	1.80E-03
Mean	278.2	614.9	3.14E-06	463.6	2787.6	465.6	3253.2	0.005568
Median	215.5	478	2.9E-08	25.5	159	27.5	186.5	0.004775
Standard Deviation	174.99	348.17	9.44E-06	951.12	5706.727	951.121	6657.849	0.004892
Range	573	1144	3E-05	2667	16002	2667	18669	0.01487
Minimum	147	361	6E-09	20	126	22	148	0.00013
Maximum	720	1505	0.00003	2687	16128	2689	18817	0.015

Table 6.6 Comparison with back propagation training (2-2-1 ANN XOR Problem) with small range of starting points

The mean terminal function value is $3.14x10^{-6}$ with the proposed method and is a better training performance. The maximum and minimum numbers of epoch are 720 and 147 resulting in the range statistics 573. The corresponding values with the standard back propagation training are 2687 and 20 respectively and the rage statistics is 2667. The total number of maximum and minimum function evaluations are 1505 and 361 respectively and the range statistics is 1144 with the proposed method. The related figures are 18817 and 148 respectively with the standard back propagation method. The range statistics is 18669. The experimental results suggest that the average performance of the restart training method is better than the standard back propagation training method.

The median performance of the standard back propagation is better than the restart training method. The median number of the epoch is 25.5 with the standard back propagation method against 215.5 with the proposed method. Similarly the median value of the total number of function evaluation is 186.5 with the standard back propagation method. The corresponding value is 478 with the proposed method, which however, improves in median performance in terminal function value at $2.9x10^{-8}$ against the median terminal function value $4.77x10^{-3}$ with the standard back propagation method. This implies that the proposed method finds better local minimum.

6.6.4 Comparison with the Results in Literature

Table 6.7 compares the results with the proposed method, standard back propagation method and the methods reported in Jacobs (1988) and Salomon (1996) with the XOR problem. The proposed training method and the standard back propagation method is initiated with small magnitude random starting points. The simulation result for the standard back propagation method is taken from the Chapter 4 in Table 4.8. The function evaluations including gradient evaluations are not given in the reports in literature.

	The Restart Training Method			Standard Back Propagation Method					Jacobs (1988)	Salomon (1996)	P-R (C.G)
Experiment #	Epoch	Function evaluations	Terminal function value	Epoch	Gradient evaluations	Function evaluations	Total Function Evaluations	Terminal function value	Epoch	Epoch	Epoch
Mean	278.2	614.9	3.14E-06	463.8	2787.6	465.6	3253.2	0.005568	250	62	14
Median	215.5	478	2.9E-08	25.5	159	27.5	186.5	0.004775			
Standard Deviation	174.986	348.17	9.44E-06	951.121	5706.727	951.121	26657.849	0.004892	60		
Range	573	1144	3E-05	2667	16002	2667	18669	0.01487			
Minimum	147	361	6E-09	20	126	22	148	0.00013			
Maximum	720	1505	0.00003	2687	16128	2689	16817	0.015			

Table 6.7 Comparison with other method (2-2-1 ANN XOR problem)

The proposed method needs 278.2 epochs on average to train the 2-2-1 ANN XOR problems. The back propagation method due to Salomon (1996) takes 62 epochs, while the delta bar delta method reported in Jacobs (1988) takes 250 epochs to train the XOR problem. The standard back propagation method needs 463.6 numbers of epochs to train the 2-2-1 ANN XOR problems. The simplex restart training method improves over the standard back propagation method in average epoch. It is derivative free and can train an error function which is ill conditioned, discontinuous and contains stiff ridges. The average numbers of function evaluations are 614.9 with the restart training method while with the standard back propagation method it is 3253.2. The proposed method works on function evaluations and needs no gradient information. Kamarthi et al. (1999) report that the Polak and Ribiere conjugate gradient takes 14 epochs to train XOR problem.

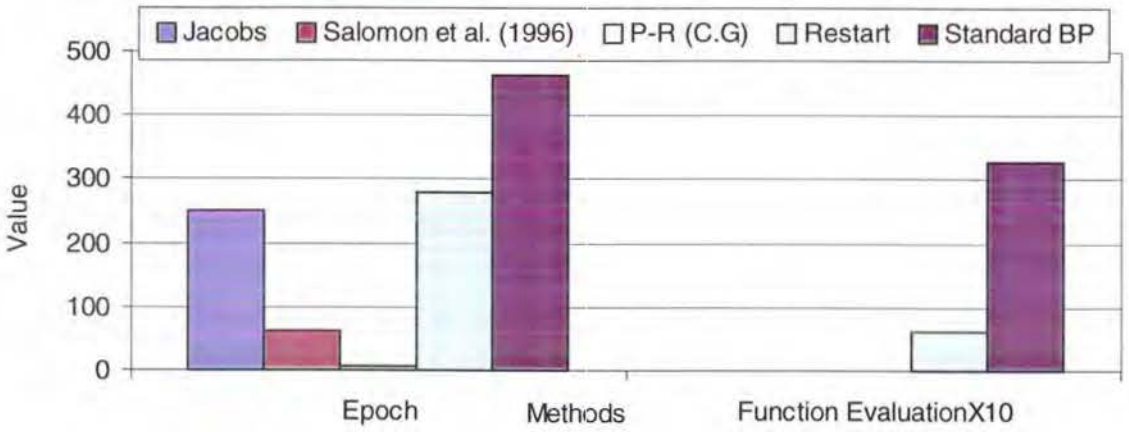


Figure 6.10 Compariosn with different training methods

6.7 Discussions

The *restart* training method improves against the standard back propagation training method in average number of epoch and total number of function evaluations. A factor (463.6 /278.2) 1.67 in number of epoch and (3253.2/614.9) 5.3 in total number of function evaluations indicate the relative improvement. The terminal function value is significantly low with the proposed method, implying a well-trained ANN is found. The restart training algorithm does not need gradient information. It performs on the basis of function evaluations. Ill conditioned and discontinuous error functions can be trained and this is an important characteristic of this training method. Additional performance analyses with the proposed training method are given in Chapter 7.

Chapter
7

Additional Simulations and Other Results

7.1 Introduction

This chapter presents additional experimental results obtained from the three developed training algorithms with 5-5-1 parity, the 9-2-1 L-T letter recognition, the Australian peak electric load forecast with 5-5-1 and Australian Hotel occupancy rate analysis problem with 7-4-1 ANN configuration. The standard back propagation method is used for comparison in convergence analysis. To compare the performance of the last two problems as forecast and multivariate regression model, the standard statistical regression method is used as the benchmark for comparison.

It is noticed earlier that in general the small initial starting points improves the convergence of the gradient based algorithms. Therefore, all the experiments are initiated with small random starting weights.

7.2 Performance Measure of Training Methods

It is a common practice to report the performance of a training algorithm based on number of epoch. To compare different algorithms; for example a variant of the first order, second order and derivative free training methods, number of epoch is not sufficient. For instance, in second order training methods, the number of epoch is less due to the extensive Hessian matrix computations, which determine descent directions and learning rates. The derivative free training methods on the other hand depend on the function evaluations. The following discussions are provided to point out the issues that relate to the performance of an algorithm.

7.3 Evaluation Metric

The factors, which are important to compare the algorithms, are the following:

a.) Generality, reliability and precision;
b.) Sensitivity to parameter and data;
c.) Computational efforts and

d.) Convergence.

- *Generality* of an algorithm refers to the wide variety of problems that the algorithm can handle. It may be possible to construct a test problem that a given algorithm cannot solve effectively. This property also suggests that a trained ANN can replicate the results in the presence of similar data pattern.

- *Reliability* or robustness points to the ability of the training method to solve most of the problems with reasonable accuracy in the similar class for which it is designed. The relationship between reliability of a training method, the problem size and its structure should be taken into account. Some training algorithms are reliable if the problem size is small and face difficulty if the problem size grows in size.

- *Precision* of an algorithm is its ability to achieve convergence quickly with satisfactory limit. High precision can be realized at the cost of computations for long time.

- *Sensitivity* of an algorithm is related with the initialization by user defined parameters such as:
  i.) Starting/initial weight vector;
  ii.) Learning rate;
  iii.) Momentum term;
  iv.) Accelerating factor and
  v.) Termination criteria.

  Some algorithms are sensitive to these parameters and to the problem data. Initialization or starting point greatly influence the training performance and produce different results. With a fixed set of parameters, the training algorithm should solve a problem for a wide range of data and should be *scale invariant*. The algorithm should be insensitive to data scaling or transformation.

- *Computational efforts* in total required by an algorithm are another measure for performance analysis. The first and second order derivatives evaluation require significant amount of time and has the advantage of fast convergence. Often these efforts are not measured in comparing performance of an algorithm. The algorithm that uses this information is relatively faster (Al-Sultan, 1997, Johansson et al., 1992; Luenberger, 1984 and Bazaraa et al., 1993). The computational efforts and its related computational burden should be taken into account for unbiased assessment. The computer time, the number of iterations, and the number of function evaluations measure the computational efforts of an algorithm. Any of these measures alone is not entirely a satisfactory metric. The efficiency of an algorithm not only depends on computer time but also on the type of machine used, the existing load on the machine,

the efficiency of coding and the character of the measured time. The number of iterations alone cannot be used as the only measure because the efforts per iteration may vary significantly from one method to another. The number of function evaluations can also be misleading, since it ignores the large computational burden of matrix multiplication, matrix inversion and evaluation of suitable directional vectors. The computations of the first and second order derivatives should be taken into account to measure the efficiency of an algorithm.

- *Convergence* is an important characteristic and usually occurs in a limiting sense. The quality or precision of solution generated by a trained ANN provides an indication of the algorithm's efficiency with the given amount of computational efforts. The convergence rate of an algorithm measures the amount by which an error function can be improved per epoch. It is one of the most important properties of an algorithm. Given two algorithms that converge, they could be compared on the basis of speed of convergence or order of convergence (Luenberger, 1984).

7.4 Experiment with 5-5-1 ANN Parity Problem

The 5-5-1 parity problem is simulated with small random weights. The training data set and related information are given in Chapter 3 and Appendix A. The experimental results with these proposed methods and the standard back propagation method are given below.

7.4.1 Analysis with Self-adaptive Back Propagation Method

The self-adaptive back propagation training algorithm is tested with the parity problem and Table 7.1 shows the results against the standard back propagation method. First, we observe the standard deviation and the range statistics to compare worst case behavior of the algorithm. The standard deviations are higher than the mean value in all performance measures with the proposed self-adaptive back propagation training method. It is therefore, apparent that the method exhibits erratic behavior in some experiments. Notice the experiment numbers 2, 8, 9 and 10, which show inconsistent performance.

The performance of the algorithm is inconsistent throughout with the standard back propagation method. Now let us look at the average performance of the self-adaptive algorithm in number of epoch and total number of function evaluations, which are 1534.5 and 67,636.6 respectively. The corresponding figures are 17,354.4 and 593,182.6 with the standard back propagation method. The maximum and minimum values for the number of epochs are 6,700 and 483 respectively with the proposed method while they are 49,709 and 765 with the standard back propagation method. Therefore, these are the probable limits within which we expect to reach a solution. Some iteration with the standard back propagation method is shown in Figure 7.1 to visualize the oscillation during convergence. An example of convergence with the proposed method is shown in Figure 7.2 while the

Figure 7.3 show the example of self-adaptive parameter generated by the algorithm for few epochs. The proposed method is showing better average performance against the standard back propagation training method and the relative improvements are (17,354.4/1,534.5) 11.31 in number of epoch and (538,072.4/67,636.6) 7.95 in total number of function evaluations. The average quality of the solution is better with the proposed method, since the terminal function value is significantly lower than the standard back propagation training method.

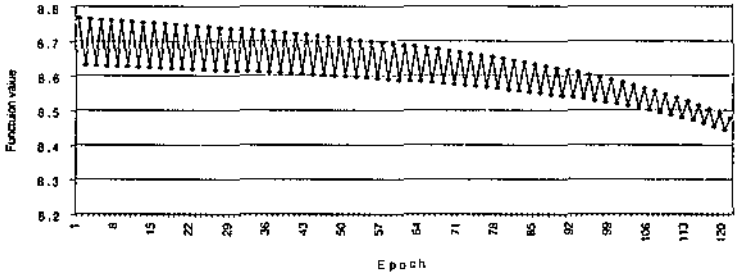


Figure 7.1 Function convergence with standard back propagation training (Parity problem: Some intermediate steps of long epoch length)

Expt #	Proposed Self-Adaptive Gradient Descent Training Method					Standard Back Propagation Training Method				
5-5-1 ANN Parity Problem	Epoch	Gradient evaluations	Function evaluations	Total No. of function evaluations	Terminal function value	Epoch	Gradient evaluations	Function evaluations	Total No. of function evaluations	Terminal function value
1	877	26340	9447	35787	3.10E-08	1339	1341	40200	41541	1.58E-04
2	1027	30840	10611	41451	5.40E-08	1246	37410	1278	38718	1.02E-02
3	483	14520	6330	20850	2.20E-08	32380	971430	32412	1003872	3.90E-02
4	898	26970	11859	38829	3.60E-06	7855	235680	7887	243597	5.16E-02
5	758	22770	10154	32924	2.10E-08	26576	797310	26608	823948	3.90E-02
6	664	19950	6969	26919	3.40E-08	49709	1491300	49741	1541071	4.57E-02
7	791	2360	9569	33329	2.96E-06	18795	563880	18827	582737	5.15E-02
8	1737	52140	24778	76918	1.20E-07	6848	205470	6880	212380	5.00E-02
9	6700	201030	109936	310966	8.70E-06	765	22980	797	23807	4.60E-03
10	1410	42330	16063	58393	1.80E-08	28031	840960	28063	869053	1.20E-02
Mean	1534.5	43925	21571.6	67636.6	1.56E-06	17354.4	516776.1	21269.3	538072.4	0.030376
Median	887.5	26655	10382.5	37308	4.4E-08	13325	399780	22717.5	413167	0.039
Standard Deviation	1851.52	56896.17	31500.15	87011.9	2.85E-06	16551.26	500848.9	16926.95	513095.4	0.021049
Range	6217	198670	103606	290116	8.68E-06	48944	1489959	48944	1517264	0.051442
Minimum	483	2360	6330	20850	1.8E-08	765	1341	797	23807	0.000158
Maximum	6700	201030	109936	310966	8.7E-06	49709	1491300	49741	1541071	0.0516

Table 7.1 Comparison with standard back propagation method and gradient descent self-adaptive training method (5-5-1 ANN: parity problem) with starting vector in small range

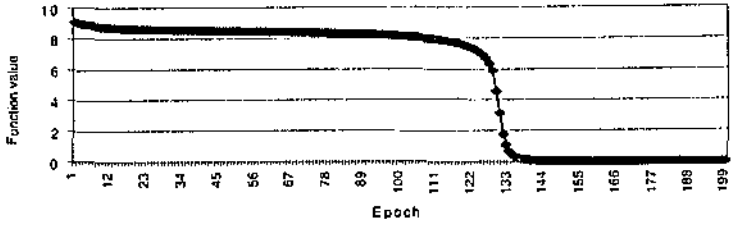


Figure 7.2 Function value with self-adaptive back propagation training method

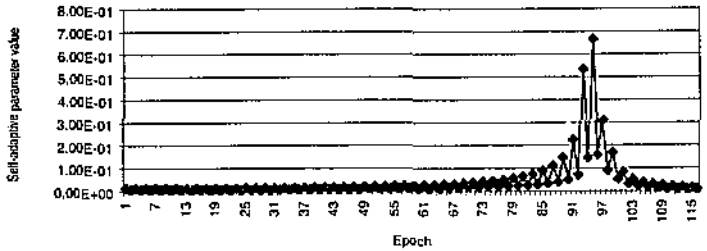


Figure 7.3 Self-adaptive parameter genetation with self-adaptive back propagation training method (Parity Problem)

7.4.2 Analysis with Multi-directional Training Method

Table 7.2 displays the multi-directional training results with the 5-5-1 ANN parity problem. Figure 7.4 shows the function convergence. The self-adaptive parameters generated by the algorithm that has different magnitudes are shown in figure 7.5 and Figure 7.6 shows the self-adaptive momentum term generated by the algorithm.

The average number of epoch is only 45.1 and the standard deviation is 21.48. The significant reduction in number of epoch, however, is not surprising. An interpolation search that determines the self-adaptive parameters reduces the number of epoch. The total numbers of function evaluations are important information to measure the total efforts made by the algorithm. Since this algorithm does not use gradient information, it is also important to take into account the total number of gradient evaluations made by a gradient-based algorithm. The average number of epoch with the back propagation training method is 17,354.4. The standard deviation in number of epoch is 16,551.26. With the proposed self-adaptive and standard back propagation training method, the average numbers of function evaluations are 40,945.2 and 538072.4 respectively.

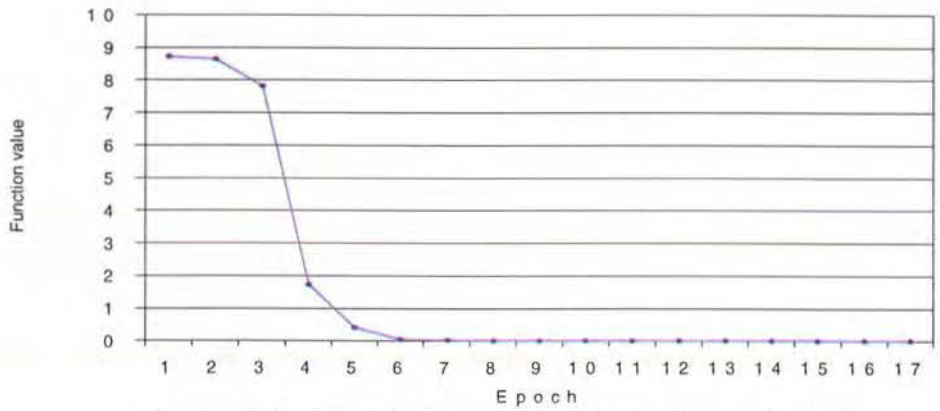


Figure 7.4 Convergence of Function value with multi-directional training method (Parity propbem)

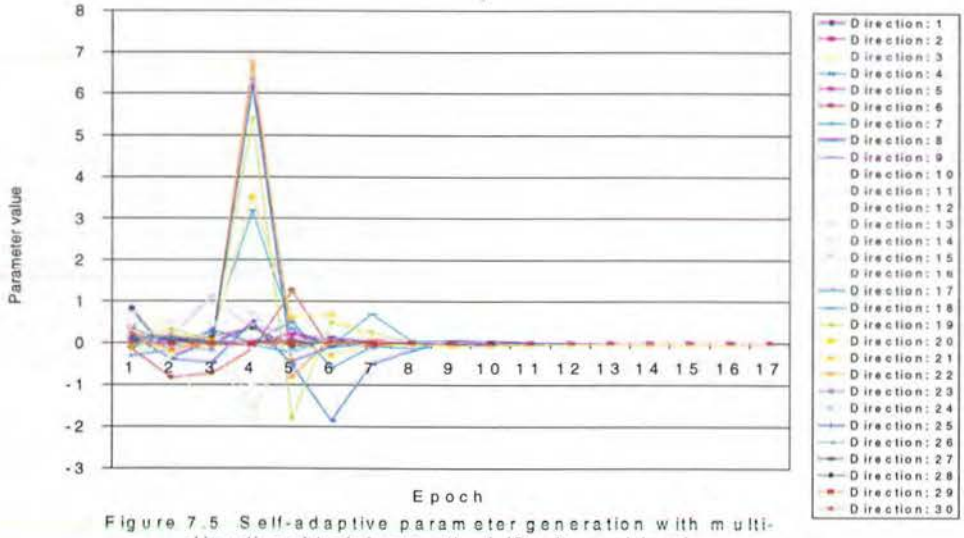


Figure 7.5 Self-adaptive parameter generation with multi-directional training method (Parity problem)

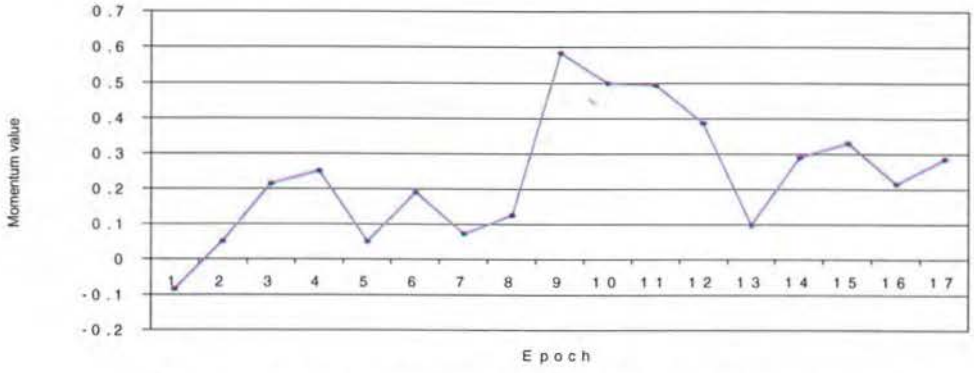


Figure 7.6 Momentum Term generated by the multi-directional training method (Parity problem)

The relative efficiency of the multi-directional training over the standard BP training in function evaluation is (538072.4/40,945.2) 13.14. The improvement can be attributed to the acceleration step, which includes an exact momentum search and the application of an oriented search vector. The efficiency is also gained by computing the self-adaptive learning rates for all the network weights independently. The algorithm trains successfully an ANN without computing the derivative of the error function.

