Application of Multilayer Feedforward Neural Networks to Precipitation Cell-Top Altitude Estimation

by

Michelle S. Spina

B.S.E.E., Rochester Institute of Technology (1991)

Submitted to the Department of Electrical Engineering and Computer Science

in partial fulfillment of the requirements for the degree of

Master of Science in Electrical Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 1, 1994

Author........[illegible]........

Department of Electrical Engineering and Computer Science

September 1, 1994

Certified by........[illegible]........

David H. Staelin

Professor of Electrical Engineering

Thesis Supervisor

Accepted by........[illegible]........

Frederic R. Morgenthaler

Chair, Department Committee on Graduate Students

Application of Multilayer Feedforward Neural Networks to Precipitation Cell-Top Altitude Estimation

by

Michelle S. Spina

Submitted to the Department of Electrical Engineering and Computer Science on September 1, 1994, in partial fulfillment of the requirements for the degree of Master of Science in Electrical Engineering

Abstract

The use of passive 118-GHz O_2 spectral observations of rain cells for precipitation cell-top altitude estimation is demonstrated using a multilayer feedforward neural network retrieval system. Data was derived from a collection of 118-GHz rain cell observations along with estimates of the cell-top altitude obtained by optical stereoscopy. The observations were made using the millimeter-wave temperature sounder (MTS) scanning spectrometer aboard the NASA ER-2 research aircraft, flying near 20 km altitude, during the Genesis-of-Atlantic-Lows-Experiment (GALE) and the Cooperative-Huntsville-Meteorological-Experiment (COHMEX), 1986. The neural network estimator applied to MTS spectral differences between clouds and nearby clear air yielded an RMS discrepancy of 1.76 km for a combined cumulus, mature and dissipating cell set and 1.44 km for the cumulus-only set. An additional improvement in RMS discrepancy to 1.36 km was achieved by including additional MTS information on the absolute atmospheric temperature profile. An incremental method for training neural networks was developed which yielded robust results despite the use of as few as 56 training spectra. Comparison of these results with a nonlinear statistical estimator shows that superior results can be obtained with a neural network retrieval system. The neural network estimator was then used to create imagery of cell-top altitudes estimated from 118-GHz CAMEX spectral imagery gathered from September through October, 1993, and from spectral imagery gathered from cyclone Oliver on February 7, 1993.

Thesis Supervisor: David H. Staelin
Title: Professor of Electrical Engineering

Acknowledgments

I would like to thank a number of people who have helped to make this thesis possible. First, I would like to thank my research advisor, Professor Staelin, for his support and guidance. Also, thanks to Michael Schwartz and Carlos Cabrera for all of their help and for making the lab a fun place to work. I would also like to acknowledge a number of faculty at RIT for their encouragement for me to continue on to graduate school; Professors Unnikrishnan, Salem, Walker, and Madhu - thank you very much.

I can not thank my family enough for the ongoing support they have given me. Their emotional and financial help has been invaluable throughout my college career. Thanks for stressing the value of an education, and for giving me the support and genetic material to succeed. I would also like to thank all of my friends from home and college, Katy, Lisa, Darlene, Kathy, Sheri and Pam for their calls and letters, which always cheer me up. Also, thanks to Matt, Joel, and Chave for making graduate student life at MIT a lot of fun.

Finally, although I could never possibly put into words my love and gratitude, I would like to thank my husband Robert for everything. I would not have been able to get through this without his guidance, support and love. Thank you for believing that I can accomplish anything that I dream.

This research has been supported by the Research Laboratory of Electronics at MIT under NASA contract NAS5 31376.