Expt #	Proposed Multi-Directional Training Method			Proposed Restart Training Method			Standard Back Propagation Training Method		
5-5-1 Parity Problem	Epoch	Total No. of function evaluations	Terminal function value	Epoch	Total No. of function evaluations	Terminal function value	Epoch	Total No. of function evaluations	Terminal function value
1	50	44713	1.62E-05	33019	43699	0.00348	1339	41541	1.58E-04
2	83	77839	9.45E-07	8260	11148	0.0083	1246	38718	1.02E-02
3	37	34271	1.85E-06	10631	14023	0.00038	32380	1003872	3.90E-02
4	33	29811	9.30E-06	4524	6198	1.1E-06	7855	243597	5.16E-02
5	49	45835	2.24E-06	12416	16146	2E-07	26576	823948	3.90E-02
6	51	45450	2.07E-06	7506	10105	3.1E-06	49709	1541071	4.57E-02
7	19	16131	1.36E-06	4439	5989	0.000008	18795	582737	5.15E-02
8	26	22363	9.60E-07	1477	18802	0.000049	6848	212380	5.00E-02
9	26	22293	6.20E-08	17686	22617	0.000013	765	23807	4.60E-03
10	77	70746	8.56E-07	13142	17105	0.0013	28031	869053	1.20E-02
Mean	45.1	40945.2	3.58E-06	11310	16583.2	0.001353	17354.4	538072.4	0.030376
Median	43	39492	1.61E-06	9445.5	15084.5	0.000031	13325	413167	0.039
Standard Deviation	21.476	20507.73	5.14E-06	9010.46	10930.14	0.002677	16551.3	513095.4	0.021049
Range	64	61708	1.61E-05	31542	37710	0.0083	48944	1517264	0.051442
Minimum	19	16131	6.2E-08	1477	5989	2E-07	765	23807	0.000158
Maximum	83	77839	1.62E-05	33019	43699	0.0083	49709	1541071	0.0516

Table 7.2 The multi-directional, restart and standard back propagation training methods (5-5-1 ANN: parity problem) with starting vector in small range

7.4.3 Analysis with Restart Training Method

The restart training results are listed in Table 7.2 and Figure 7.7 shows the convergence of the algorithm in early stage of iterations. Interesting enough, the average value of the number of epoch is 11,310 while the average numbers of total function evaluations are only 16,583.2. It updates all the network weights concurrently in a single phase and the method identifies better function value within few numbers of iterations. The average gains in epoch and function evaluations against the standard back propagation method are (17,354.4/11,310) 1.53 and (538072.4/16,583.2) 32.45 respectively.

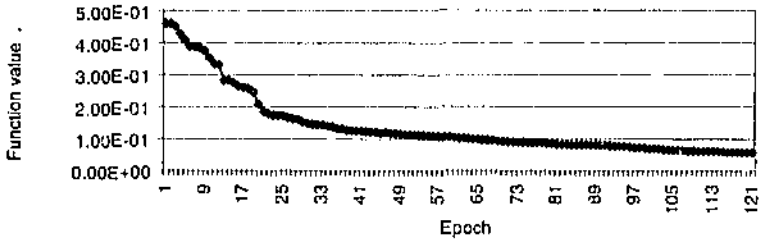


Figure 7.7 Function convergence with restart training method (5-5-1: Parity Problem)

7.4.4 Comparison with Different Training Methods

Table 7.3a shows the comparison with different training methods. The restart training needs less number of function evaluations to reach acceptable solution. We exclude number of epoch to rank an algorithm. It is not a correct measure, since we have two algorithms that do not use gradient information at all. The relative effectiveness of the restart training over the standard back propagation method is 38.60. The restart training algorithm ranks first followed by the multi-directional and the self-adaptive back propagation-training algorithm in function evaluations.

The result reported in Johansson et al. (1992) with Polak-Ribiere (1969) conjugate gradient method is considered for comparison. The average function evaluations are 10947.67. The training experiments that converge are considered. The minimum and maximum numbers of function evaluations are 21,633 and 1966 respectively. The proposed method does not improve over this method in function evaluations. Figure 7.8, 7.9 and 7.10 show the comparison in epoch, function evaluations and relative efficiencies between the training methods.

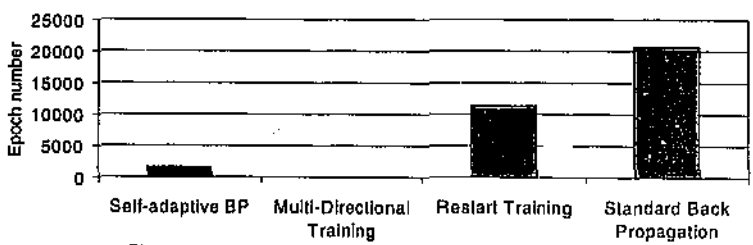


Figure 7.8 Average epoch comparison with different training methods (5-5-1 parity problem)

5-5-1 Parity Problem	Performance Measures	Self-adaptive BP	Multi-Directional Training	Restart Training	Standard Back Propagation	C.G: P-R Johansson et al. (1992)
epoch	Average	1534.5	45.1	11310	17354.4	
	S. Deviation	1851.52	21.48	9010.5	16551.26	
	Max	6700	83	33019	49709	
	Min	483	19	1477	765	
	Speed up	13.46	457.9	1.83	-	
Function Evaluation	Average	67636.6	40945.2	16583.2	538072.4	10947.67
	S. Deviation	87011.9	20507.7	10930.14	513095.4	
	Max	370966	77839	43699	1541071	21633
	Min	20850	16131	5989	23807	1966
	Speed up	9.47	15.64	38.6	-	-
Function Value	Average	0.00000156	0.00000358	0.00135300	0.03038	
	S. Deviation	0.00000285	0.00000514	0.00267700	0.02105	
	Max	0.00000870	0.00001620	0.00830000	0.0516	
	Min	0.00000870	0.00001620	0.00830000	0.000158	
	Gain	16001.9	6972.9	18.45	-	

Table 7.3a Comparison with different training methods

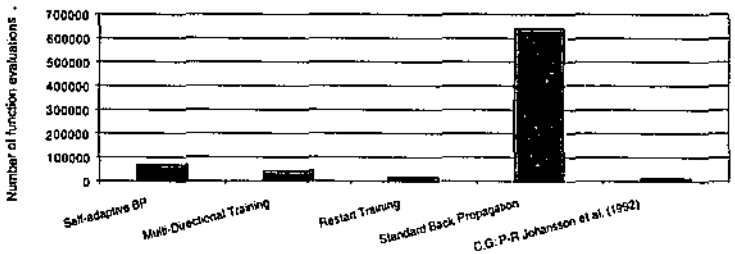


Figure 7.9 Average function evaluations comparison with different training methods (Parity Problem: 5-5-1)

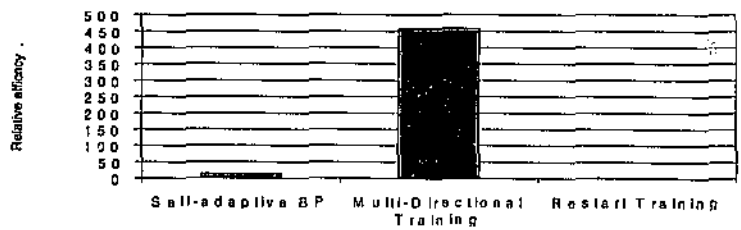


Figure 7.10 Average speed up in epoch measure of the proposed methods over standard back propagation training method (5-5-1: Parity problem)

7.4.5 Examples of Trained ANN Weights

Table 7.3b shows the examples of trained ANN weights obtained by the self-adaptive back propagation training method (Ahmed, Cross and Bouzerdoum, 2000b). The input and output comparison is shown in Table 7.3c.

Input Layer Weights (w_{in})

$n_j \rightarrow$ / $i \downarrow$	1	2	3	4	5
1	0.282172	2.201292	0.133201	0.605853	0.073884
2	0.340192	2.345989	0.161205	0.7227a9	0.089164
3	0.329249	2.498341	-0.18031	0.858935	0.500701
4	-0.24439	-3.0464	1.509324	-2.41014	1.548676
5	0.349717	2.365912	0.165818	0.741631	0.091688

Output Layer Weights (w_{no})

$n_j \rightarrow$ / $o \downarrow$	1	2	3	4	5
1	-0.906871	0.711706	0.803807	0.917288	0.658517

Table 7.3b Trained weights for 5-5-1 parity problem

#	1	2	3	4	5	6	7	8	9	10	11
Actual	0	1	1	0	1	0	0	1	1	0	0
Estimate	0	1.0002040	0.999718	0.000317	1.000149	-0.00011	0.00016	0.999967	1.00019	-0.00015	0.000319
Error	0	-0.0002	0.000282	-0.00032	-0.00015	0.000107	-0.00016	3.26E-05	-0.00019	0.000148	-0.00032

#	12	13	14	15	16	17	18	19	20	21	22
Actual	1	0	1	1	0	1	0	0	1	0	1
Estimate	0.999746	-9.9E-05	1.000209	0.999953	0.000144	1.000082	-4.7E-05	0.000333	0.999731	-3.9E-05	1.000177
Error	0.000254	9.93E-05	-0.00021	4.74E-05	-0.00014	-8.2E-05	4.74E-05	-0.00033	0.000269	3.86E-05	-0.00018

#	23	24	25	26	27	28	29	30	31	32
Actual	1	0	0	1	1	0	1	0	0	1
Estimate	0.999838	0.00028	-3.4E-05	1.000177	0.999728	0.000469	1.000171	-7.7E-05	0.000297	0.999763
Error	0.000162	-0.00028	3.36E-05	-0.00018	0.000272	-0.00047	-0.00017	7.65E-05	-0.0003	0.000237

Table 7.3c The Input and output value comparison (5-5-1 ANN Parity problem)

7.5 Simulations with L-T Letter Recognition Problem

The 9-2-1 ANN model to recognize the letter L-T is now considered. The small random starting points are used to initiate the training. The learning rate for standard back propagation method is 0.001. The learning rates 0.1 and 0.01 face convergence difficulties.

7.5.1 Analysis with Self-adaptive Back Propagation Method

The self-adaptive back propagation training method is tested with the L-T letter recognition problem. Table 7.4 shows the results against the standard back propagation training method. The convergence of the algorithm is shown in Figure 7.11 and the self-adaptive parameter generated by the training algorithm is shown in Figure 7.13. The average number of epoch with the self-adaptive back propagation method is 178.8. The maximum and minimum values in numbers of epoch are 323 and 53 respectively. The average numbers of total function evaluations are 6485.8 and the maximum and minimum values are 10565 and 2967 respectively. The average terminal function value is 1.28×10^{-9}.

Expt #	Self-adaptive back propagation					Standard back propagation				
L-T Letter Recognition	Epoch	Function evaluations	Gradient evaluations	Total Function evaluations	Function value	Epoch	Function evaluations	Gradient evaluations	Total Function evaluations	Function value
1	323	4085	6480	10565	9.06E-11	2434	2436	48700	51136	4.87E-04
2	257	3290	5160	8450	4.57E-10	2569	2571	51400	53971	4.96E-05
3	265	3584	5320	8904	2.25E-10	1203	1205	24080	25285	4.87E-04
4	115	1360	2320	3680	5.83E-10	2277	2279	45560	47839	4.80E-04
5	303	3671	6080	9751	1.41E-09	2254	2256	45100	47356	4.88E-04
6	123	1545	2480	4025	1.04E-09	3165	3167	63320	66487	3.50E-04
7	117	1379	2360	3739	2.42E-10	2263	2265	45280	47545	2.71E-04
8	92	1107	1860	2967	1.77E-10	2242	2244	44860	47104	4.76E-04
9	53	7039	1080	8119	7.63E-09	2568	2570	51380	53950	4.70E-04
10	140	1838	2820	4658	9.64E-10	802	804	16060	16864	4.99E-04
Mean	178.8	2889.8	3596	6485.8	1.28E-09	2177.7	2179.7	45753.7	43574	0.000406
Median	131.5	2564	2650	6388.5	5.2E-10	2270	2272	47692	45420	0.000478
Standard Deviation	97.54406	1844.29	1950.881	2922.523	2.27E-09	684.3963	684.3963	14372.32	13687.93	0.000146
Range	270	5932	5400	7598	7.54E-09	2363	2363	49623	47260	0.000449
Minimum	53	1107	1080	2967	9.06E-11	802	804	16864	16060	4.96E-05
Maximum	323	7039	6480	10565	7.63E-09	3165	3167	66487	63320	0.000499

Table 7.4 Comparison with standard back propagation and self-adaptive training method (9-2-1 ANN: L-T letter recognition problem) with starting point vector in small magnitude

The average number of epoch is 2177.7 with the standard back propagation training. The maximum and minimum numbers of epoch are 3165 and 802 respectively. The convergence pattern is shown in Figure 7.12. The average value of the total number of function evaluation is 43547. The maximum and minimum values are 63,320 and 16,060 respectively. The relative efficiency of the self-adaptive back propagation training algorithm over the standard back propagation training algorithm in number of epoch is (2177.7/178.8) 12.18, while the efficiency in total number of function evaluation is

(43,574/6485.8) 6.72. The standard back propagation training method does not show oscillations in convergence with this problem. The reason can be attributed to the number of parameters, which is more than the number of training set available to the ANN.

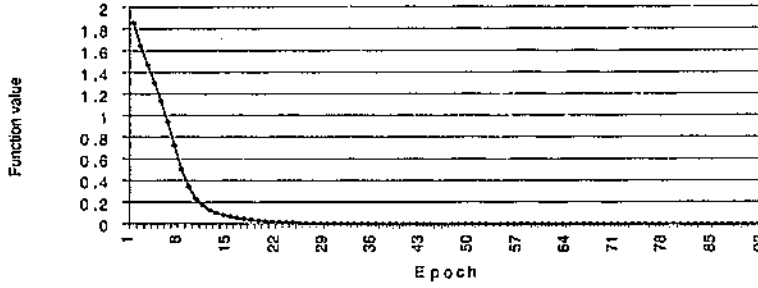


Figure 7.11 Convergence with self-adative back propagation training method (L-T letter recognition)

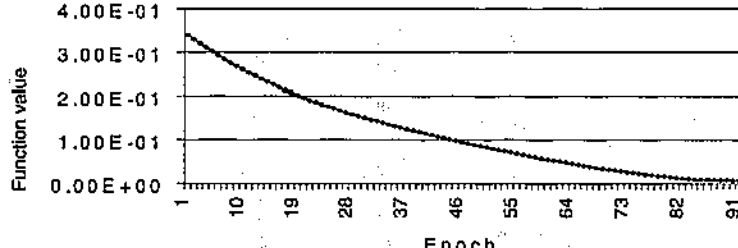


Figure 7.12 Convergence with standard BP training method (L-T letter recognition problem)

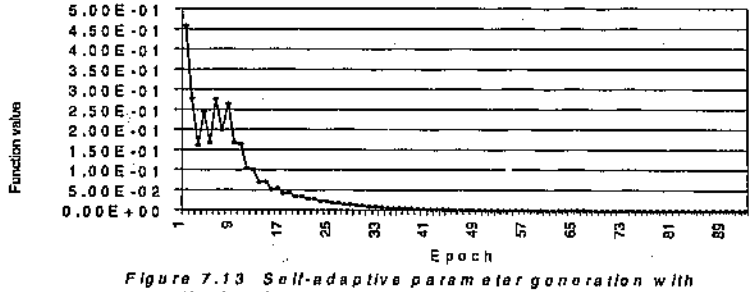


Figure 7.13 Self-adaptive parameter generation with self-adaptive back propagation training method

7.5.2 Analysis with Multi-directional Training Method

Table 7.5 also displays the results with the L-T letter recognition problem. Figure 7.15 shows the convergence of the method with momentum parameters. The dynamic self-adaptive parameters generated by the algorithm are shown in Figure 7.14.

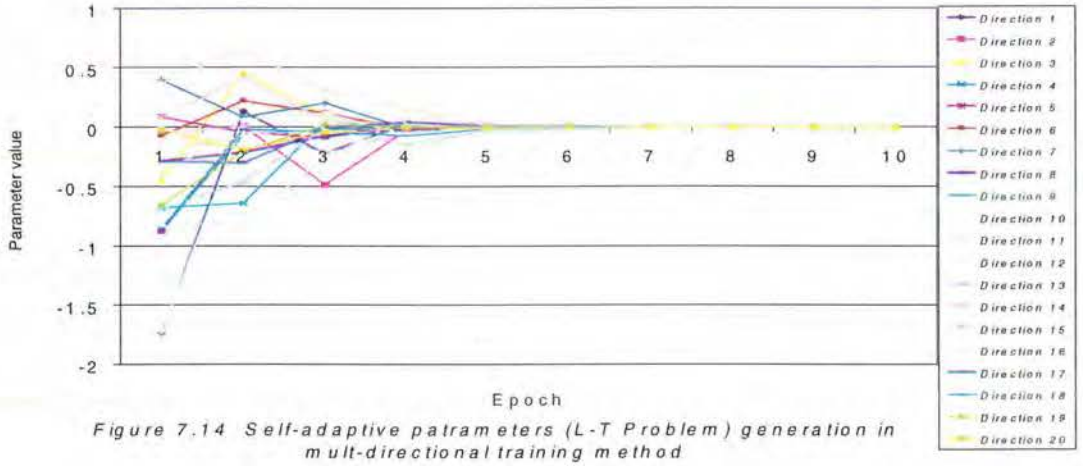


Figure 7.14 Self-adaptive patrameters (L-T Problem) generation in mult-directional training method

Expt #	Multi-Directional			Restart			Standard back propagation		
L-T Letter Recognition	Epoch	Function evaluations	Function value	Epoch	Function evaluations	Function value	Epoch	Function evaluations	Function value
1	15	8645	1.51E-09	1891	2815	2.65E-08	2434	51136	4.87E-04
2	12	6909	5.44E-10	2653	3806	0.00109	2569	53971	4.96E-05
3	16	9547	3.07E-09	867	1452	1.57E-08	1203	25285	4.87E-04
4	11	6361	4.97E-10	1582	2424	5.5E-09	2277	47839	4.80E-04
5	11	5981	1.45E-10	958	1674	7.2E-09	2254	47356	4.88E-04
6	10	5372	8.18E-10	1118	1795	1.49E-08	3165	66487	3.50E-04
7	13	7249	1.90E-10	992	1598	3.4E-09	2263	47545	2.71E-04
8	9	5032	1.24E-09	741	1246	7.07E-7	2242	47104	4.76E-04
9	26	15422	4.99E-09	1169	1866	3.12E-08	2568	53950	4.70E-04
10	14	7958	1.73E-09	1196	1786	3.54E-08	802	16864	4.99E-04
Mean	13.7	7847.6	1.47E-09	1316.7	2046.2	0.000109	2177.7	43574	0.000406
Median	12.5	7079	1.03E-09	1143.5	1790.5	2.11E-08	2270	45420	0.000478
Standard Deviation	4.8545	3016.57	1.52E-09	580.1847	769.327	0.00035	684.396	13687.9	0.00015
Range	17	10390	4.85E-09	1912	2560	0.00109	2363	47260	0.000449
Minimum	9	5032	1.45E-10	741	1246	3.4E-09	802	16060	4.96E-05
Maximum	26	15422	4.99E-09	2653	3806	0.00109	3165	63320	0.000499

Table 7.5 Comparison with multi-directional, restart and standard back propagation

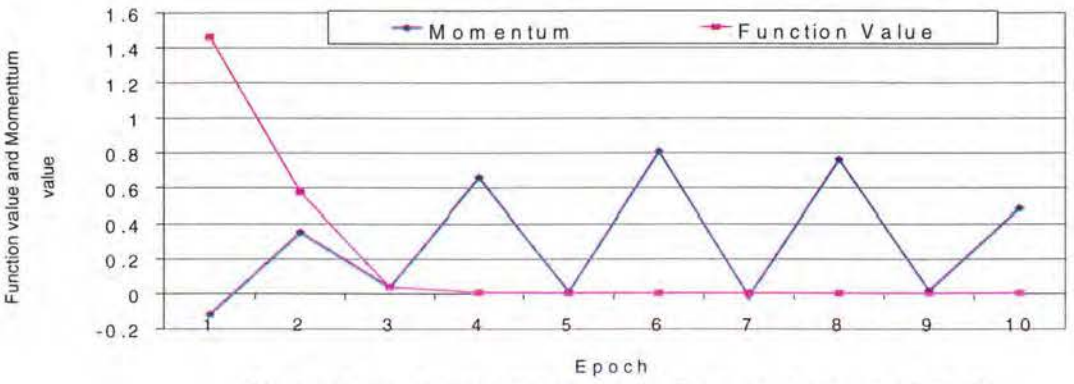


Figure 7.15 Function value and momentum term in multi-directional training method

The average terminal error function value is of the order 1.47×10^{-9}. This indicates that the training is able to escape local minimum. The average numbers of epoch are 13.7 and the standard deviation is 4.85. The maximum and minimum numbers of epochs are 26 and 9 respectively.

The average numbers of function evaluations are 7847.6 and the maximum and minimum values are 15,422 and 5032 respectively.

In comparison, the multi-directional training method improves by the factor (43574/7847.6) 5.55 in total number of function evaluations. The epoch comparison is not appropriate, since the proposed method used specialized interpolation search at the cost of function evaluations. However, the relative efficiency is (2177.7/13.7) 158.95 over the standard back propagation method in number of epoch.

7.5.3 Analysis with Restart Training Method

Table 7.5 also includes the results with the L-T letter recognition problem and Figure 7.16 shows the convergence of the method. The average terminal error function value is 0.000109. This indicates that the ANN is well trained. The average number of epoch is 1316.7 while the standard deviation is 580.18. The maximum and minimum numbers of epochs are 2653 and 741 respectively. The average numbers of function evaluations are 2046.2 and the maximum and minimum values are 3806 and 1246 respectively.

In comparison, the restart training method improves by a factor (43574/2046.2) 21.3 in total number of function evaluations against the standard BP training. The relative efficiency is (2177.7/1317.7) 1.65 over the standard back propagation training method in number of epoch.

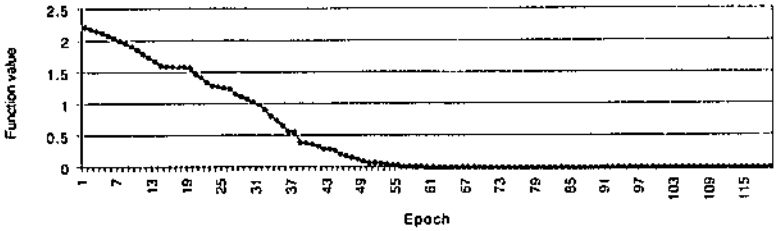


Figure 7.16 Function convergence with restart training method (L-T letter recognition problem)

7.5.4 Comparison with Different Training Methods

Table 7.6 shows the comparison with different training methods including the result given in Kamarthi et al. (1999) and Vogl et al. (1988).

All the proposed methods perform better than the weight extrapolation method suggested by Kamarthi et al. (1999). They train L-T letter recognition problem in 1811 epoch and the terminal function value is 0.00001 at the end of training. The self-adaptive and multi-directional training methods find terminal function value, which is comparatively less.

L-T Letter Recognition	Performance Measures	Self-adaptive BP	Multi-Directional Training	Restart Training	Standard Back Propagation	Kamarthi et al. (1999)	Vogl et al. (1988) with T-C	C.G (P-R) in Kamarthi et al. (1999)
Epoch	Average	178.8	13.7	1316.7	2177.7	1811	826	5
	S. Deviation	97.54	4.8	580.18	684.4			
	Max	323	26	2653	3165			
	Min	53	9	741	802			
	Speed up	12.18	158.95	1.65	-	-	-	
Function Evaluation	Average	6485.8	7847.6	2046.2	43574			
	S. Deviation	2922.5	3016.6	769.32	13687.9			
	Max	10565	15422	3806	63320			
	Min	2967	5032	1246	16060			
	Speed up	6.72	5.55	21.3	-			
Function Value	Average	1.28E-9	1.47E-9	0.00011	0.000406	0.00001		
	S. Deviation	2.27E-9	1.52E-9	0.000345	0.000146			
	Max	7.63E-9	4.99E-9	0.0011	0.0005			
	Min	9.06E-11	1.45E-10	3.4E-9	4.96E-5			
	Speed up	-	-	-	-			

Table 7.6 Training performance with 9-2-1 ANN: L-T letter recognition problem

In a related study Vogl et al. (1988) report computational experience with T-C letter recognition problem. The back propagation training method suggested by them takes 826

epochs to train the ANN. The task is equivalent to the L-T letter recognition problem. The proposed self-adaptive and multi-directional training method is efficient for this job. The restart training method takes 1316.7 numbers of epochs to train the ANN and does not improve against these methods. Kamarthi et al. (1999) report that the conjugate gradient method takes 5 epochs to train this problem.

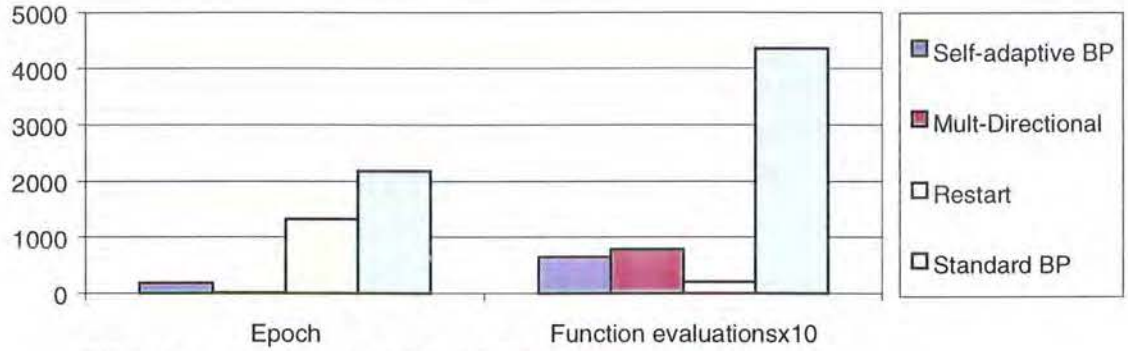


Figure 7.17 Function evalutions and epoch measure comparison

7.6 Simulation with Seasonal Time Series Problem

The forecast models are used extensively for prediction or estimation. Before a model is released for use, validation should be made. We point out two measures namely, adequacy checking and model validation. Model adequacy checking includes residual analysis, testing for lack of fit and other internal analysis that investigate the fit to the forecast model to the available data. The model adequacy checking is preformed based on the measures given in Chapter 3, Section 3.5.1.

Model validation is directed towards determining if the model will function successfully in its intended-operating environment. Three types of procedure are useful for validating a forecast model. They are:

a.) analysis with the predicted values and comparison with prior experience;
b.) collection of fresh data with which to investigate the model's predictive performance and
c.) data splitting: that is setting aside some of the original data and using these observations to investigate the model's predictive performance.

We use the last measure as discussed in Section 3.4.3 in Chapter 3. The 5-5-1 ANN model is trained to fit the seasonal time series data. The small magnitude random staring points are used for simulation experiments.

7.1.1 Analysis with Standard Statistical Regression Method

The time series model given in Chapter 3 is solved using standard statistical regression method. The following Table 7.7 summarizes the results in error measure. The model is evaluated with the data points in test period as explained in Chapter 3. The forecast is validated in validation period data. The statistics of interest are MAPE, MAE, R^2 and SSE.

The mean error in test period is zero, so that the model seems to produce approximately unbiased predictions. Later, we will see that the standard statistical regression method and the multivariate training method produce exactly the same results with a 5-5-1 ANN configuration that models quarterly seasonal time series and hence the forecast performance is equivalent.

Error measure	SSE	MSE	MPE	MAE	MAPE	R^2	ME
Test Period	46924099	461723.8	-0.04952	566.1418	1.801725	0.986	0
Validation Period	97003724	6062733	-5.34	52323.9	5.48	-	-2260

Table 7.7 Results with standard statistical method

7.1.2 Analysis with Self-adaptive Back Propagation Training

Table 7.8 shows the self-adaptive training results against the standard back propagation training and Figure 7.19 show the convergence of the forecasting problem with self-adaptive training method. Figure 7.18 shows the plot in test and validation period.

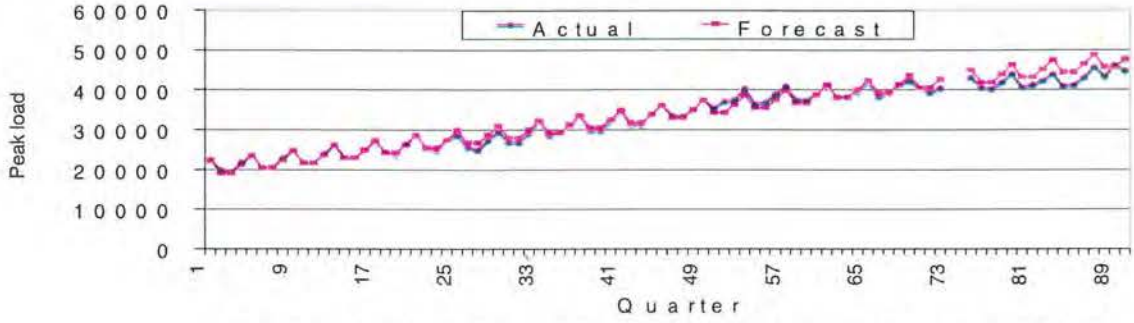


Figure 7.18 Training with self-adaptive BP (Seasonal time series: 5-5-1: ANN)

From the given set of simulation results, we select an experiment that provides the best function value in forecast problem. The results are compared with the standard regression method. As usual, the convergence properties are compared against the standard back propagation training. The results in Table 7.8 show that on average the self-adaptive

training needs 1897 numbers of epochs and 75,941.4 function evaluations. The standard back propagation training is not converging within 50,000 epochs and the function value at the end of training is reported. Therefore, the comparison is limited within the developed algorithm that converge and the standard statistical regression method.

The average function value is 46924312 with the self-adaptive back propagation training and the range statistics is 1308.79. In a set of experiment the algorithm finds function values that are not significantly different. The minimum function value is 46924099, which could be used in forecast applications. Incidentally this value coincides with the statistical forecast method. The self-adaptive parameters generated for this problem is shown in Figure 7.20 before convergence.

Expt #	Self-adaptive BP					Standard BP				
Time Series Problem	Epoch	Function evaluation	Gradient evaluation	Total Function evaluation	Function value	Epoch	Function evaluation	Gradient evaluation	Total Function evaluation	Function value
1	707	7417	21240	28657	46924242.06	50000	*	*	*	54699986.66
2	556	5390	16710	22100	46924279.59	50000	*	*	*	54701871.15
3	3897	36219	116940	153159	46924366.13	50000	*	*	*	54715517.66
4	4079	32426	122400	154826	46924175.58	50000	*	*	*	54727725.94
5	238	2960	7170	10130	46924202.92	50000	*	*	*	54712167.02
6	1590	18236	47730	65966	46925407.64	50000	*	*	*	54722449.65
7	5414	59002	162450	221452	46924099.67	50000	*	*	*	54715157.89
8	838	11816	25170	36986	46924147.59	50000	*	*	*	54720516.26
9	649	6229	19500	25729	46924098.85	50000	*	*	*	54719702.94
10	1002	10319	30090	40409	46924100.65	50000	*	*	*	54710140.07
Mean	1897	19001.4	56940	75941.4	46924312			not	converging	54714524
Median	920	11067.5	27630	38697.5	46924189					54715338
Standard Deviation	1846.06	18086.9	55382	73200.9	394.6892					8802.965
Range	5176	56042	155280	211322	1308.792					27739.34
Minimum	238	2960	7170	10130	46924099					54699987
Maximum	5414	59002	162450	221452	46925408					54727726

Table 7.8 Convergence with self-adaptive back propagation training method

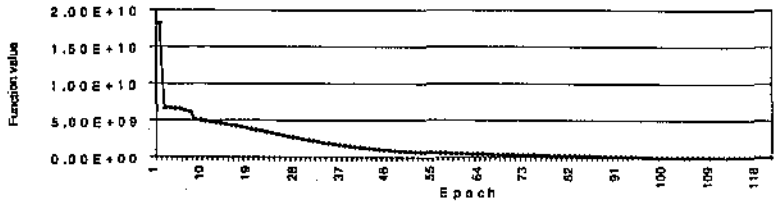


Figure 7.19 Convergence with self-adaptive back propagation training (Seasonal time series: 5-5-1 ANN)

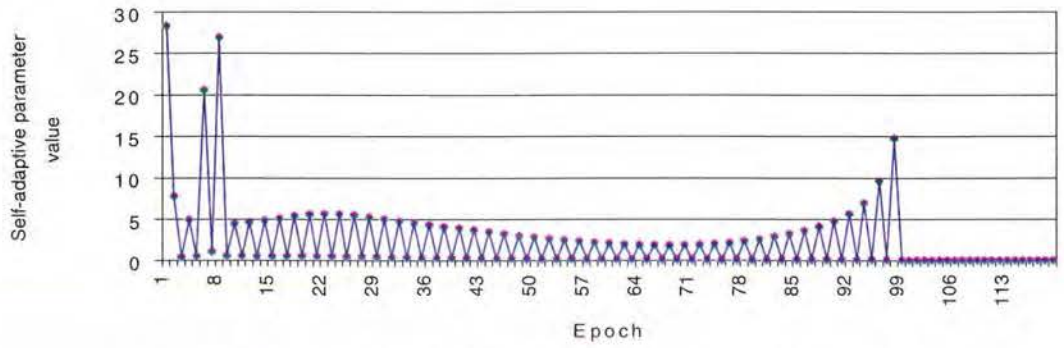


Figure 7.20 Self-adaptive parameter generation with seasonal time series problem

7.1.3 Analysis with Multi-directional Training Method

Table 7.9 displays the results with the multi directional training method. Figure 7.21 shows the convergence of the method. The average number of epoch to train this problem is 13.4 and the average numbers of function evaluations are 21379.1. The average terminal function value is 46924099. The Figure 7.24 shows the training performance of the algorithm in test and validation period while the Figure 7.22 and 7.23 show the self-adaptive and momentum parameters generated by the algorithm during training.

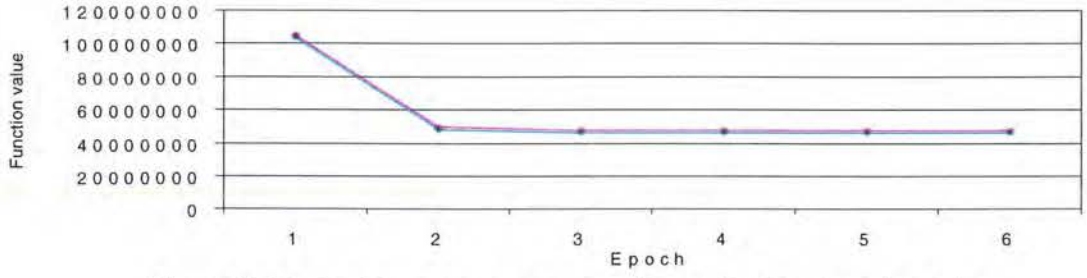


Figure 7.21 Training convergence with multi-directional training method (Seasonal time series)

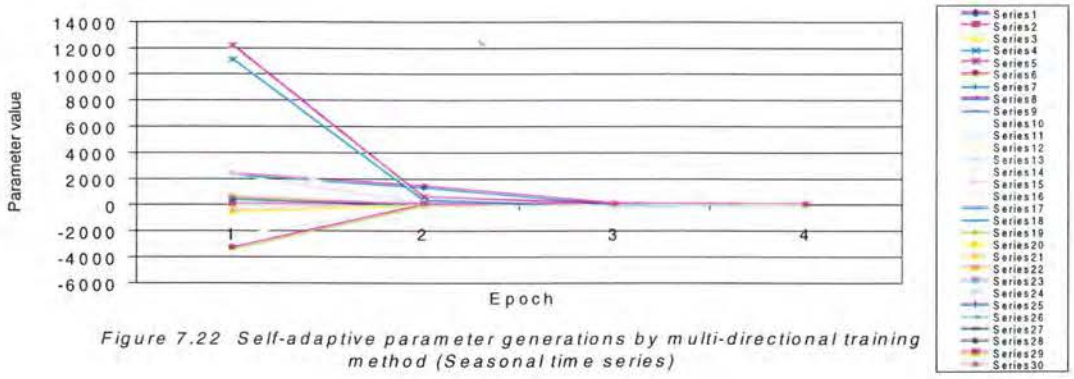


Figure 7.22 Self-adaptive parameter generations by multi-directional training method (Seasonal time series)

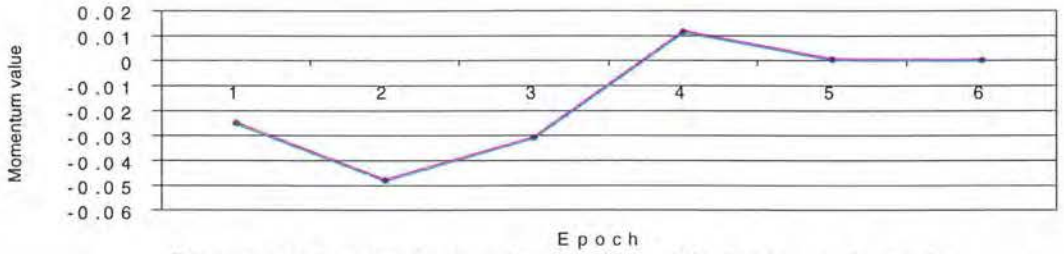


Figure 7.23 Momentum term with multi-directional training method (Seasonal time series)

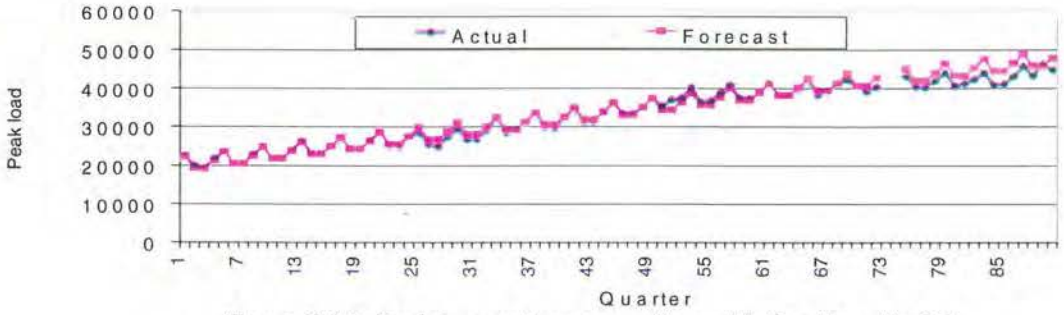


Figure 7.24 Training performace with multi-directional training method (Seasonal time series)

Expt #	Multi-Directional			Restart			Standard BP		
Seasonal Time Series	Epoch	Function evaluation	Function value	Epoch	Function evaluation	Function value	Epoch	Function evaluation	Function value
1	13	20694	46924098.8	7058	11288	46924030	n/c	n/c	54699986.6
2	15	23074	46924098.8	52217	27772	46894170	n/c	n/c	54701871.1
3	11	18288	46924098.8	45529	28725	46920720	n/c	n/c	54715517.6
4	11	18163	46924098.8	12411	18025	46922080	n/c	n/c	54727725.9
5	13	20915	46924098.8	84829	162497	46889040	n/c	n/c	54712167.0
6	11	18732	46924098.8	48698	96635	45853060	n/c	n/c	54722449.6
7	18	27512	46924098.8	7958	13091	46923890	n/c	n/c	54715157.9
8	17	26367	46924098.8	21953	39011	46923790	n/c	n/c	54720516.3
9	13	20710	46924098.8	6892	10708	46923020	n/c	n/c	54719702.9
10	12	19336	46924098.8	16181	10038	46923100	n/c	n/c	54710140.3
Mean	13.4	21379.1	46924099	30372.6	41779	46809690			54714524
Median	13	20702	46924099	19067	22898.5	46922550			54715338
Standard Deviation	2.50333	3292.69	0.666667	26247.6	49717.1	336381.6			8802.965
Range	7	9349	0.04099	77937	152459	1070970			27739.34
Minimum	11	18163	46924099	6892	10038	45853060			54699987
Maximum	18	27512	46924099	84829	162497	46924030			54727726

Table 7.9 Comparison multi-directional, restart and standard back propagation method with random starting point in small range

7.1.4 Analysis with Restart Training Method

Table 7.9 also lists the restart training results with the seasonal time series problem and Figure 7.25 shows the training performance. The forecast performance in training and validation period is shown in Figure 7.26. The average number of epoch to train this problem is 30,372.6 and the average numbers of function evaluations are 41,779. The average terminal function value is 46809690. The minimum function value for this training algorithm is 45853060, which is less than the statistical regression method.