Contents

page

1. Introduction **13**

2. Passive Microwave Remote Sensing of Cell-Top Altitude **15**

2.1 Physics of Downlooking Sensors 15

2.2 Retrieval of Cell-Top Altitude from 118.75-GHz Spectra 19

2.3 Discussion of Previous Methods 23

2.3.1 Linear Regression 24

2.3.2 Non-Linear Statistical Estimator 25

2.4 Why Neural Networks 28

3. Introduction to Multilayer Feedforward Neural Networks **31**

3.1 Artificial Neurons 31

3.2 Multilayer Feedforward Neural Networks 34

3.3 Training of Neural Networks by Back-Propagation 35

page

4. Development of Altitude Retrieval System **41**

4.1 Problem Description 41

4.2 System Design 41

4.2.1 Description of Neural Network Model 42

4.2.2 Description of Data Sets 43

4.2.3 Topology Comparison 45

4.2.4 Development of Incremental Neural Network 47

4.3 Summary of Results 50

4.3.1 Images of Retrieved Cell-Top Altitude 51

5. Conclusions / Suggestions for Further Research **55**

Appendix A MATLAB Code **57**

References **73**

List of Figures

page

2-1 Illustration of downlooking sensor measurements 17

2-2 Temperature weighting functions 18

2-3 Collection of 118.75-GHz delta brightness temperature spectra and clear-air reference spectra observed by the MTS over rain cells during GALE and COHMEX 21

2-4 Scatter plot of the optical cell-top altitude versus cell size for the 118-GHz rain cell observations 23

2-5 Illustration of cell-top altitude estimation problem 24

2-6 Eigenvalues of covariance matrix of the perturbation spectra 27

2-7 Scatter plot of KLT mode 1 amplitude versus KLT mode 2 amplitude 28

3-1 Representation of the jth node in a neural network 32

3-2 Three common activation function nonlinearities 33

3-3 A schematic of a multilayer feedforward neural network 35

4-1 Performance of neural network retrieval system with one hidden layer with five nodes on a training and test set 47

4-2 Performance of neural network retrieval system that is overfit to the training data 48

4-3 Illustration of incremental neural network system 50

4-4 Images of retrieved precipitating cell-top altitudes 52

4-5 Cell-top altitude retrieval error versus scan angle 53

4-6 Retrieved cell-top altitudes from Cyclone Oliver 54

List of Tables

page

4.1 Summary of data sets used in development of cell-top altitude retrieval system 44

4.2 Comparison of RMS errors (km) from different neural network topologies 46

4.3 Comparison of rms errors (km) for cell-top altitude estimators 51

Chapter 1

Introduction

Cloud and precipitation properties can be remotely sensed from brightness temperature measurements made at millimeter wavelengths. In particular, the estimation of rainfall rate would provide a means for monitoring the hydrologic cycle over inaccessible regions as well as providing advanced warning of foul weather to highly populated regions. The potential of passive remote sensing of rainfall rate has been discussed [1] and retrievals of rainfall rate have been demonstrated from 19.35-GHz passive data [2] and from 18- and 37-GHz passive data [3]. Frequencies within 2.5-GHz of the 118.750-GHz O_2 line have been shown to support retrievals of precipitation parameters, in particular, cell-top altitude. It has been shown that thunderstorm cloud height is statistically related to rainfall rate, although the relationship is strongly influenced by climatological region [4]. In addition, correlations between the maximum cell top altitude and both the total rainfall volume and the maximum rainfall volume rate have been revealed [5]. Information provided by independent cell-top altitude estimates would be beneficial over highly glaciated cells, where rainfall-rate retrievals using only 19- and 37-GHz frequencies is compromised by a layer of strongly scattering ice.

Cell-top altitude retrievals have been demonstrated using passive measurements of the infrared radiance emitted at the cloud top [6]. However, because of scattering and absorption of infrared radiation, passive infrared observations cannot probe beneath non-precipitating cloud canopies. At microwave frequencies there is significantly less extinction of the radiation, allowing for the direct probing of precipitation particles located directly beneath non-precipitating cloud canopies.

This thesis evaluates retrievals of cell-top altitude using multilayer, feedforward neural networks and high-resolution, passive 118-GHz multichannel precipitation cell imagery. The high performance of these networks as function approximators has been shown in the literature [7],[8]. The use of neural networks for precipitating cell-top altitude estimation from 118-GHz spectral data addresses both the complex statistical nature of the spectral data and the unknown, non-linear relationship that exists between the 118-GHz data and cell-top altitude. An optimal mapping of 118-GHz spectral data to cell-top altitude is accomplished by training a multilayer, feedforward neural network using a set of training exemplars which characterize the statistical complexity of the estimation problem. The cell-top altitude estimate produced by the neural network is optimized with respect to mean-squared-error criterion. This is done using the backpropagation algorithm to adjust the parameters of the neural network.