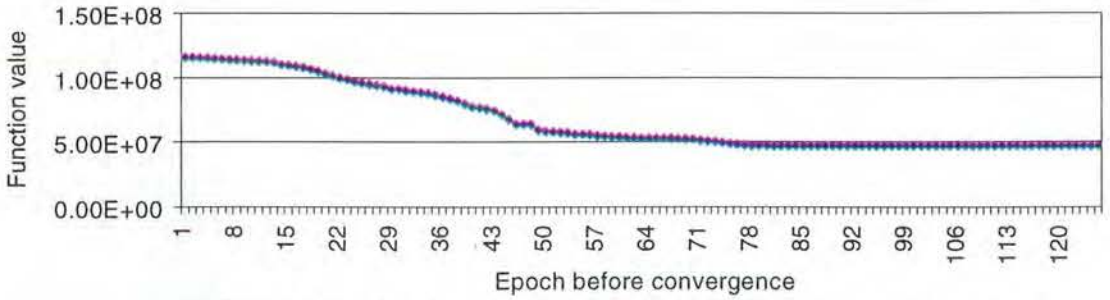


Figure 7.25 Function convergence with multi-directional training method (Seasonal time series problem)

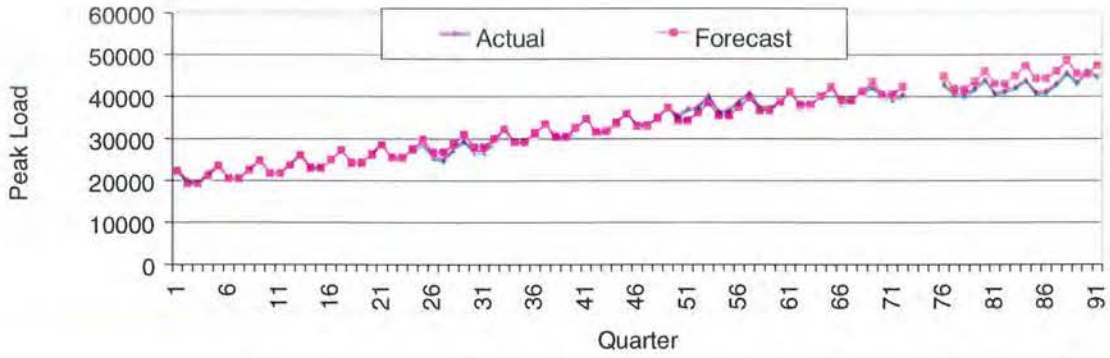


Figure 7.26 Training performance with restart training method (Seasonal time series: 5-5-1 ANN)

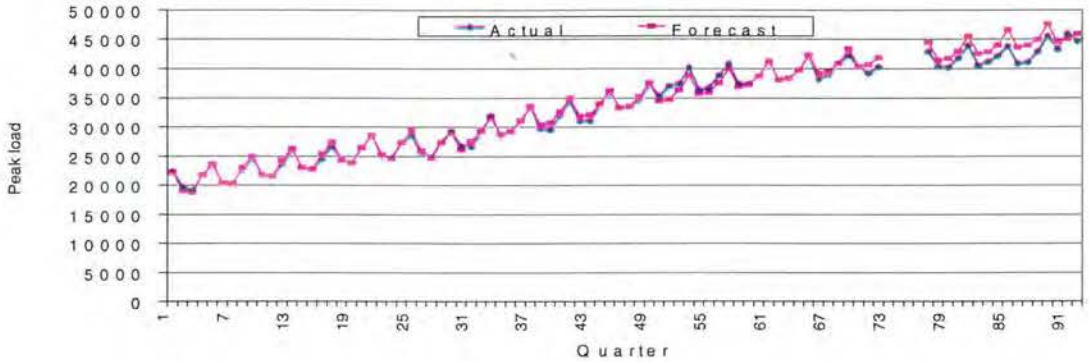


Figure 7.27a An example of forecast with 5-4-1 ANN configuration (Restart training)

As an example; a different 5-4-1 ANN configuration produces terminal function value, which is 28495221 in training period and 48711721 in validation period. These values are significantly less in comparison with the standard statistical regression method. The corresponding MAPE values are 1.35 and 3.9 respectively. Figure 7.27a show the comparison of forecast with the actual data.

7.1.5 Comparison with Different Training Methods

Table 7.10 shows the comparison between the different training methods. Figure 7.27b compares the performance of the algorithms in function evaluations.

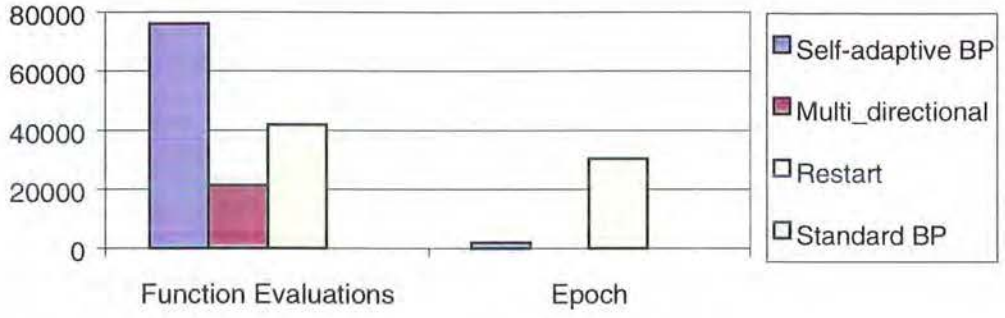


Figure 7. 27b Comparison with average number of evaluation

Seasonal Time Series	Performance Measures	Self-adaptive BP	Multi-Directional Training	Restart Training	Standard Back Propagation	Remarks
epoch	Average	1897	13.4	30372.6	50000(*)	Standard BP training reached maximum limit of epoch and not converging. Hence excluded from comparison.
	S. Deviation	1846	2.5	26247.6		
	Max	5414	18	84829		
	Min	238	11	6892		
	Speed up	-	-	-		
Function Evaluation	Average	75941.4	21379	41779		
	S. Deviation	73200.94	3292.7	49717.13		
	Max	221452	27512	162497		
	Min	10130	18163	10038		
	Speed up	-	-	-		
Function Value	Average	46924312	46924099	46809690	54714524	
	S. Deviation	394.69	.66	336381.6	8803	
	Max	46925408	46924099	46924030	54727726	
	Min	46924099	46924099	45853060	54699987	

Table 7.10 Comparison with different training methods

The statistical error measures in training and validation period are shown in Table 7.11 and Figure 7.28, 7.29 and 7.30 show the comparisons in MAPE, MSE and MPE measures. The forecast performance of the restart algorithm is the best with the given set of initial conditions.

Seasonal Time Series Problem	Performance Measures	Self-adaptive BP	Multi-Directional Training	Restart Simplex Training	Standard BP	Statistical Method
Training Period	MSE	651723.6	651723.6	6368458.5	689644	651723.6
	MAPE	1.8017	1.802	1.754	1.962	1.8017
	MPE	-0.0495	-0.050	0.163	-.45	-0.0495
	R^2	0.9857	0.986	0.99	.945	0.986
	Epoch	1897	13.4	30372.6	Max limit	-
	Function Evaluation	75941	21379	41779	Max limit	-
Test Period	MSE	6062795	6062732.7	5187565	7485377	6062732.7
	MAPE	5.475	5.476	5.0951	6.08	5.476
	MPE	-5.34	-5.336	-4.85	-6.05	-5.336

Table 7.11 Statistical measures in training and validation period

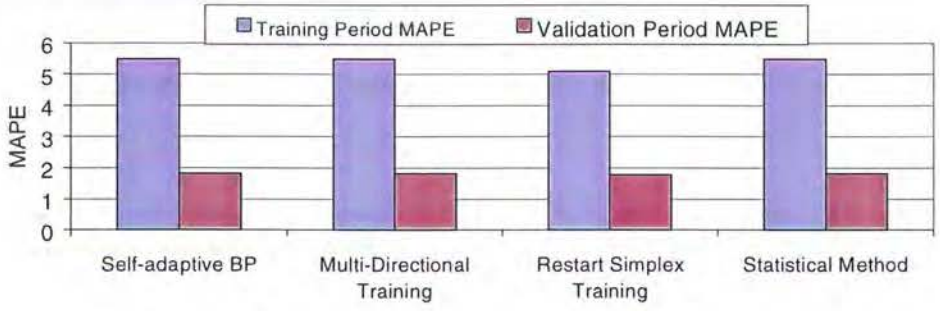


Figure 7.28 Performance measure in MAPE: Training period and Validation period

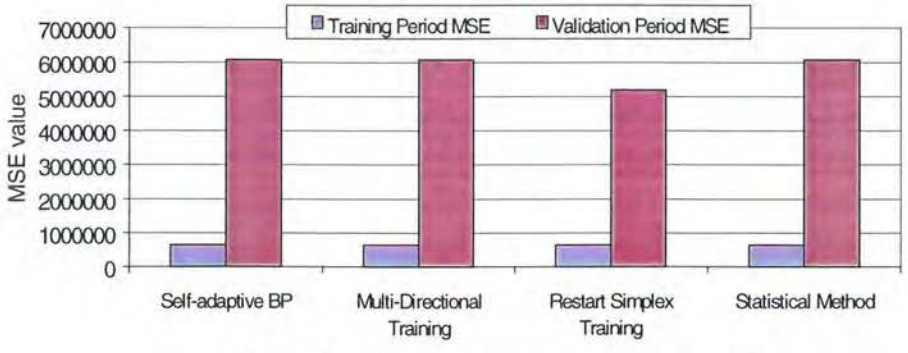


Figure 7.29 MSE comparison in training and validation period

Figure 7.30 Comparison in MPE in training and validation period

7.2 Simulation with Hotel Occupancy Rate Problem

The training performance with 7-4-1 ANN configuration to model the hotel occupancy rate is now considered. The random staring points are small in magnitude in all set of experiments.

7.2.1 Analysis with Standard Statistical Regression Method

The hotel occupancy rate problem is modeled as multivariate statistical regression problem. Table 7.12 summarizes the results obtained form the standard statistical regression method. The ANN is trained as multivariate statistical model. The modeling aim is to calibrate the problem as *spatial time series* model (Ahmed and Cross, 1999d).

Error measure	SSE	MSE	MPE	MAE	MAPE	SE	ME
Test Period	581.4	72.69	-6.3xE-7	7.94	0.2023	.0000148	.00000186
Validation Period	Analysis as	Multi-	variate	statistical	(calibration)	Problem	

Table 7.12 Results with standard statistical method

7.2.2 Analysis with Self-adaptive Back Propagation Method

Table 7.13 shows the self-adaptive training results against the standard BP training with the hotel occupancy rate problem. The self-adaptive and the standard back propagation training do not yield acceptable terminal function value in any experiments. The training terminates at a point, which is far from the minimum function value that is found in statistical regression method. This training problem appears to be difficult with the given set of initial condition and ANN configuration. The first order gradient based algorithms do not improve in function values and therefore the training performance with the standard BP and self-adaptive BP training is not considered in detail.

Expt #	Self-adaptive BP					Standard BP			
Hotel Occupancy	Epoch	Function evaluations	Gradient evaluations	Total Function evaluations	Function value	Epoch	Gradient evaluations	Total Function evaluations	Function value
1	5	118	192	310	195297	26	892	920	4.84E+07
2	3	67	128	195	195298	50	1984	2036	8.26E+07
3	3	72	128	200	195032	Not	Converging		
4	5	108	192	300	195283	96	3202	3300	3.60E+07
5	3	65	128	193	195266	61	2047	2110	376179
6	4	79	160	239	195066	65	2179	2246	379582
7	3	73	128	201	195297	44	1486	1532	1.19E+08
8	2	58	96	154	195294	33	1123	1158	206615
9	2	61	96	157	195301	13	463	478	197056
10	4	112	160	272	195314	17	595	614	195335
Mean	3.4	81.3	140.8	222.1	195244.8	45	1552.333	1599.333	31928301
Median	3	72.5	128	200.5	195295.5	44	1486	1532	379582
Standard Deviation	1.074968	22.57851	34.39897	55.73838	104.2484	26.4481	885.511	911.79	4394311
Range	3	60	96	156	282	83	2739	2822	1.19E+08
Minimum	2	58	96	154	195032	13	463	478	195335
Maximum	5	118	192	310	195314	96	3202	3300	1.19E+08

Table 7.13 Training performance with self-adaptive back propagation and standard back propagation training method

7.2.3 Analysis with Multi-directional Training Method

Table 7.14 displays the multi-directional training results with the hotel occupancy rate problem. Figure 7.31 shows the convergence of this training method, which also face difficulty in some experiments. The training method identifies 800.89 as the lowest function value and is higher than the value 581.5 found by regression method.

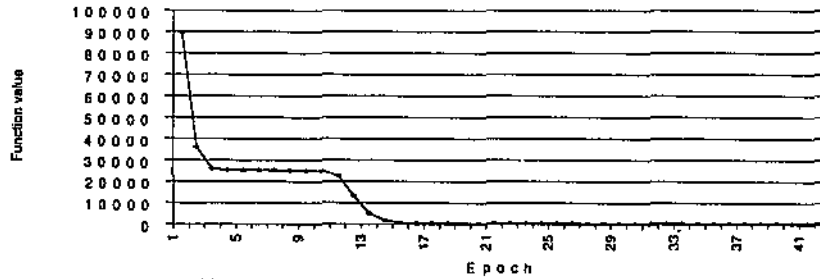


Figure 7.31 Function convergence with multi-directional training method

Expt #	Multi-Directional			Restart		
Hotel Occupancy	Epoch	Function evaluations	Function value	Epoch	Function evaluations	Function value
1	1493	1637617	800.89	146914	223386	584.4
2	386	360507	4614.7	1026183	1628766	581.5
3	874	771190	1184.36	295	912	195216
4	2501	2696055	914.76	861	2501	195319
5	45	38541	859.47	1020788	1627220	582
6	2501	2969178	23620.5	1087	2861	195289
7	2501	2734675	1883	510	1215	195310
8	2512	2609462	1162	3117	8778	195215
9	2501	2597691	3901.7	35980	140577	4371.65
10	2508	2353164	3857.8	256917	417930	3858
Mean	1782.2	1876808	4279.918	249265.2	405414.6	98632.66
Median	2501	2475428	1533.68	19548.5	74677.5	99793.33
Standard Deviation	1000.099	1097560	6949.564	416683.3	658410.2	101872.7
Range	2467	2930637	22819.61	1025888	1627854	194737.5
Minimum	45	38541	800.89	295	912	581.5
Maximum	2512	2969178	23620.5	1026183	1628766	195319

Table 7.14 Training performance with Multi-directional and restart training method

7.2.4 Analysis with Restart Training Method

Table 7.14 includes the restart training results with the hotel occupancy rate problem. Figure 7.32 shows the convergence of the restart training algorithm. Although the problem is difficult to train in some experiment, there is hope with the restart training that identifies a function value in accordance with the standard statistical regression method. Therefore, the restart training algorithm can provide an estimate that could be a reference point. Given a problem that is difficult to train with the gradient-based algorithm, the restart training method is able to reach a solution in a set of few experiments. The question of improvement over the standard statistical regression method has to be addressed from the modeling aspect and the configuration of an ANN. This altogether is a different issue.

As an example, a different 7-2-1 ANN structure produce results in function value 216.86, which is lower that the value 581.5. Figure 7.33 shows the last few steps before convergence to the minimum point. It takes 24,543 numbers of iterations and 42,113 numbers of function evaluations. Figure 7.34 shows the comparison in fit with different ANN configurations and multivariate statistical regression method.

The important aspect of this training method is not only the improvement over the statistical regression method but also its ability to train an ANN, which face difficulty with gradient based training methods. This demonstrates the merit of the algorithm. These experiments suggest that we have a training method at hand that is at least as good as

statistical model. Therefore, there is chance to improve upon the calibration performance based on further relevant research design (Badiru et al., 1998; Hurrion, 1998).