This thesis is organized as follows. Chapter 2 gives a basic introduction to the physics of passive remote sensing of cell-top altitude, a description of the data used to develop a retrieval system, and a discussion of previous retrieval systems developed for the estimation of cell-top altitude from 118-GHz imagery. Chapter 3 introduces artificial neural network topologies and training strategies. Chapter 4 describes the development of the neural network based retrieval system and presents a comparison of results with previously used systems. Chapter 5 presents conclusions and suggestions for further work.

Chapter 2

Passive Microwave Remote Sensing of Cell-Top Altitude

2.1 Physics of Downlooking Sensors

The field of remote sensing deals with the acquisition of information about an object without being in direct contact with it. Information is generally acquired by measuring perturbations that an object makes on a surrounding field. Remote sensing techniques have been instrumental in providing us with information about the Earth's atmosphere. In particular, passive microwave remote sensing allows us to determine a number of atmospheric parameters from measurable atmospheric thermal emission data.

Microwave sensors are used to measure the radiation emitted from molecules present in the atmosphere. The radiation emitted by an object is directly related to its physical temperature. An ideal body whose emitted radiation is dependent only on its physical temperature is referred to as a black body. The radiation emitted by a black body at temperature T (degrees Kelvin) and frequency f (Hz) is described by Plank's Law:

$$I = \frac{2hf^3}{c^2(e^{hf/kT} - 1)} W \cdot m^{-2} \cdot ster^{-1} \cdot Hz^{-1} \qquad (2.1)$$

where $h\ (J \cdot s)$ is Plank's constant, $k\ (J / K)$ is Boltzmann's constant, and $c\ (m / s)$ is the speed of light. At microwave frequencies ($hf << kT$), Plank's Law can be simplified by the Raleigh-Jeans approximation:

$$I \approx \frac{2kf^2T}{c^2} W \cdot m^{-2} \cdot ster^{-1} \cdot Hz^{-1} \tag{2.2}$$

The radiation emitted by a real body is typically described by the temperature of a black body emitting the equivalent amount of radiation. This is referred to as the brightness temperature (T_B) of the real body. The equation of radiative transfer relates the brightness temperature (T_B) of a slab of the atmosphere to the atmospheric composition and temperature $T(z)$ as follows [9]:

$$T_B(f) \approx \int_0^{z_{max}} T(z)\alpha(f,z)e^{-\tau(z,z_{max})}[1 + re^{-2\tau(0,z)}]dz + (1-r)e^{-\tau(0,z_{max})}T_s + re^{-2\tau(0,z_{max})}T_{cb} \tag{2.3}$$

where $T(z)$ is the temperature (K) at height z, $\alpha(f,z)$ describes the absorptivity of the atmosphere (m^{-1}) at frequency f and height z, r and T_s are the reflectivity and temperature of the surface, τ(x,y) is the total opacity of the atmosphere between levels x,y and is given by $\tau(x,y) = \int_x^y \alpha(f,z)dz$, and T_{cb} is the cosmic background temperature. Equation (2.3) illustrates that the measured brightness temperature is the sum of the background radiation and the radiation emitted at each point along the trajectory of the sensor, with each component attenuated by the atmosphere above it. Figure 2-1 illustrates the various measured components described by Equation (2.3).

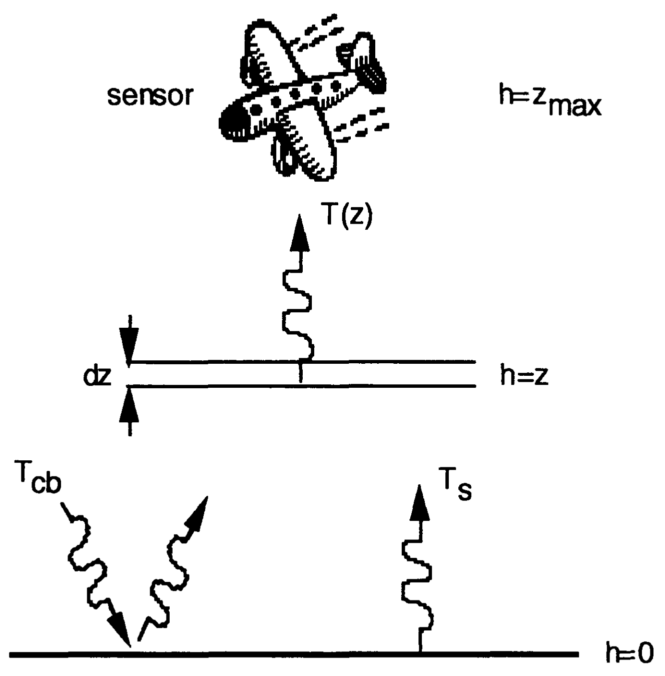

Figure 2-1. *Illustration of downlooking sensor measurements.*

By measuring the emitted radiation at a number of frequencies, it is possible to determine a temperature profile of the atmosphere. This can be explained as follows. The density of the Earth's atmosphere is very low at high altitudes. Therefore, the contribution of the measured radiation from these altitudes is low. Lower in the atmosphere the density increases. This causes an increase in the contribution to the measured radiance from this layer of the atmosphere. Atmospheric layers very near the surface of the earth have the highest density. However, as the radiation from these layers propagates up through the atmosphere toward the sensor, it is absorbed. Therefore, the contribution to the measured radiance from these layers is low. There exists a layer of the atmosphere for which the combination of atmospheric density and absorption above it are such that it contributes the most to the measured radiance. The altitude of this optimum atmospheric layer is a function of observation frequency. As we make measurements at a number of neighboring frequencies, we can determine temperature profiles of the atmosphere. This can also be seen by simplifying Equation (2.3) as follows:

$$T_B(f) \approx \int_0^{z_{max}} T(z)W(f,z)dz + (1-r)e^{-\tau(0,z_{max})}T_s + re^{-2\tau(0,z_{max})}T_{cb} \tag{2.4}$$

where $W(f,z) = \alpha(f,z)[1 + re^{-2\tau(0,z)}]e^{-\tau(z,z_{max})}$, known as the temperature weighting function, shows the contribution to the measured brightness temperature from each layer in the atmosphere at frequency f. Figure 2-2 illustrates the set of weighting functions for the frequencies about 118-GHz. As illustrated by Figure 2-2, as the offset frequency increases, the atmosphere appears more transparent and the peak of the weighting function occurs lower in the atmosphere.