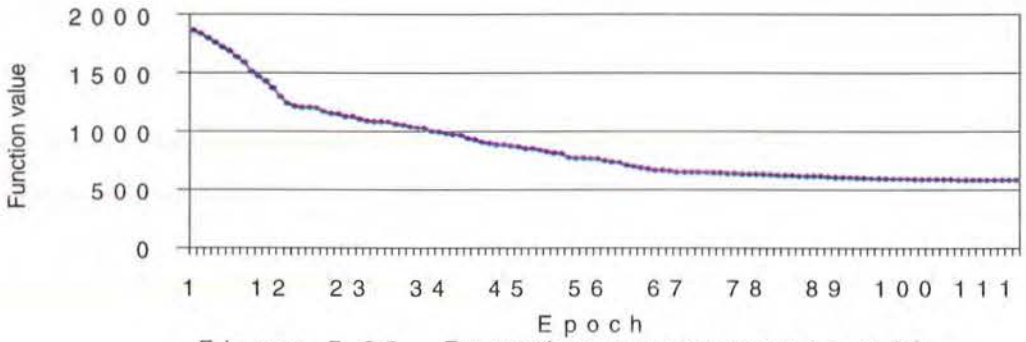


Figure 7.32 Function convergence with restart training method

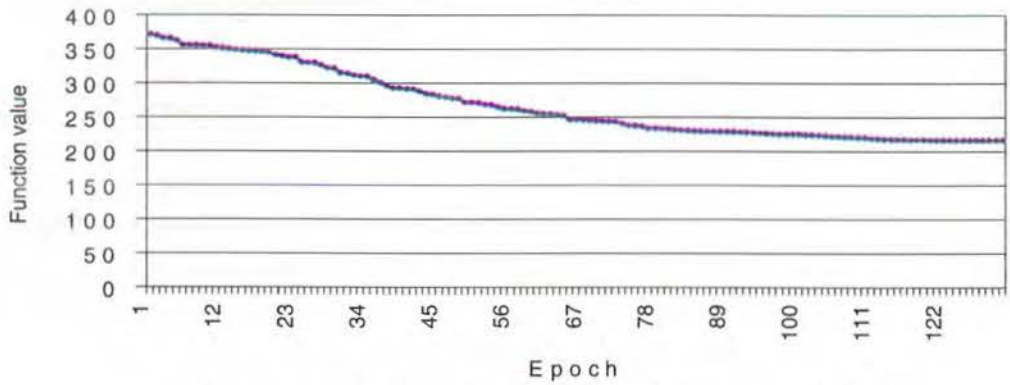


Figure 7.33 Convergence with resatrt (7-2-1: ANN) Hotel occupany rate

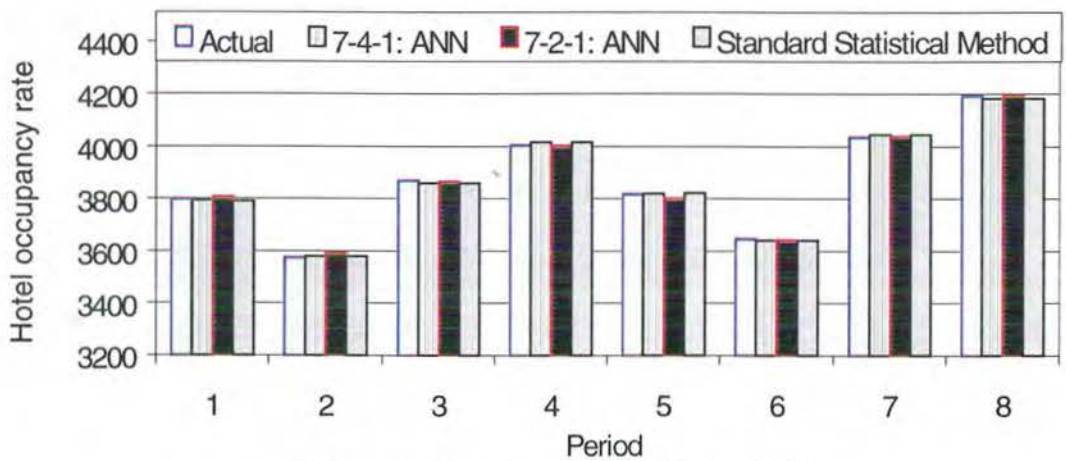


Figure 7.34 Comparison of fit with actual data

7.2.5 Comparison with Different Training Methods

Table 7.15 shows the comparison between the different training methods and Table 7.16 shows the performance in statistical measure in training and validation periods.

Hotel Occupancy Rate	Performance Measures	Self-adaptive BP	Multi-directional Training	Restart Training	Standard Back Propagation
Epoch	Average	3.4	1782	249265.2	45
	S. Deviation	1.07	1001	416683.3	26.4
	Max	5	2512	1026183	96
	Min	2	45	295	13
	Speed up	Not improving			Not improving
Function Evaluation	Average	222.1	1876808	405414.6	1599.3
	S. Deviation	55.7	1097560	658410.2	911.7
	Max	310	2969178	1628766	3300
	Min	154	38541	912	478
	Speed up	Not improving			Not improving
Function Value	Average	1952448	4279.92	98632.66	31928301
	S. Deviation	104.25	6949.6	101872.7	43943113
	Max	195314	23620.5	195319	1.19E8
	Min	195032	800.9	581.5	195335
	Gain	-	-	-	-

Table 7.15 Performance of different training methods

Hotel Occupancy	Performance Measures	Self-adaptive BP	Multi-Directional Training	Restart Simplex Training	Standard BP	Statistical Method
Training Period	MSE	Not converging	117.4	72.64	Not converging	72.69
	MAPE		0.238	0.201		0.2023
	MPE		-0.001	-2.4E-5		-6.3E-7
	R^2		.92	.99		0.998
	Epoch		1453	471442		-
	Func. Eval.		3213807	753447		-
Validation Period	MSE					
	MAPE		Analysis as	multi-variate	statistical	
	MPE		Calibration	problem		

Table 7.16 Statistical measures in training and validation period

7.3 Summary of Performance

Table 7.17 shows the ranking of the proposed methods based on three different criteria. In general no single training method performs well in all the test problems. Nevertheless, the

restart training method appears to perform well with all the test problems. It has the ability to identify better solutions in few experiments.

Kamarthi et al. (1999) report that the Polak-Ribiere method solves XOR problem in 14 and L-T letter recognition problem in 5 epochs. The result is included for comparison. This is however a second order training method. It uses second order approximation of gradient to generate search directions.

Ranking criteria	Problem	Standard Back propagation	Self-adaptive BP	Multi-Directional	Restart training	Jacobs (1988)	Salomon (1996)	Shamma (in Johansson et al. (1992)	Kamarthi et al. (1999)	Vogl et al. (1988)	P-R: (C.G) in Kamarthi et al. (1999)
Function value	5-5-1: Parity	4	1	2	3						
	L-T	6	2	1	3						
	Seasonal	4	3	2	1						
	Hotel	failed	failed	2 [5]	1 [5]						
	XOR	4	2	1	3						5
Function Evaluation	5-5-1: Parity	5	4	3	2			1 [5]			
	L-T	4	2	3	1						
	Seasonal	4	3	1	2						
	Hotel	failed	failed	2 [5]	1 [5]						
	XOR	4	1	3	2						
Epoch measure	5-5-1: Parity	4	2	1	3						
	L-T	7	3	2	4				6	5	1
	Seasonal	4	2	1	3						
	Hotel	failed	failed	1 [5]	2 [5]						
	XOR	7	3	1	6	5	4				2
	Legend	Best = 1		Worst = 7		[5] =	Cases	that	converge		

Table 7.17 Summery of performance of the proposed training methods and their ranking

7.4 Better Performance of a Training Method

Among the three new proposed training methods, the *restart* training method finds the best function value and trains an ANN efficiently in all the experiments. The average number of epoch to train 5-5-1 parity problem is 11310 against 17,354.4 with the standard back propagation method. The corresponding total numbers of function evaluations are 16583.2 and 538,072.4 respectively. In L-T letter recognition problem the restart training method takes 1316.7 epoch and 2046.2 number of function evaluations. The time series and multivariate statistical modeling problems are solved with this method. The other two methods face convergence difficulty in solving multivariate statistical modeling and seasonal time series problems. Considering the terminal function value as the ranking criteria, the restart training algorithm performs well in all the test problems.

7.5 Examples of Trained ANN with XOR, L-T, Time Series problems

Table 7.18a, 7.18b, 7.18c, 7.18d show the trained ANN for the L-T letter recognition task, XOR, hotel occupancy rate analysis and seasonal time series-forecasting problem. The network weights are obtained using the restart training method. Also note that the network weights are large in magnitude. The example of large weights and its sample calculations are shown in Chapter 6. The restart algorithm can produce trained network weights that are small as well as large in magnitudes. It is also possible to train the ANN with large initial weights as noted earlier.

Input Layer Weights (w_{in})

n→ / i↓	1	2
1	14.58	2.567
2	26.71	4.363
3	0.941	-2.266
4	9.379	4.440
5	2.956	-2.868
6	-1.432	-0.236
7	7.538	-3.298
8	8.0008	0.885
9	10.58	1.533

Output Layer Weights (w_{on})

n→ / o↓	1	2
1	-1.171	0.631

Table 7.18a Trained weights for 9-2-1 L-T letter recognition problem (Terminal function value 0.00051)

Input Layer Weights (w_{in}) Example 1

n→ / i↓	1	2
1	-0.0909	41.377
2	-6.3873	1.182

Output Layer Weights (w_{on})

n→ / o↓	1	2
1	-0.9389	1.5524

Input Layer Weights (w_{in})Example 2

1	-0.5351	0.2516
2	-0.3062	0.3462

Output Layer Weights (w_{on})

n→ / o↓	1	2
1	0.2374	-0.1391

Table 7.18b Trained weights for XOR problem (Terminal function values are 0.002294 & 0.0000597)

Input Layer Weights (w_{in})

n→ / i↓	1	2	3	4
1	-95086.48	-27632.5	7395.24	3337.29
2	230649.5	-8007.84	89596.86	-41892.25
3	76393.43	82448.94	1583.18	153949.3
4	4325.56	-339.32	-8732.1	-55280.92
5	225615.9	-80032.07	43121.35	-94798.13
6	-8056.88	-6009.94	-78023.65	-18870.78
7	-17834.13	12.21	-32164.06	19.331

Output Layer Weights (w_{on})

n→ / o↓	1	2	3	4
1	-396076.3	78362.39	259801.6	64293.97

Table 7.18c Trained weights for 7-4-1 hotel occupancy rate problem (Terminal function value 237.32)

Input Layer Weights (w_{ih})

n→ i↓	1	2	3	4	5
1	46915520000	-357467500	1826941000	4684613000	6967306
2	139778500	1238163	-15568.1	52121750	-34761900
3	87191900	2745410000	246719200	4563206000	357589400
4	-41204300	442934900	62475900	454709800	-966644
5	950983.4	-906982100	-3657912000	732146600	-238716900

Output Layer Weights (w_{ho})

n→ o↓	1	2	3	4	5
1	-80110760	3713776	66726680	-150285.2	226937.8

Table 7.18d Trained weights for 5-5-1 seasonal time series problem (Terminal function value 43787180)

7.6 Training Performance with Seasonal Time Series Problems

With the given ANN architecture and initial condition the forecast and multivariate analysis problem proved difficult to train, while the parity, XOR and the letter recognition problems are relatively easy to solve with the proposed methods.

The purpose of this research, however, is not to investigate further the forecast/multivariate problem with different ANN set up. This is a separate investigation by itself. The Research standardized the ANN architecture and initial starting points to study the convergence pattern. However, some examples of trained ANN results are shown in previous sections with different ANN configurations. To overcome the difficulty with the forecasting problems the following steps are suggested.

a.) Look for initial starting points that are appropriate for initialization. This can be done in a number of ways. One approach is to repeat the experiment with the optimized ANN and start form this point changing the network weights with 10 % variations. Repeat the experiments until a better ANN is obtained.

b.) Secondly, one can use other alternative method to identify the initial estimate of the forecast problem to initialize the ANN training.

c.) Thirdly, the ANN architecture can be varied both in the number of hidden layer neurons and different transfer functions (Ahmed and Cross; 1999a; Ahmed Cross, 1999b; Ahmed and Cross, 1999c). Also the number of layers in ANN can be varied to observe the performance of the neural network model. For example we have two different ANN configurations in Chapter 6 where the restart method is examined

against the Nelder and Mead (1965) method. In Section 7.6.4 and 7.7.4 we have two different ANN that produce better results.

7.7 Rectification of Slow Convergence

The interpolation search method developed in Chapter 4 prevents cycling of the algorithm on the error surface, oscillations during convergence and overshooting the local minimum. These characteristics are the limitations with the standard back propagation training (Kamarthi et al., 1999). The algorithms developed in this research improve on these drawbacks. The multi-directional training and restart training algorithm do not use gradient information to explore descent directions. The rectilinear direction and the direction from the centroid of a simplex along one of its worst vertex in opposite side provide the search directions. It is due to these search strategies; the training methods explore several local minimums with the possibility of improving the training performance. Consequently, the training methods developed in this research can be considered as a valuable and viable alternative to the existing training methods. The degree of improvement with the three proposed training methods in solving difficult problems is greater than the standard back propagation method.

7.8 The Rate of Convergence with Self-Adaptive BP Training Method

It is noticed that the time series problems face convergence difficulty. To observe the rate of convergence of the self-adaptive BP training method and the complex nature of the error surface, the eigenvalues of the Hessian matrix are computed. The error function is expanded in the neighborhood of a minimum w^* with $\nabla f(w^*)=0$ and neglecting higher order terms the following error equation is approximated as quadratic function:

$$f(w)=f(w^*)+\tfrac{1}{2}(w-w^*)^T H(w-w^*) \qquad (7.1)$$

where, $H=\dfrac{\partial^2}{\partial w_i \partial w_j}$, $(i=1,2,\ldots,m; j=1,2,\ldots,m)$ is the mxm Hessian matrix of $f(w)$.

The rate of convergence of the multi-directional training algorithm is observed with the quarterly seasonal time series problem as discussed in Chapter 3. The eigenvalues of the ANN time series are identified from the Hessian matrix H using standard eigenvalue computation method such as Jacobi method as explained in Schwarz (1973). Given $f: w \in E^m$ the algorithm theoretically converges to the unique minimum point w^*. Also during convergence, the following inequality exists (Luenberger, 1984):

$$\frac{\|f(w_{k+1})\|}{\|f(w_k)\|} \le \left(\frac{A-a}{A+a}\right)^2. \qquad (7.2)$$

The quantity, (A,a) are the largest and smallest eignevalues of the Hessian matrix. The Hessian matrix is positive definite implying that the values of $(A,a) > 0$.

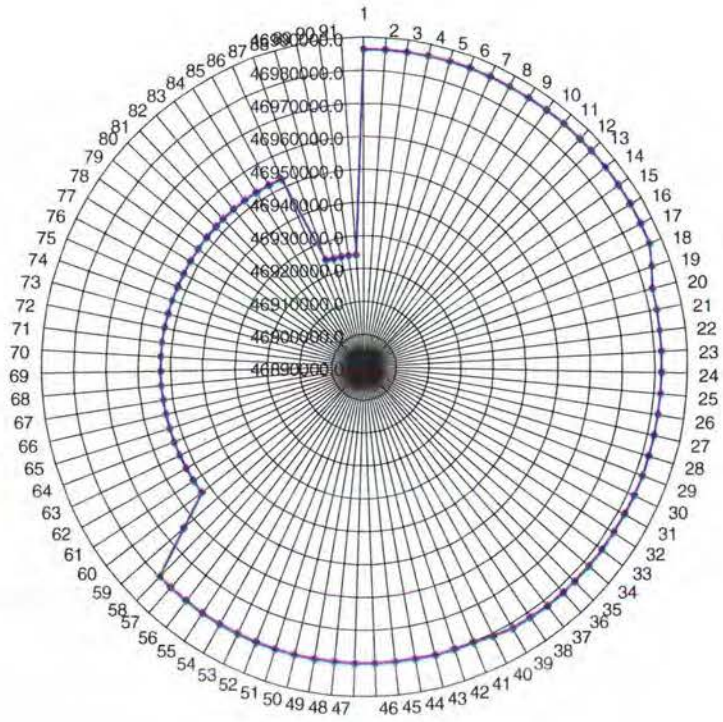


Figure 7.35 Function convergence with self-adaptive back propagation

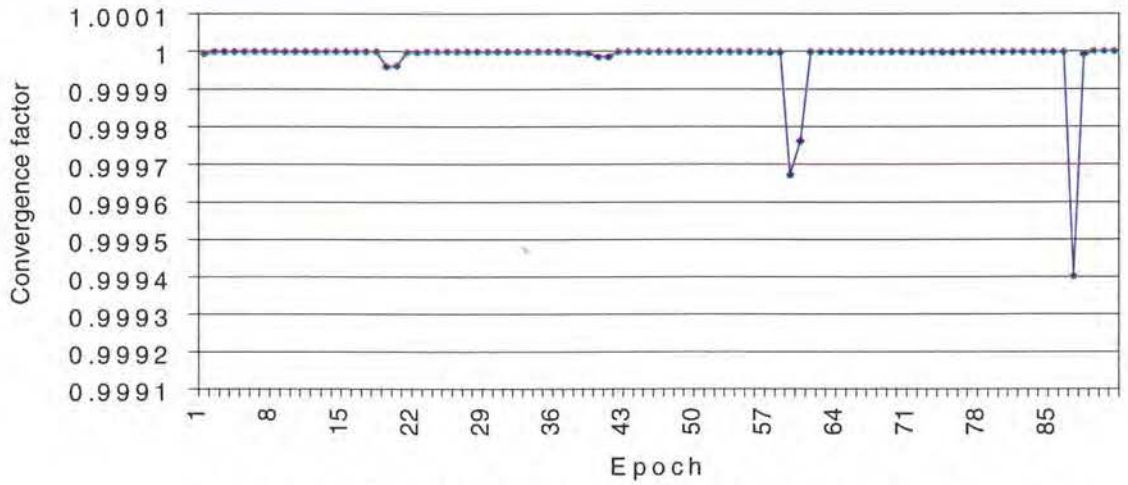


Figure 7.36 Rate of function convergence with self-adaptive BP (Convergence ratio)

If $A = a$, the contours are circular and the convergence is achieved with less effort. The contours are more eccentric if the eigenvalues are at a greater distance and the convergence of the training method will be slow. A single unfavorable eigenvalue from the set of m eigenvalues will cause trouble in convergence of steepest descent methods. The relation 7.2 suggests that the method converge linearly with a convergence ratio no greater than $[(A-a)/(A+a)]^2 = \left[(A/a-1)/(A/a+1)^2\right]$. The ratio A/a of the largest and lowest eigenvalue determines the condition number and it influences the convergence rate. The convergence ratio can be represented by $\left(\frac{r-1}{r+1}\right)^2 = \left(\frac{(A/a)-1}{(A/a)+1}\right)^2$. It is this factor by which the error function during training is reduced per iterations. The ratio A/a governs the convergence in proposed method. The Hessian matrix of the 5-5-1 ANN seasonal time model is shown in Table A.7 (Appendix A). The maximum and minimum eigenvalue of the matrix are 17.882 and 0.0001 respectively. The convergence ratio is 0.99997 and, therefore, convergence of the algorithm is bounded by this factor. The corresponding condition number is 179873. It suggests that the ANN error surface is skewed and forms ridges (Bishop, 1995; Jacobs, 1988; Bazaraa et al., 1993 and Luenberger, 1984). This will slow the convergence of the training method due to the fact that the algorithm needs extra efforts to climb up a valley and explore minimum in several flat surfaces as well as in narrow valleys. The last few iterations of the convergence are shown in Figure 7.35 and the rate of convergence is shown in Figure 7.36. Although the error surface has bad structure the self-adaptive BP training algorithm still can train the ANN.

The training algorithm generates a convergent sequence $\{f(w_k)\}$, which can be seen in Figure 7.35. Experimentally, the training algorithm after a finite number of iterations terminates according a ratio bounded by $\{(r-1)/(r+1)\}^2$. The term r is the condition number, which is defined as $r = A/a$. This result provides bound on the convergence to the proposed method and is a reference point in convergence analysis. If $r \to \infty$, this ratio approaches to 1 from below and the rate of convergence becomes slower.

Notice the change in convergence factor in Figure 7.36 in the interval 22 to 29, 36 to 43, 57 to 64 and 85 till the end of epoch. Compare the function convergence in Figure 7.35 in the same interval. It shows that the search moves from the higher contour surface of the error function to the next lower contour surface during iteration where there is change in convergence ratio. Also notice that when the convergence ratio is nearly equal to 1, the change in error function is not significant. It indicates that the training face difficulty in this region and the convergent sequence $\{f(w_k)\}$ is large due to the slow reduction in error function.

The theoretical and experimental convergence ratio is nearly equal and this investigation shows some of the difficulties inherent with ANN training.

7.9 Remarks

The performances of the three derivative free training methods are discussed in detail with the predefined experimental set up. The self-adaptive back propagation training and the multi-directional training algorithm train the ANN in parity and letter recognition problems efficiently. These two methods however face convergence difficulty with the seasonal time series and multivariate statistical analysis problems. The restart training method on the other hand is proved to be the best working tool with all type problems. It not only trains an ANN efficiently, but also improves the terminal function value. The experiment suggests that the restart training method improves over the standard statistical regression method and finds better local minimum in seasonal time series problem. It is shown that the training problem in seasonal forecasting is difficult to train due to the nature of the Hessian matrix. The condition number of the error function in 5-5-1 ANN seasonal time series problem is found of the order 179873. Therefore, the error surface is complex in geometry and function convergence is slow. It often terminates at a local minimum far from the desired local minimum. The experiments in this study show that the derivative free restart training method reduces the error function monotonically and converges to a solution that finds a solution better than the standard statistical regression method.

Chapter 8

Conclusions

8.1 Introduction

The major concern of this research has been the developments of three *new derivative free self-adaptive training algorithms* and the analysis of their convergence properties. The variable learning rates of the *self-adaptive back propagation are determined* dynamically in training iterations and hence, this approach eliminates the ad hoc method of parameter selection in ANN training. The search directions in self-adaptive BP are computed from a *controlled central difference approximation scheme* that utilizes the convergence characteristics of the error function. The proposed *multi-directional* training algorithm is *derivative free* and its training parameters are determined by the algorithm rather than by a user. An optimized *interpolation search* determines the momentum term to accelerate training performance. The *restart* training algorithm explores the geometry of the error surface along a direction from the worst vertex to the centroid of a simplex. The training algorithm finds better local minimum. All the training algorithms are faster than the standard back propagation algorithm.

The *restart* training algorithm performs better than others. It is a derivative free training method and has the advantage to train an error function that is discontinuous or ill conditioned or the corresponding Hessian matrix is singular. It successfully trains an ANN, where other training algorithms fails. The *multi-directional* training algorithm needs few epochs to train an ANN at the cost of function evaluations. These algorithms are designed such that the user needs minimum expertise to train an ANN.

8.1.1 Performance of the Training Algorithms

The convergence of the self-adaptive and derivative free training algorithms is proved. The experiment with the XOR problem suggests that the proposed *self-adaptive back propagation* training algorithm when compared with the standard back propagation algorithm shows improvement in the total number of function evaluations by the order of magnitude 16.74 and the relative efficiency in average epoch is 42.9. Similarly the proposed *multi-directional* training algorithm improves over the standard back

propagation algorithm by the factor 4.9 and 107.8 in function evaluation and epoch measure. The restart training algorithm is an order of a magnitude faster by 5.3 over the standard back propagation algorithm in function evaluations and 1.67 in epoch measure. Since all the training algorithms are *self-adaptive and derivative free*, there is no need to pre-optimize the training parameters with these algorithms. Few experiments can find better solutions in parity, letter recognition and business forecast application problems.

8.2 Significance of the Proposed Algorithms

The algorithms presented in Chapters 5 and 6 are important due to the fact that they can improve the local convergence in training. Tabu search is a new research area (Glover and Laguna, 1997; Sexton et al., 1998) and the local convergence of the *multi-directional* and *restart* simplex training algorithm can be improved further to investigate the training quality in ANN.

Business applications using ANN are increasing due to its ability to serve as flexible form of estimators. In order to achieve a better estimate, an optimized ANN is desirable. Many applications are currently using some variant of the back propagation algorithm. An ANN that is not properly trained often performs poorly when forecasting out of sample (Saxton et al., 1998). A possible solution to this local convergence dilemma is the tabu search method (Saxton et al., 1998). In forecasting problem the extrapolation and the generalization capability of the proposed algorithm is as good as the statistical regression methods and the *restart* algorithm improves over the standard statistical regression methods. The proposed algorithms are efficient in solving parity and letter recognition problems. Training a multivariate statistical problem with small data set is difficult.

8.3 Theoretical Implications

The convergence analysis of the training algorithm is provided. The *multi-directional* training algorithm solves forecast problems to produce results that are identical to the regression method. The mean square error is the same. This implies that the generalization capability is as good as the standard regression method. However, different ANN structure can be tested to see if there is further improvement over the standard statistical regression method. The *restart* training algorithm, on the other hand, improves the local minimum and finds better mean squared error value than the standard statistical regression method without scaling data. The *self-adaptive back propagation* training algorithm performs closely in identifying the mean squared error against the standard regression method. A modeler, therefore, needs minimum efforts to train a problem.

The convergence difficulty of the ANN computation is highlighted with the analysis of a time series problem in Chapter 7. The convergence becomes slow due to the high

condition number in the Hessian matrix of the error function. Such incidents are not rare in ANN computations as Ill conditioning in error function arising due to high condition number causes premature termination of the algorithm (Jacobs, 1988; Salomon et al., 1995). The proposed algorithms developed in Chapter 5 and 6 can resolve some of the difficulties.

The training with a multivariate statistical model with a small data set in some experiments converges to a bad local minimum. The reason can be attributed to the small data set and the learning is insufficient. The error surface is complex and the training terminates at a point far from the minimum. However, the *restart* training algorithm finds a solution close to the regression method in mean square error measure. The remedy is to repeat the experiment from the locally optimized point or vary the ANN configuration to identify better local minimum. It is shown that a different ANN configuration identifies a better solution than the standard regression method both in training and validation period.

The proposed algorithms are able to avoid local minimum. It is observed from the terminal function values, which are often significantly low. This suggests that the algorithms find an improved local minimum.

It should be noted that the raw data for the forecasting and statistical regression problem are used without any transformation. The ANN configuration and data have been standardized. It gives the chance to test the algorithm's ability to model a problem without much intervention by the modeler or user.

8.4 Improved Performance and Convergence Without Oscillations

All the proposed algorithms are free from severe oscillations while the error function convergences to a local minimum. This is one of the salient features of the proposed algorithms. The training algorithms adjust the learning rate parameters according to the geometry of the contour surface. Hence, the *dynamic self-adaptive algorithm* attains a desired degree of accuracy in a finite number of steps. The *restart* training algorithm is able to train an ANN where other training algorithms failed. The training successfully converges to a minimum point without oscillations in most of the experiments.

8.5 Unique Properties of the Training Algorithms

The descent directions in self-adaptive BP training algorithm is derived from the central difference gradient approximation scheme, which uses convergence property of the algorithm. In contrast, the training algorithm developed in Chapter 5 and 6 do not use the gradient information. The user defined learning rate and momentum parameters are not required. Hence, these developments are important for the case when:

a.) the error function is discontinuous;
b.) the derivative of the error function is not explicit (Griewank, 1994; Griewank and Corliss, 1991);
c.) an application needs derivative free training algorithm (Conn et al., 1997).;
d.) the pre-optimization of the learning rate and the momentum term are difficult, inconvenient and time consuming;
e.) the improvement in local minimum is desirable and
f.) only few experiments are needed to find a better optimized ANN.

The *restart* training algorithm proved to be a useful algorithm that has the ability to train all the test problems. Only few experiments are needed to identify acceptable trained ANN. The quality of terminal function value is low and therefore the local minimum is avoided with this training algorithm.

When gradient based training algorithm faces difficulty, the *restart* and *multi-directional* training algorithm improves the training performance. These two training algorithms are reliable to work with when the nature of the solution is not known. For new test problems these two algorithms provide solutions that can be compared to measure the quality of a trained ANN.

8.6 Practical Benefit Over Second Order Training Algorithms

Second order training algorithms have superior convergence properties. It is discussed in Chapter 2. However, the storage requirement for the conjugate gradient method is about four times than that of standard back propagation algorithm (Kamarthi et al., 1999). Theoretically, the computational complexity of the Hessian matrix is of the order m^3 (Chen at al., 1999). The computation per cycle significantly increases due to the line search, which determines the appropriate learning rate in second order training methods. In conjugate gradient algorithm the training often converges to a bad local minimum from which the conjugate gradient method cannot escape (Kamarthi et al., 1999). As a result, it will impair the generalization ability of the network (Towsey, et al., 1995). This is a limitation of the conjugate gradient method (Kamarthi et al., 1999). Newton types training algorithms require a starting point close to the minimum point for convergence. Any arbitrary starting point does not necessarily provide convergence. The second order training method also suffers from ill conditioning due to the Hessian approximation and approximation to the quadratic function itself. The derivative free training algorithms require less storage as compared to the gradient-based training algorithms. They find better local minimum compared to statistical regression method and standard BP training and are reliable when the error function is unfavorable for training with gradient-based training algorithms.

8.7 Future Research Directions

Three new *self-adaptive derivative free* training algorithms and the related software are developed. These training algorithms solve different classes of problems and find better solution against the standard back propagation algorithm and, therefore, these algorithms provide better quality decisions. There are a number of areas where further research could improve the training performance and modeling strategies. Some of the issues are discussed below.

8.7.1 Modeling Strategy

It is noticed that the multivariate statistical problem is difficult to train with the proposed training algorithms. Therefore, it is important to investigate the underlying modeling structure in detail. Zhang, Patuwo and Hu (1998) provide review of time series forecasting using ANN and discuss model selection strategy. The simple approach is to vary the ANN structure in number of hidden neurons and number of hidden layers. The examples of variable hidden nodes to model seasonal time series problem are shown in Chapter 6, Ahmed (1999b) and Park et al., (1991). The other aspect is to look into the parameterization methodology in input layer neurons. The initial staring points are extremely sensitive to this class of problem and therefore a methodology, which can provide good initial estimate, is perhaps appropriate (Badiru and Sieger, 1998). One can also study the performance of an algorithm with different data scaling methods both to see its convergence and generalization properties.

Since the training algorithms are derivative free, it is easy to accommodate a variety of ANN configurations. The number of layers can be easily increased or any other special feed forward ANN structure can be trained using these algorithms.

8.7.2 Algorithmic Aspects

The proposed *multi-directional* derivative free training algorithm possesses important attributes. It is a class of training algorithm, which can be improved as a global training algorithm. Global training is a relatively new research area in ANN computation that works on the theory of tabu search proposed by Glover and Laguna (1997). One suggestion is to create a tabu list (Glover and Laguna, 1997; Sexton et al., 1998 and Shang et al., 1996) and explore several local minimums following the tabu list as the search progresses. The other approach is to generate stochastic descent directions following some specific distributions that may be beneficial to some training problem (Zhang and Xu, 1999).

The *restart* training algorithm is performing well but the training difficulty is noticed in multi-variate statistical problem. The search coefficients in the *restart* training algorithm are optimized by simulation experiment. It is worth investigating the performance of the training algorithm with exact optimization method to identify the *restart* search coefficients.

In *self-adaptive back propagation* training algorithm, a central difference gradient approximation computes the descent directions implicitly. Recently a new approach has emerged in automatic differentiation tools (Griewank and Corliss, 1991; Griewank, 1994 and Coleman and Jonsson, 1999). There is potential in directing research in these aspects.

8.7.3 Problem Specific Training Performance

It is also important to find some training problem, which has the corresponding error functions that are ill conditioned or cannot be differentiated explicitly or has singular Hessian matrix. The performance of the proposed algorithms with other different gradient based training algorithms in solving these problems needs attention.

8.7.4 Forecasting with ANN

The science of forecasting with ANN needs more attentions according to Adya and Collopy (1997). They study 48 published neural network applications and found only 11 are both efficiently implemented and validated. The validation can also be addressed in the context of bootstrap method (Diaconis and Effron, 1983; Effron and Tibshirani, 1986; Flachaire, 1999 and Gunter, 1991).

The method or dynamics of introducing chaos into the system other than an auto regressive process (Box et al., 1994; Kantz et al., 1977; Conway et al., 1998; Chakraborty et al., 1992) needs attention. The art of meta modeling with ANN is beginning to emerge (Kilmer et al., 1999; Hurrion, 1998; Anjum et al., 1997 and McHaney, 1997). It is a science that combines ANN computations, discrete simulation and multivariate statistics as design of experiments. Such approach can be introduced in time series applications as model selection strategy.

8.7.5 Market Research Applications

Market research and conjoint analysis is another emerging trend and have far-reaching implications. As the data processing and parameter estimation technology evolves, implementing more sophisticated techniques becomes easier for analysis (Hanssens et al., 1993; Green et al., 1993; Van Wezel et al., 1995; Hruschka and Natter, 1999). Many service organizations such as banks, credit-card companies and airlines use their extensive

customer databases as strategic assets to develop cross-selling and other customer loyalty-focused marketing strategies. To understand the driving force of purchasing behavior and to predict the likely outcomes of alternative marketing strategies, the ANN application in market research as a statistical tool remains an open issue.

8.7.6 Applications and Case Study

The applications of training algorithms can be further extended to biometric applications, fraud detection in commercial enterprises, e-commerce, data warehousing, data mining, health care (Goss and Ramchandani, 1998) and inventory control applications (Sieger and Badiru, 1993), due to the algorithm's ability to identify good local minimum.

8.8 Concluding Remarks

The first order BP training algorithm requires gradient information and a user defined learning rate and momentum parameter to train an ANN. The second order BP training algorithms depend on the first and second derivatives. This research develops three distinct *derivative free* ANN training algorithms. All the proposed algorithms are *dynamically self-adaptive* and no user defined training parameters are required to train an ANN. These parameters are determined optimally during training to provide maximum possible descent to the ANN error function. As a result fast convergence is achieved without oscillations. All the algorithms have been developed as ANN training software using FORTRAN programming language containing about 15,000 instructions in total.

These algorithms successfully train character recognition, parity and forecasting problems. Extensive training experiments suggest that the proposed *self-adaptive derivative free* algorithms are faster than the standard first order BP algorithm in number of epoch and numbers of function evaluations. The proposed algorithms also improve in terminal function value indicating that the improved local minimum is found. Hence, better training results are achieved. It is also found that the derivative free *restart* training method improves over the statistical regression method in training forecasting problems.

Further research is necessary to test these algorithms in problems where the Hessian matrix of the corresponding ANN error function is ill conditioned, error function is discontinuous and derivative information is difficult to obtain.

Reference

1 ABS (1998), Jan, # 8301.0., Australian Bureau of Statistics, Electricity, Manufacturing Production.

2 ABS (1998), Sept, # 8635.0., Australian Bureau of Statistics, Tourist Accommodation Data.

3 ABS (1999), Sept, # 6401.0., Australian Bureau of Statistics, Consumer Price Index.

4 ABS (1999), Sept, # 8635.0., Australian Bureau of Statistics, Australian Economic Indicators.

5 ABS (1999), Sept, # 8635.0., Australian Bureau of Statistics, Consumer Price Index.

6 Adya and Collopy (1997). How Effective are Neural Networks at Forecasting and Prediction? A Review and Evaluation, *Journal of Forecasting*, Forthcoming.

7 Ahmed, S. and Cross, J., (1999a). Derivative Free Training in Seasonal Time Series Using Grid Search. *MOSIM99, International Congress on modeling and Simulation Proceedings, vol. 4*, Editors Les Oxley and Frank Scrimgeour, 1057-1062.

8 Ahmed, S. and Cross, J., (1999b). Neural Network Model to Study Seasonal Time Series Data Using Derivative Free Search Method, *26 th. International conference on computers and industrial engineering*, Melbourne, Australia., 1999, Editor E. Shayan, 673-679.

9 Ahmed, S. and Cross, J., (1999c). Modeling Accommodation Demand in Australian Hotel Industry. Submitted for publication.

10 Ahmed, S. and Cross, J., (1999d). A Tourist Growth Model to Predict Accommodation Nights Spent in Australian Hotel Industry. *SIRC 99*, Dunedin, New Zealand December, 1999.

11 Ahmed, S., (1999a). A Derivative Free Optimization Method Using Improved Simplex Search in Higher Dimension, *The 15th National Conference of Australian Society for Operations Research, Gold Coast, Editor: E. Kozan, 97-104.*

12 Ahmed, S., (1999b). Derivative Free Optimization in Higher Dimension, accepted for publication in *International Transaction in Operations Research*, 2000.

13 Ahmed, S., and Cross, J., (2000). Convergence in Artificial Neural Network without Learning Parameter, Second *International Computer Science Conventions on Neural Computations, 2000*, Berlin, Germany

14 Ahmed, S., Cross, J., and Bouzerdoum, A. (2000a). A New Self-Adaptive Back Propagation Training Method, *Second International Computer Science Conventions on Neural Computations, 2000*, Berlin, Germany.

15 Ahmed, S., Cross, J., and Bouzerdoum, A. (2000b). Derivative Free and Multi-directional Training to Model Seasonal Time Series, accepted in *Decision Support Systems* Journal, (under review).

16 Ahmed, S., Cross, J., and Bouzerdoum, A. (2000c). Performance Analysis of a new multi-directional training algorithm for Feed-Forward Neural Network, Accepted for publication in

World Neural Network Journal, 2000.

17 Al-Sultan, K.S. and Al-Fawzan, M.A., (1997). A Tabu Search Hooke and Jeeves Algorithm for Unconstrained Optimization. *European Journal of Operations Research, 103*, 198-208.

19 Anjum, M.F., Tasadduq, I., and Al-Sultan, K., (1997). Response Surface Methodology: A Neural Network Approach, *European Journal of Operations Research, 101*, 65-73.

20 Ampaziz, N., Perantonis, S.J., and Taylor, J.G., (1999). Dynamics of Multilayer Networks in the Vicinity of Temporary Minima, *Neural Networks, 12*, 43-58.

21 Archer, N., Wang, S., (1993). Application of Back Propagation Neural Network Algorithm With Monotonicity Constraints for Two-Group Classification Problems. *Decision Science, 24, (1)*, 60-75.

22 Atiya, A.F., El-Shoura, S.M., Shaheen, S.I., and El-Sherif, M.S., (1999). A Comparison Between Neural-Network Forecasting Techniques-Case Study: River Flow Forecasting, *IEEE Transactions on Neural Networks, 10 (2)*, 402-409.

23 Azoff, E. M., (1994). *Neural Network Time Series Forecasting of Financial Markets*, John Wiley & Sons, Chichester, UK.

24 Barnard, E., (1992). Optimization for Training Neural Nets. *IEEE Transactions on Neural Networks, 3 (2)*, 232-240.

25 Battiti, R., (1989). Accelerated Backpropagation Learning: two Optimization Methods. *Complex Systems. 3*, 331-342.

26 Bazaraa, Mokhter. S., Sherali, Hanif. D. and Shetty, C.M., (1993). *Nonlinear Programming Theory and Algorithm., 2nd Ed.* John Wiley & Sons, Inc., NY, USA.

27 Becker, S. and Cun, Y.L. (1989). Improving the Convergence of Back-Propagation Learning With Second Order Methods. *Proceedings 1988 Connectionist Models Summer School*, Edited, Morgan Kaufman, 29-37.

28 Bishop, C.M., (1995). *Neural Networks for Pattern Recognition*, Oxford University Press, New York, USA.

29 Box, G. E. P., Jenkins, G. M., and Reinsel, G. C. (1994). *Time Series Analysis forecasting and Control*, Prentice Hall, New Jersey

30 Brad, V., (1968). On A Numerical Instability of Davidon-Like Methods, *Mathematics of Computation, 22*, 665-666.

31 Broyden, C.G., (1967). Quasi-Newton Methods and Their Application to Function Minimization, *Mathematics of Computation, 21*, 368-381.

32 Bryson, A.E. and Ho, Y.C., (1969). *Applied Optimal Control*, Bailsdell, New York, NY.

33 Cater, J. P., (1987). Successfully Using Peak Learning Rates of 10 (and Greater) in Back-Propagation Networks with the Heuristic Learning Algorithm, *Proceedings of the IEEE International Conference on Neural Networks, 2* San Francisco, CA, 645-651.

34 Chakaraborty, K., Mehrotra, K., Mohan, C.K., and Ranka, S., (1992). Forecasting the Behavior of Multivariate Time Series Using Neural Networks, *Neural Networks, 5*, 961-970.

35 Chan, L.W. and Fallaside, F., (1987). An Adaptive Training Algorithm for Back-Propagation Networks. *Computer Speech and Language, 2*, 205-218.

36 Chan, L.-W., and Shatin., N.T., (1990). Efficiency of Different Learning Algorithms of the Back-Propagation Network: In *Proceedings of IEEE Region 10 Conference on Computer and Communication Systems*, 23-27.