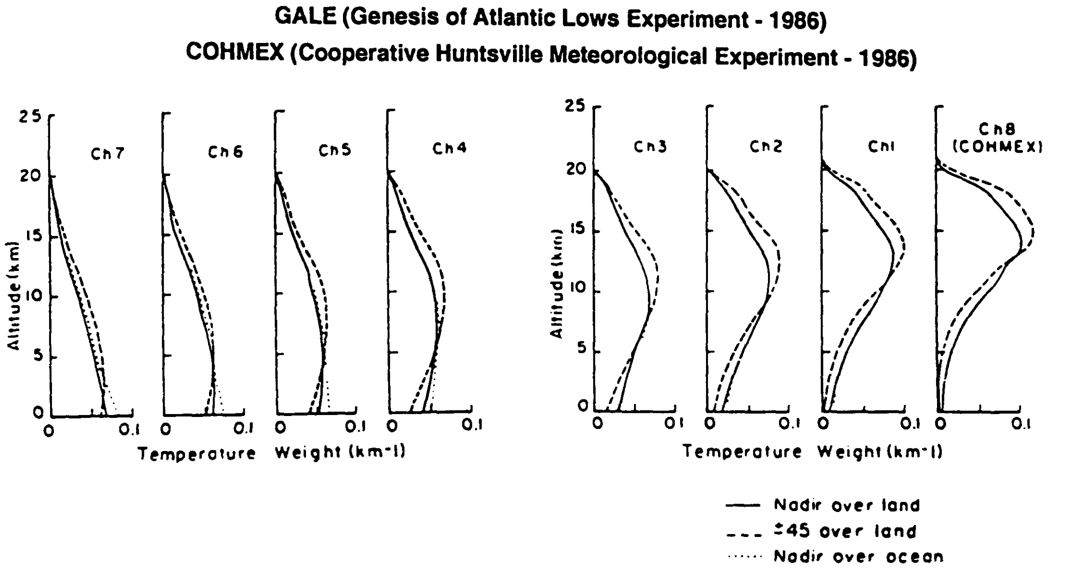

Figure 2-1: *Temperature weighting functions.*

2.2 Retrieval of Cell-Top Altitude from 118.75-GHz Spectra

The presence of precipitating clouds in the atmosphere will perturb the microwave spectrum and therefore the retrieved atmospheric temperature profiles. This occurs for two reasons. If the clouds are below the freezing level, their presence causes increased absorption of the propagating radiation, and therefore a decrease in brightness temperature. Conversely, if the clouds are above the freezing level, the layer of ice that is formed near the cloud top causes strong reflection of the cold cosmic background radiation. This also causes large negative perturbations in the brightness temperature. Referring to the temperature weighting functions shown in Figure 2-2, the clouds at height z will absorb radiation from layers of the atmosphere at heights below height z. In effect, the weighting functions have been chopped off at the height of the cloud. It has been successfully demonstrated by Gasiewski et al. [10] that cloud and precipitation properties can be remotely sensed from brightness temperature measurements made at frequencies within 2.5-GHz of the 118.75-GHz O_2 resonance.

To develop a cell-top altitude retrieval system a collection of 279 independent near-nadir brightness temperature spectra of precipitation cell cores was compiled. The collection consists of spectra produced by the MIT millimeter wave temperature sounder (MTS) scanning spectrometer aboard the NASA ER-2 aircraft during the Genesis of Atlantic Lows Experiment (GALE), in which nineteen spectra were collected, and the Cooperative Huntsville Meteorological Experiment (COHMEX), in which 260 spectra were collected. These observations represent precipitation observed during seven winter and fourteen summer aircraft flights. The observed precipitation cells were located in the southeastern United States, with most of the cells located over the Huntsville, Alabama area. The instrument consists of a scanhead housing two radiometers and a video camera. The 118-GHz scanning spectrometer has a 7.5° spot bandwidth and is band-pass filtered to yield eight non-overlapping probing channels. A rotating mirror allows for a planar scan of the antenna beam over a ±47° field of view, and 14 spots are used to sample the 94° scan sector. The plane of scan is transverse to the flight path. A CCD television camera is mounted in the scanhead and has a 99° transverse field of view. In addition, the instrument houses a time/data video overlay generator and a VHS videocassette recorder. A thorough description of the instrument, aircraft flights, and data calibration is given in Gasiewski et al. [11].

The measured spectrum for a precipitating cell is denoted by an eight-dimensional vector of brightness temperature observations;

$$\bar{T}_{Bi} = \begin{bmatrix} T_{B(118.75\pm0.50)i} \\ T_{B(118.75\pm0.66)i} \\ T_{B(118.75\pm0.84)i} \\ T_{B(118.75\pm1.04)i} \\ T_{B(118.75\pm1.26)i} \\ T_{B(118.75\pm1.47)i} \\ T_{B(118.75\pm1.67)i} \\ T_{B(118.75\pm1.90)i} \end{bmatrix}$$

where the subscripts indicate the channel center and sideband frequencies (GHz) for the MTS, arranged in order of decreasing opacity. For each rain cell spectrum ($\bar{T}_{Bi}$) a corresponding clear-air reference spectrum ($\bar{T}_{Bri}$) was estimated from MTS observations in the vicinity of the cell. A delta brightness spectrum ($\Delta\bar{T}_{Bi}$) was determined as the difference between the cell spectrum and clear-air reference spectrum;

$$\Delta\bar{T}_{Bi} = \bar{T}_{Bi} - \bar{T}_{Bri} \tag{2.5}$$

Delta brightness spectra were used in this retrieval system rather than the absolute brightness spectra to ensure that any fluctuations in the baseline brightness spectra among cell observations due to fluctuations in the ambient atmospheric temperature profile were removed. Typical delta brightness spectra and clear-air reference spectra are plotted in Figure 2-3.