37 Chauvin, Y., (1990). Dynamical Behavior of Constrained Back-Propagation Networks. In Touretzky, D.S., (Ed.), *Advances in Neural Information Processing Systems Vol. 2*. Morgan Kaufmann, San Mateo, CA, 642-649.

38 Chen, J.R., Mars, P., (1990). Step Size Variation Methods for Accelerating the Back Propagation Algorithm. *Proceeding of the International Joint Conference on Neural Networks, Washington, DC, 1*, 601-604.

39 Chen, K., Xu, L., and Chi, H. (1999). Improved Learning Algorithms For Mixture of Experts in Multi-class Classification. *Neural Networks, 12*, 1229-1252.

40 Codrington, C. W. and Mohandes, M., (1994). Projection Based Methods for Step Size Adaptation and Their Application to the Training of Feed Forward Artificial Neural Networks, *Proceedings of the IEEE International Conference on Neural Networks, 1* Orlando, FL; 72-77.

41 Coleman, T. F., and Jonsson, G. F., (1999). The Efficient Computation of Structured Gradients Using Automatic Differentiations, *Siam Journal on Scientific Computation, 20 (4)*, 1430-1437.

42 Conn, A.R., Scheinberg, K., Toint, P. L. (1997). Recent Progress in Unconstrained Nonlinear Optimization Without Derivatives. *Mathematical Programming, 79*, 397-414.

43 Conway, A. J., Macpherson, K. P., and Brown, J. C., (1998). Delayed Time Series Predictions with Neural Networks, *Neurocomputing, 18*, 81-89.

44 Corana, A., Marchesi, M., Martini, C. and Ridella, S., (1987). Minimizing Multimodal Functions of Continuous Variables With the Simulated Annealing Algorithm, *ACM Transactions Mathematical Software, 13, (3)*, 262-280.

45 Curry, B., and Peel, M.J., (1998). Neural Networks and Business Forecasting: An Application to Cross-Sectional Audit Fee Data. *IJCM, Vol. 8, No.2*, 94-120.

46 Davidon, W.C., (1959). Variable Metric Method for Minimization, *AEC Research Development Report*, ANL-5990.

47 Deco, G. and Schurmann, B., (1994). Recurrent Neural Networks Capture the Dynamical Invariance of Chaotic Time Series, *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Science, 77-A (11)*, 1840-1845.

48 Deco, G., and Obradovic, D., (1996). *An Information Theoretic Approach to Neural Computing*, Springer-Verlag, New York, USA.

49 Dennis, J, E., and Schnabel, R.B., (1983). *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Englewood Cliffs, NJ: Prentice Hall.

50 Dennis, J. E., and Torczon, V., (1991). Direct Search Methods on Parallel machines., *SIAM Journal on Optimization, 1*, 448-474.

51 Diaconis, P., and Effron B., (1983). Computer-Intensive Methods in Statistics, *Scientific American, 248*, 116-130.

52 Effron, B., (1982). The Jackknife, the Bootstrap and Other Resampling Plans. *Society for Industrial and Applied Mathematics.*, Philadelphia, PA.

53 Efron, B., Tibshirani, R., (1986). Bootstrap Methods for Standard Errors, Confidence Intervals, and Other Measures for Statistical Accuracy., *Statistical Science, 1*, 54-77.

54 Efron, B., and Tibshirani, R., (1993). *An Introduction to Bootstrap*, London, Chapman and Hall.

55 Eisenpress, H., and Greenstadt, J., (1966). The Estimation of Non-linear Economic System; *Econometrica. V 34*, 851-861.

56 Fahlman, S.E., (1988). Faster-learning variations on back-propagation: an empirical study, in Touretzky, D., Hinton, G. and Sejnowski, T. (Eds), *Proceedings of the Connectionist Models Summer School*, Morgan Kaufmann, San Mateo, CA , 38-51.

57 Fahlman, S.E., and Lebiere, C., (1990). The Cascade-Correlation Learning Architecture. In Touretzky (Eds), *Advances in Neural Information Processing Systems, Vol. 2*, Morgan Kaufmann, San Mateo, CA , 524-532.

58 Flachaire, E., (1999). A Better Way to Bootstrap Pairs. *Economic Letters, 64*, 257-262.

59 Flatcher, R., and Reeves, C. (1964). Function Minimization by Conjugate Gradients. *Computer Journal, 7*, 149-154.

60 Fletcher, R., and Powell, M., (1963). A Rapidly Convergent Descent Method for Minimization, *Computer Journal, 6*, 163-168.

61 Franzini, M. A., (1987). Speech Recognition With Back Propagation. *Proceedings of the IEEE Ninth Annual Conference of the Engineering in Medicine and Biology Society, Boston, MA, 9*, 1702-1703.

62 Fukuoka, Y., Matsuki, H., Minamitani, H., Ishida, A., (1998). A Modified Back-Propagation Method to Avoid Flase Local Minima. *Neural Networks, 11*, 1059-1072.

63 Glover, F., and Laguna, M., (1997). *Tabu Search.*, Kluwer Academic Publishers, Norwell, MA., USA.

64 Goldfrab, D., (1969). Sufficient Conditions for the Convergence of a Variable Metric Algorithm, Chapter 18, in *Optimization*, R. Fletcher (Ed.), Academic Press, London.

65 Goldfrab, D., (1970). A Family of Variable Metric Methods Derived by Variational Means, *Mathematics of Computation, 24*, 23-26.

66 Goss, E. P., and Ramchandani, H., (1998). Survival Prediction in the Intensive Care Unit: A Comparison of Neural Networks and Binary Logit Regression., *Socio-Economic Planning and Science, vol. 32, no.3*, 189-198.

67 Green, P. E., and Krieger, A.M., (1993). Conjoint Analysis with Product Positioning Applications, in *Handbook in Operations research and Management Science, Vol 5, Marketing*, Editors, Nemhauser, G. L., and Rinnooy Kan, A.H.G., Edited by Eliashberg, j. and Lilien, G. L., North Holland, Amsterdam, 1993.

68 Griewank, A. (1994). Computational Differentiation and Optimization, in J.R. Birge and K. G. Murty, Eds., *Mathematical Programming: State of the Art.*, The University of Michigan, Ann Arbor, MI, 102-131.

69 Griewank, A. and Corliss, G., (1991). Automatic Differentiation of Algorithms, *SIAM*, Philadelphia, PA. USA.

70 Gunter, B., (1991). Bootstrapping: How to Make Something from Almost Nothing and Get Statistically Valid Answers. Part I: Brave New World. *Quality Progress, 24*, December, 97-103.

71 Gupta, N., and Mehra, R., (1974). Computational Aspect of Maximum Likelihood Estimation and Reduction in Sensitivity Function Calculations., *IEEE Transactions on Automation Control, Vol. 19*, 774-783.

72 Hanssens, D. M., and Parsons, L. J. (1993). Econometric and Time-Series Market Response Models, in *Handbook in Operations research and Management Science, Vol 5, Marketing*, Editors, Nemhauser, G.L., and Rinnooy Kan, A.H.G., Edited by Eliashberg, J,. and Lilien, G.L., North Holland, Amsterdam, 1993.

73 Haykin, Simon., (1994). *Neural Networks: A Comprehensive Foundation.*, Macmillan College Publishing, 1994.

74 Hebb, D. O., (1949). *The Organization of Behavior.*, New York: John Wiley, USA.

75 Hecht-Nielsen, R., (1990). *Neuro Computing.*, Addison-Wesley Publishing Company, Reading, Massachusetts, USA.

76 Hestenes, M. R. and Stiefel, E.L. (1952). Methods of Conjugate Gradients for Solving Linear Systems. *Journal of Research of the National Bureau of Standards, 49, 6*, 409-436.

77 Hinton, G. E., (1987). Learning Translation Invariant Recognition in Massively Parallel Networks. In De Bakker, J.W., Nijman, A.J., and Treleaven, P.C. (Eds.), *Proceedings PARLE Conference on Parallel Architectures and Languages Europe*, 1-13. Berlin: Springer-Verlag.

78 Holt, M. J., Semmani, S., (1990). Convergence of Back Propagation in Neural Networks Using a Log-Likelihood Cost Function. *Electron Letters, 26*, 1964-1965.

79 Hooke, R. and Jeeves, T.A., (1961). Direct search Solution of Numerical and Statistical Problems. *J. Association Computer Machinery, 8*, 212-229.

80 Hopfield, J.J., (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences, Vol. 79*, 2554-15888.

81 Hopfield, J.J., (1984). Neurons with graded responses have collective computational properties like those of two stated neurons. *Proceedings of the National Academy of Sciences, USA, Vol. 81*, 3088-3092.

82 Hruschka, H., and Natter, M., (1999). Comparing Performance of Feed Forward Neural Nets and K-means for Cluster-Based Market Segmentation, *European Journal of Operations Research, 114*, 346-353.

83 Hsiung, J.T., Suewatanakul, W., Himmelblau, D.M., (1990). Should Backpropagation Be Replaced by More Effective Optimization Algorithms? *Proceedings of the Internal Joint Conference on Neural Networks (IJCNN), 7*, 353-356.

84 Hush, D.R. and Salas, J.M. (1988). Improving the Learning Rate of the Back Propagation With the Gradient Re-use Algorithm. *IEEE International Conference on Neural Networks, 1*, San Diego, CA, 441-447.

85 Hush, D.R., Horne, B. and Salas, J.M. (1992). Error Surfaces for Multilayer. *IEEE Transactions on Systems, Man and Cybernetics, 22*, 1152-1161.

86 Jacobs, R.A. (1988). Increased Rate of Convergence Through Learning Rate Adaptation. *Neural Networks, 1*, 295-307.

87 Jang, J.-S.R., Sun, C.-T., Mizutani, E., (1997). *Neuro-Fuzzy and Soft Computing.*, Prentice Hall International, Inc, NJ, USA.

88 Johansson, E.M., Dowla, F.U. and Goodman, D.M. (1992). Backpropagation Learning For Multilayer Feed-Forward Neural Networks Using the Conjugate Gradient Method. *International Journal of Neural Systems, vol 2, no 4*, 291-301.

89 Kamarthi, S.V., and Pittner, S., (1999). Accelerating Neural Network Training Using Weight Extrapolations, *Neural Networks, 12*, 1285-1299.

90 Kantz, H. and Schreiber T. (1997). *Nonlinear Time Series Analysis.*, Cambridge University press, Cambridge, UK.

91 Kelley, C.T., (1999). Detection and Remediation of Stagnation in the Nelder-Mead Algorithm Using a Sufficient Decrease Condition, *SIAM J. on Control and Optimization, Vol. 10, No.1*, 43-55.

92 Kent, T. (1982). Robust Properties of Likelihood Ratio Test. *Biometrika, 69*, 19-27.

93 Kiefer, J., (1953). Sequential Minimax search for a Maximum. *Proceedings of the American mathematical Society, 4*, 502-506.

94 Kilmer, R. K., Smith, A. E., and Shuman, L. J., (1999). Computing Confidence Intervals for Stochastic Simulation Using Neural Network Metamodels, *Computers and Industrial Engineering, 36*, 391-407.

95 Kolen, J.F. and Pollack, J.B. (1991). Back Propagation is Sensitive to Initial Conditions. In R.P.Lippman, J.E. Moody, and D.S. Tpuretzky, (eds.), *Advances in Neural Information Processing Systems, 3*, 860-867, San Mateo, CA: Morgan Kaufmann.

96 Kolmogorov, A.N., (1957). On the Representation of continuous functions of many variables by superposition of continuous functions of one variable and addition., *Dokl. Akad. Nauk., USSR, 114*, 953-956.

97 Kowalik, J., Osborne, M.R., (1968). *Methods for Unconstrained Optimization Problems.*, American Elsevier, New York, USA.

98 Kramer, A.H. and Sangiovanni-Vincentelli, A. (1989). Efficient Parallel Learning Algorithms For Neural Networks. In *Advances in Neural Information Processing Systems 1*, Ed. D.S.Touretzky, San Mateo: Morgan Kaufmann, 40-48.

99 Krzyzak, A., Dai, W., and Suen, C.Y., (1990). Classification of Large set of Handwritten Characters Using Modified Back Propagation Model. *Proceedings of the International Joint Conference on Neural Networks, Vol.III, 225-232.* Piscataway, NJ: IEEE Press.

100 Lagarias, J.C., Reeds, J.A., Wright, M.H., and Wright, P.E., (1998). Convergence Properties of the Nelder-Mead Simplex Method in Low Dimensions, *SIAM J. on Control and Optimization, Vol.9, no. 1*, 112-147.

101 Law, R., (1998). Room Occupancy Rate Forecasting: A Neural Network Approach, *International Journal of Contemporary Hospitality Management*, 10 (6), 234-239.

102 Le Cun, Y., Simard, P.Y., Pearlmutter, B., (1993). Automatic Learning Rate Maximization by On-Line Estimation of The Hessian's Eigenvectors. In S.J. Hanson, J.D. Cowan, and C.L. Giles (Eds.), *Advances in Neural Information Processing Systems, Volume 5*, 156-163. San Mateo, CA: Morgan Kaufmann.

103 LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., Jackel, L.D., (1989). Back Propagation Applied to Handwritten Zip Code Recognition. *Neural Computation, 1, 4*, 541-551.

104 Lee, J., Bien, Z., (1991). Improvement on Function Approximation Capability of Backpropagation Neural Networks. *Proceedings of the Internal Joint Conference on Neural Computation, 1*, 541-551.

105 Lenard, M., Alam, P., and Madey, G., (1995). The Applications of Neural Networks and a Qualitative Response Model to the Auditor's Going Concern Uncertainty Decision. *Decision Science, 26, (2)*, 209-227.

106 Levenberg, K. (1944). A Method for the Solution of Certain Non-Linear Problems in Least Squares. *Quarterly Journal of Applied Mathematics, II, 2*, 164-168.

107 Li, H. X., and Da, X. L., (2000). A Neural Network Representation of Linear Programming. *European Journal of Operations Research, 124*, 224-234.

108 Lisi, F. and Schiavo, R.A. (1999). A Comparison Between Neural Networks and Chaotic Models for Exchange Rate Prediction. *Computational Statistics & Data Analysis, 30*, 87-102.

109 Luenberger, D. G., (1984). *Introduction to Linear and Nonlinear Programming., 2nd ed.* Addision-Wesley, Reading, Mass., USA.

110 Ma, J., Tian, P., and Zhang, D-M., (2000). Global Optimization by Darwin and Boltzmann Mixed Strategy. *Computers and Operations Research, 27*, 143-159.

111 Makram-Ebeid, S., Sirat, J.A., Viala, J.R., (1989). A Rationalized Back Propagation Learning Algorithm. In *Proceedings of the Internal Joint Conference on Neural Networks, 2*, 373-380, New Jersey, USA.

112 Mandenhall, W., and Sincich, T., (1996). *A Second Course in Statistics: Regression Analysis, 5th Edition*, Prentice Hall, NJ.

113 Marquardt, D.W., (1963). An Algorithm for Least-Squares Estimation of Non-Linear Parameters. *Journal of the Society of Industrial and Applied Mathematics, 11, 2*, 431-441.

115 Mathews, J.H., (1992). *Numerical Methods for Mathematics, Science and Engineering., 2nd Edition*, Prentice Hall, Inc., Englewood Cliffs, NJ, USA.

116 Matsuoka, K., Yi, J., (1991). Backpropagation Based on the Logarithmic Error Function and Elimination of Local Minima., *Proceedings of the International Joint Conference on Neural Networks, Singapore, 2*, 1117-1122.

117 McClelland, James L., Rumelhart, David, E., and the PDP Research Group., (1986). *Parallel Distributed Processing. Vol 2:*, Psychological and Biological Models. MIT, Cambridge, MA, USA.

118 McCormic, G., and Pearson, J., (1969). Variable Metric Methods and Unconstrained Minimization in *Optimization*, R. Fletcher (Ed.) Academic Press, London, 307-325.

119 McCulloch, W. S., and Pitts, W., (1943). A Logical Calculus of the Ideas Immanent in Nervous Activity, *Bulletin of Mathematical Biophysics, 5*, 115-133.

120 McHaney, R.W., Douglas, D.E., (1997). Multivariate Regression Metamodel: A DSS Application in Industry, *Decision Support Systems, 19*, 43-52.

121 McKinnon, K.I.M., (1998)., Convergence of the Nelder-Mead Simplex Method to Nonstationary Point, *SIAM J. on Control and Optimization, 9*, 148-158.

122 McMenamin, J.S. and Monforte, F.A. (1998). Short Term Energy Forecasting with Neural

Networks. *The Energy Journal, 19, (4)*, 43-61.

123 Mehra, R., and Stepner, D., (1973). Maximum Likelihood Identification and Optimal Input Design for Identifying Air Craft Input Design for Identifying Aircraft Stability and Control Derivatives. *NASA, CR-2000*.

124 Mohandes, M., Codrington, C. W., and Gelfand, S. B., (1994). Two Adaptive Step Size Rules for Gradient Descent and Their Application to the Training of Feed Forward Artificial Neural Networks, *Proceedings of the IEEE International Conference on Neural Networks, I* Orlando, FL 555-560.

125 Moller, Martin., (1997). Efficent training of feed forward neural networks, in Brown Anthony, editor, *Neural Network: Analysis, Architectures and Applications*,: Institute of Physics Publishing, Bristol and Phiadelphia.

126 Morgan, N., Bourlard, H., (1990). Generalization and Parameter Estimation in Feed Forward Nets: Some Experiments. In Touretzky, D.S., (Ed.), *Advances in Neural Information Processing Systems Vol. 2.*, Morgan Kaufmann, San Mateo, CA, 630-637.

127 Mosteller, F., and Tukey, (1968). Data Analysis Including Statistics. In G. Lindzey and E. Aronson (Eds). *Handbook of Social Psychology, 2*, Addision-Wesley, Reading Mass.

128 Nachtsheim, P. R., (1994). A First Order Adaptive Learning Rate Algorithm for Back Propagation Networks, *Proceedings of the IEEE International Conference on Neural Networks. I* Orlando, FL 257-262.

129 Nelder, J. A., and Mead, R., (1965). A Simplex Method for Function Minimization, *The Computer Journal, 7*, 308-313.

130 Oren, S. and Luenberger, D., (1974). Criteria and Sufficient Conditions for Scaling a Class of Algorithms, Part-I, *Management Science*, 845-861.

131 Oren, S., (1972). *Self-Scaling Variable Metric Algorithms for Unconstrained Minimization.*, PhD Thesis, Standford University.

132 Park, D.C., El-Sharkawi, M.A., Mark II, R.J., Atlas, L.E., and Damborg, M.J., (1991). Electric Load Forecasting using an Artificial Neural Network, *IEEE Transactions on Power Systems, 2*. 442-449.

133 Parker, D.B., (1985). Learning logic: Casting the Cortex of the Human Brain in Silicon. (*Tech. Rep. TR-47*). Center for Computational Research in Economics and Management Science, MIT. USA.

134 Pierre, D.A., (1969). *Optimization Theory With Applications.*, John Wiley and Sons, New York.

135 Pirez, Y. M., and Sarkar, D., (1993). Back-Propagation Algorithm with Controlled Oscillation of Weights. *Proceedings of the IEEE International Conference on Neural Networks, I* San Francisco, CA. 21-26.

136 Plaut, D., Nowlan, S. and Hinton, G.E., (1986). Experiments on Learning by Back Propagation. Technical *Report CMU-CS-86-126.*, Department of computer Science , Carnegie Mllon University, Pittsburgh, PA., USA.

137 Polak, E. and Ribiere, G., (1969). Note Sur la Convergence de Methods de Directions Conjures. *Revue Francaise Information Recherche Operationnelle, 16*, 35-43.

138 Polyak, B. T., (1969). The Method of Conjugate Gradient in Extremum Problems, USSR *Computational Mathematics and Mathematical Physics (English Translation), 9*, 94-112.

139 Powell, M.J.D., (1964). An Efficient Method For Finding the Minimum of a Function of Several Variables Without Calculating Derivatives, *Computer Journal, 17*, 155-162.

140 Powell, M.J.D., (1974). Unconstrained Minimization Algorithms Without Computation of Derivatives. *Bollettino della Unione Matematica Italiana, 9*, 60-69.

141 Qian, N., (1999). On Momentum Term in Gradient Descent Learning Algorithms, *Neural networks, 12*, 145-151.

142 Rosenblatt, F., (1958). The Perceptron: A Probabilistic Model for Information Storage and Organisation in the Brain. *Psychol. Rev. 65*, 386-408, 1958.

143 Rosenblatt, F., (1962). *Principle of Neuro Dynamics: Perceptrons and the Theory of Brain Mechanisms., Spartan*: Washington DC, USA.

144 Rothe, H., (1998). Approaches to Pattern Recognition, RTO Lecture Series 214, Advanced Pattern Recognition Techniques, NATO Research & Technology Organization; Rue Ancelle, France, 1-29.

145 Rumelhart, D.E. and McClelland, J.L., (1986). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol.1.*, MIT Press, Cambridge, MA.

146 Rumelhart, D.E., Hinton, G.E., and Williams, R. J., (1986). Learning Internal Representation by Error Propagation. In Rumelhart, D.E. and McClelland, J.L and PDP research group (Eds.), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1*: Foundations, 318-362, MIT Press, Cambridge, MA., USA.

147 Salomon, R., and Van Hemmen, L., (1996). Accelerating Backpropagation Through Dynamic Self-Adaptation, *Neural Networks, 9, (4)*, 589-601.

148 Samad, T., (1990). Backpropagation Improvements Based Heuristic Arguments. Proceedings of the *International Joint Conference on Neural Networks, Washington, DC, 1*, 565-568.

149 Schwarz, H.R., (1973). *Numerical Analysis of Symmetric Metrics.*, Prentice Hall, Inc, Englewood, Cliffs, NJ.

150 Schwefel, Hans-Paul., (1981). *Numerical Optimization of Computer Models*, John Wiley & Sons, Chichester.

151 Sexton, R.S., Alidaee, B., Dorsey, R.E., and Johnson, J.D. (1998). Global Optimization for Artificial Neural Networks: A Tabu Search Application. *European Journal of Operations research, 106*, 570-584.

152 Shang, Y., Wah, B.W., (1996). Global Optimization for Neural Network Training, *Computer, March*, 45-54.

153 Shanno, D.F., (1970). Conditioning of Quasi Newton Methods for Function Minimizations., *Mathematics of Computation, 24*, 641-656.

154 Shanno, D.F., (1978). Conjugate Gradient Methods with Inexact Line Searches, *Mathematics of Operations Research, 3*, 244-256.

155 Shanno, D.F., (1980). On Variable Metric Methods for Sparse Hessian: *Mathematics of Computation, 34*, 499-514.

156 Shoemaker, PA, Carline, M.A.J., Shimabukuro, R.L., (1991). Backpropagation Learning With Trinary Quantization of Weight Updates, *Neural Networks*, 231-241.

157 Sieger, D. B., and Badiru, A. B., (1993). An Artificial Neural Network Case Study: Prediction Versus Classification in Manufacturing Application, *Proceeding of the 15th Annual Conference on Computers and Industrial Engineering*, Cocoa Beach, Florida.

158 Sietsma, J., Dow, R.J.F., (1991). Creating Artificial Neural Networks That Generalize. *Neural Networks, 2*, 67-79.

159 Silva, F.M., and Almeida, L.B., (1990). Acceleration Techniques For the Back-Propagation Algorithm. *Lecture Notes in Computer Science, 412*, 110-119.

160 Snee, R.D., (1977). Some Aspects of Nonorthogonal Data Analysis, Part I. Developing Prediction Equations, *Journal of Quality Technology, 5*, 67-79.

161 Solla, S.A., Levin, E., Flesher, M., (1988). Accelerated Learning in Layered Neural Networks., *Complex Systems, 2*, 34-39.

162 Sorenson, H.W., (1969). Comparison of Some Conjugate Direction Procedures for Function Minimization, *J. Franklin Institute, 288*, 421-441.

163 Spendley, W, Hext, G.R., and Himsworth, F.R., (1962). Sequential Application of Simplex Design in Optimization and evolutionary Operations, *Technometrics, 4*, 441-461.

164 Stone, M., (1974). Cross-validation Choice and Assessment of Statistical Predictions (with discussions). *Journal of Royal Statistical Society Ser. B, 36*, 111-147.

165 Subramanian, V., Hung, M.S., (1990). A GRG2- Based Systems for Training Neural Networks: Design and Computational Experience, *ORSA Journal on Computing, 5 (4)*, 386-394.

166 Swann, W. H., (1964). Report on the Development of a New Direct Search Method of Optimization. Imperial Chemical Industries Ltd. *Central Instr. Res. Lab. Research Note, 64/3*, London.

167 Tibshirani, R., (1996). A Comparison of Some Error Estimates for Neural Network Models., *Neural Computation, 8*, 152-163.

168 Torczon, V., (1989). *Multi-Directional Search: A Direct Search Algorithm for Parallel Machines*, Ph.D. thesis, Rice University, Houston, TX, USA.

169 Torczon, V., (1997). On the Convergence of Pattern Search Algorithms, *SIAM J. on Control and Optimization, 7, (1), 1-25.*

170 Towsey, M., Alpsan, D. and Sztriha, L, (1995). Training a Neural Network With Conjugate Gradient Methods. *Proceedings of the IEEE International Conference on Neural Networks, I*, Perth, Australia, 373-378.

171 Tseng, P., (1995). *Fortified-Descent Simplicial Search Method: A General Approach*, Technical Report, Department of Mathematics. U of Washington, Seattle, WA, USA.

172 Van Ooyen, A., and Nienhuis, B., (1992). Improving the Convergence of the Back Propagation Algorithm. *Neural Networks, 5*, 465-471.

173 Van Wezel, M.C, and Baets, W. R.J., (1995). Predicting Market Responses with a Neural Network: The Case of Fast Moving Consumer Goods, *Marketing Intelligence and Planning, 13, (7)*, 23-30.

174 Vogl, T. P., Mangis, J.K., Rigler, A.K., Zink, W.T., and Alkon, D.L., (1988). Accelerating the Convergence of the Back-Propagation Method. *Biological Cybernetics, 59*, 257-263.

175 Wang, S., (1995). The Unpredictability of Standard Backpropagation Neural Networks in Classification Applications, *Management Science, 41 (3)*, 555-559.

176 Wang, Y.J. and Lin, C.T., (1998). A Second-Order Learning Algorithm for Multilayer Networks Based on Block Hessian Matrix. *Neural Networks, 11*, 1607-1622.

177 Warner, B., and Misra, M., (1996). Understanding Neural Networks as Statistical Tools, *The American Statistician, 50 (4)*, 284-239.

178 Watrous, R. L., (1987). Learning Algorithms for Connectionist Networks: Applied Gradient Methods of Nonlinear Optimization, *IEEE First International Conference on Neural Networks, 2*, San Diego, CA, 619-627.

179 Webb, A.R., and Lowe, D., (1988). A Hybrid Optimization Strategy for Adaptive Feed-Forward Layered Networks. *RSRE Memorandum 4193*, Royal Signals and Radar Establishment, St Andrews Road, Malvern, UK.

180 Weigend, A., Rumelhart, D. and Huberman, B., (1990). Back-Propagation, Weight Elimination and Time Series Prediction, In *Connectionist Models*, Proc. 1990, Touretzky, Elman, Sejnowski and Hinton Editors, 105-116.

181 Weigend, A.S., Huberman, B.A., and Rumelhart, D.E.,(1990). Predicting the Future: A Connectionist Approach, *International Journal of Neural Systems, 1*, 193-209.

182 Weigend, A.S., Rumelhart, D.E. and Huberman, B.A. (1991). Generalization by weight elimination with application to forecasting, in Lippmann, R.P., Moody, J. and Touretzky, D.S. (Editors), *Advances in Neural Information Processing Systems, Volume 3*, Morgan Kaufmann, San Mateo .

183 Weir, M.K., (1991). A Method for Self-Determination of Adaptive Learning Rates in Back Propagation. *Neural Networks, 4*, 371-379.

184 Werbos, P., (1974). *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. PhD thesis, Harvard University.

185 White, H., (1987). Some Asymptotic Results for Back-Propagation. *Proceedings of the IEEE Conference on Neural Networks, 3*, IEEE, San Diego, CA, 261-266.

186 Widrow, B. and Lehr, M.A., (1990). 30 years of Adaptive Neural Networks: Perceptron, Madaline, and Backpropagation. *Proceedings of IEEE, 79, 9*, 1415-1442.

187 Wu, Chih-Hung, Ventura, J. A., and Browning, S., (1998). Computational Comparisons of Dual Conjugate Gradient Algorithms for Strictly Convex Networks. *Computers Operations Research, 25, (4)*, 333-349.

188 Yu, X., Loh, N. K., and Miller, W. C., (1993). A New Acceleration Technique for the Back Propagation Algorithm, *Proceedings of the IEEE International Conference on Neural Networks, 1* San Francisco, CA, 1157-1161.

189 Zhang, G., Hu, M. Y., Patuwo, B. E., and Indro, D. C., (1999). Artificial Neural Networks in Bankruptcy Prediction: General Framework and Cross Validation Analysis, *European Journal of Operations Research, 116*, 16-32.

190 Zhang, J. H., and Xu, X. H., (1999). An Efficient Evolutionary Programming Algorithm. *Computers and Operations Research, 26*, 645-663.

Appendix

A

Experimental Data Set

A.1 Parity Problem

The experimental data set for the parity problem is listed in Table A1. There are 32 input patterns with the corresponding output value. The ANN training determines whether an algorithm can classify this data set or not. If so to what extent the output matches with the actual data set. The performance of an algorithm in epoch measure, function evaluation and terminal function value is noted.

Input Pattern						Output	Input Pattern						Output
#	1	2	3	4	5		#	1	2	3	4	5	
1	0	0	0	0	0	0	17	1	0	0	0	0	1
2	0	0	0	0	1	1	18	1	0	0	0	1	0
3	0	0	0	1	0	1	19	1	0	0	1	0	0
4	0	0	0	1	1	0	20	1	0	0	1	1	1
5	0	0	1	0	0	1	21	1	0	1	0	0	0
6	0	0	1	0	1	0	22	1	0	1	0	1	1
7	0	0	1	1	0	0	23	1	0	1	1	0	1
8	0	0	1	1	1	1	24	1	0	1	1	1	0
9	0	1	0	0	0	1	25	1	1	0	0	0	0
10	0	1	0	0	1	0	26	1	1	0	0	1	1
11	0	1	0	1	0	0	27	1	1	0	1	0	1
12	0	1	0	1	1	1	28	1	1	0	1	1	0
13	0	1	1	0	0	0	29	1	1	1	0	0	1
14	0	1	1	0	1	1	30	1	1	1	0	1	0
15	0	1	1	1	0	1	31	1	1	1	1	0	0
16	0	1	1	1	1	0	32	1	1	1	1	1	1

Table A.1 Input pattern for 5-5-1 parity problem with one output

A 2 Random starting points

The random starting points for the training problems are listed in Table A.2. To create different starting points the numbers in Table A.2 can be factored according to the requirements.

$w_i \downarrow$	Experiment #									
	1	2	3	4	5	6	7	8	9	10
1($\equiv w_{i=1}$)	.02	-.01	-.1	.1	-1	1	20	.22	100	323
2	.04	.04	.2	.5	3	5	34	.34	545	245
3	.02	.02	.2	.2	6	3	22	.97	122	322
4	.03	.03	-.3	.1	4	1	13	.73	431	231
5	.02	.02	.1	.3	2	1	42	.10	326	126
6	.05	.08	.2	.3	2	2	57	.20	200	125
7	.09	.04	-.3	.1	-1	6	94	.55	541	141
8	.05	.05	.7	.3	7	3	55	.35	553	253
9	.07	.05	.1	.9	9	3	75	.25	559	159
10	.02	-.02	.8	.9	8	9	22	.82	229	229
11	.03	.03	.1	.3	5	3	30	.93	300	433
12	.03	.02	.8	.4	2	1	32	.83	321	311
13	.03	.03	-.4	.5	-5	4	33	.73	335	421
14	.03	.03	.1	.9	9	3	33	.63	339	789
15	.01	.01	.9	.2	7	2	11	.51	117	643
16	.04	.04	.1	.4	4	7	40	-.22	400	973
17	.04	.03	.7	.2	2	0	43	-.33	432	232
18	.03	.04	.9	.5	5	1	34	-.44	345	345
19	.03	.04	-.4	.9	-9	2	37	-.67	449	249
20	.04	-.07	.9	.6	6	3	44	-.54	446	146
21	.09	-.02	.2	.3	5	6	92	.41	120	431
22	.03	.07	.2	.2	5	3	30	.35	300	727
23	.04	.04	.7	.1	5	4	44	.34	445	145
24	.05	.03	.4	.9	9	5	53	.23	439	239
25	.02	.07	-.1	.8	-2	2	23	.13	432	132
26	.03	.02	.1	.1	1	3	33	.21	121	133
27	.05	.01	.2	.2	2	2	25	.53	221	254
28	.03	.04	.2	.1	1	9	43	.24	122	243
29	.07	.07	.5	.3	3	3	37	.25	322	355
30($\equiv w_{i=30}$)	.08	.02	.4	.1	5	7	88	.61	511	634

Table A.2 Random starting points with different magnitudes

A.3 The L-T Letter Recognition Problem

The orientations of the L-T letters in four different orientations according to the figure 3.3 and 3.4 (Chapter 3) in are listed in Table A.3 and A.4. These data constitute the training set for the letter recognition problem.

		1	2	3	4	5	6	7	8	9	output
Training Set	a	1	0	0	1	0	0	1	1	1	0
	b	0	0	1	0	0	1	1	1	1	0
	c	1	1	1	0	0	1	0	0	1	0
	d	1	1	1	1	0	0	1	0	0	0

Table A.3 Input Training Pattern (L)

		1	2	3	4	5	6	7	8	9	output
Training Set	a	1	1	1	0	1	0	0	1	0	1
	b	1	0	0	1	1	1	1	0	0	1
	c	0	1	0	0	1	0	1	1	1	1
	d	0	0	1	1	1	1	0	0	1	1

Table A.4 Input Training Pattern (T)

A.3 Quarterly Seasonal Time Series Data

The seasonal time series data shown in Table A.5 for the Australian peak electric load is collected from the reports available with the Australian Bureau of Statistics (ABS, #8301.0, Jan, 1998). The data covers the period from September quarter 1976 to June quarter 1997. These data are used to train an ANN time series model.

	1976	*77*	*78*	*79*	*80*	*81*	*82*	*83*	*84*	*85*	*86*
Sep.Q1	22272	23270	24494	25994	26690	28345	28590	29290	31856	33165	34375
Dec.Q2	19684	20107	21396	22802	24132	24890	25371	26684	28343	29648	31046
Mar.Q3	19031	20084	21323	22604	23639	24675	24687	26641	28982	29506	31015
Jun.Q4	21545	22634	23644	24510	26320	27122	27285	29042	30838	32002	33685

	87	*88*	*89*	*90*	*91*	*92*	*93*	*94*	*95*	*96*	*97*
Sep.Q1	35850	37154	40113	40766	41124	41936	42198	42891	43894	43742	45521
Dec.Q2	33211	35303	36278	37353	37800	38187	40222	40326	40452	40761	43169
Mar.Q3	33142	36905	36715	37254	38024	38927	39177	40075	41076	41011	45920
Jun.Q4	34664	37333	38801	38450	39466	40822	40215	41773	42122	42901	44629

Table A.5: Peak Electric Load Quarterly Data

A.4 Hotel Occupancy Rate in Australian Hotel Industry

The data listed in Table A.6 show the hotel occupancy rate in terms of *room nights spent.* These data are collected form the Australian Bureau of Statistics report (ABS, #6401.1, Sept, 1999; ABS, # 1350, Aug, 1998; # 1350, Sept, 1999; ABS #86350.0, Sept, 1999). There are only limited data available. The training experiment involves in finding a trained ANN that can be used as a calibration model similar to multi-variate statistical analysis approach.

Quarter	CPI	GDP	Room Nights
	All groups	$Million	x1000
March 97	120.5	133002	3797.5
Jun	120.2	135537	3575.8
Sep	119.7	136710	3869.8
Dec 97	120.0	138489	4004.8
Mar 98	120.3	140081	3816.8
Jun	121.0	141637	3645.1
Sep	121.3	143024	4036.0
Dec 98	121.1	145240	4194.4

Table A.6 Hotel occupancy rate as room nights

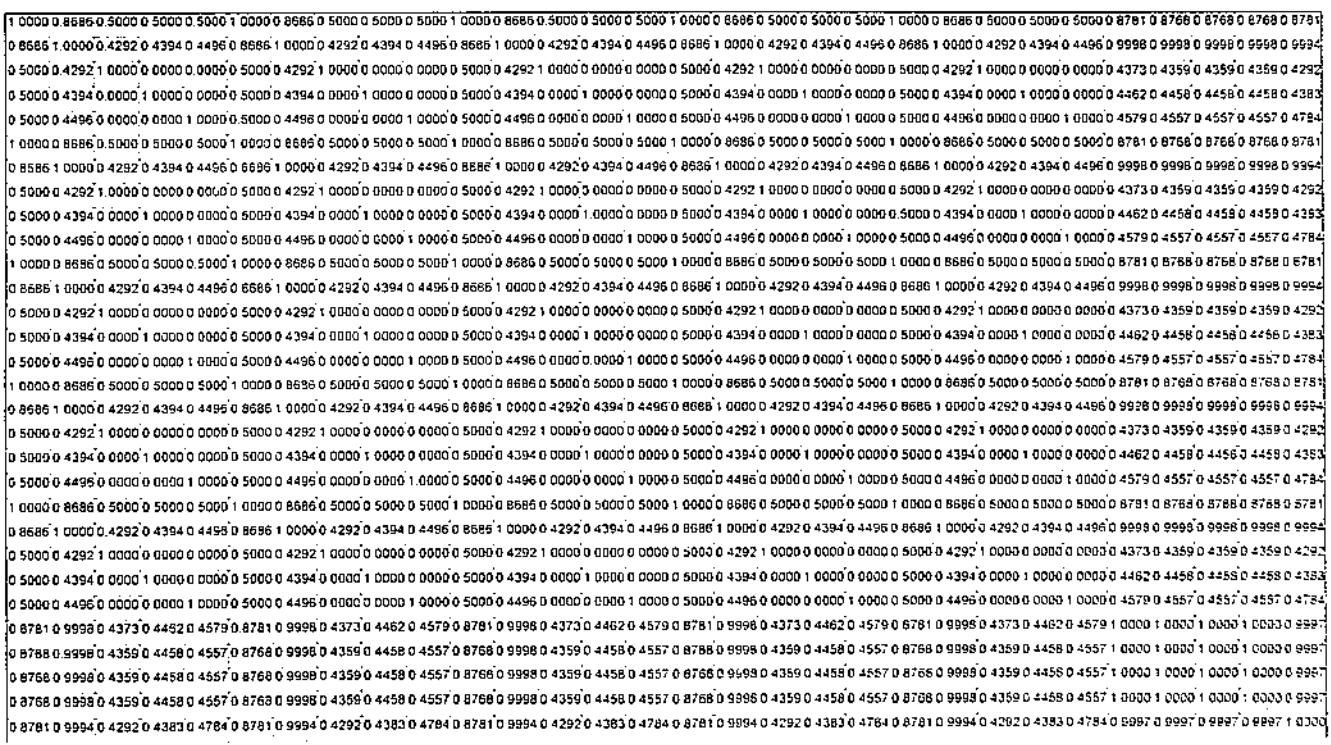


Table A.7 Hessian matrix of the error function with quarterly seasonal time series (Peak Electric Load Forecast: 5-5-1 ANN configuration)