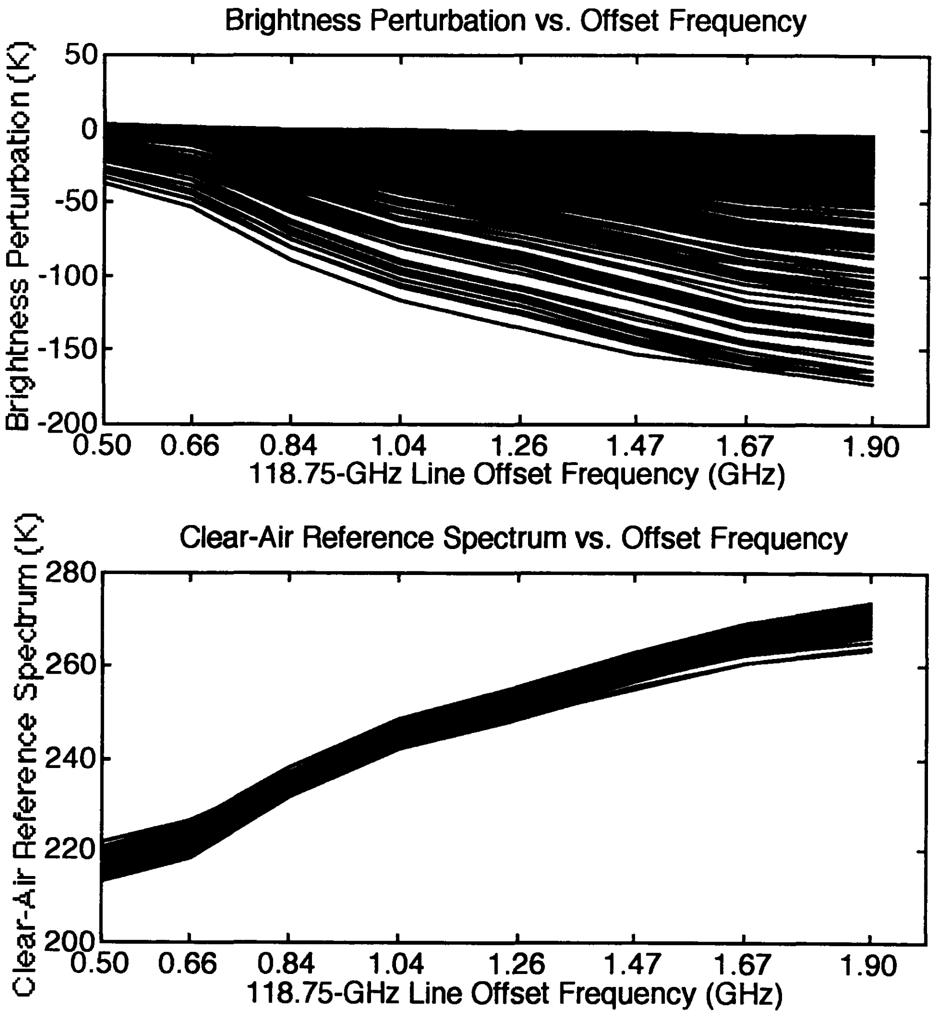

Figure 2-3: *Collection of 118.75-GHz delta brightness temperature spectra and clear-air reference spectra observed by the MTS over rain cells during GALE and COHMEX.*

The altitude of each cell-top was estimated by stereoscopy, using MTS video images (VHS color images through a 99-degree wide-angle lens) and the known altitude and speed of the aircraft. The visually estimated cell-top altitude has an associated rms error of ~1 km due to uncertainty in aircraft speed (±10%), aircraft altitude (±500m), and time of passage of a particular feature of a cell-top through the video field of view (± 2sec).

The size of each cell was estimated using the MTS spectral imagery. The size of each cell was taken to be the distance along the aircraft flight track over which the MTS transparent channel brightness perturbation decreased to half its maximum value. The geometric mean of the major and minor horizontal dimensions was used for elongated rain cells.

In addition to the collection of brightness temperature spectra and cell-top altitudes, observed cells used in this retrieval were classified as one of two types. Cells that appeared to be in their early stages of convection were designated "cumulus-type", while those exhibiting anvils were designated "mature-type". The cells were visually classified using the MTS video imagery. A scatter plot of optically estimated cell-top altitudes versus cell-size is shown in Figure 2-4. The collection of cells range in cell-top altitude and cell-size from ~2km-17km and ~5km-200km, respectively. As shown by Figure 2-4, the logarithm of cell size is linearly related to the cell-top altitude. This apparent relationship will be exploited in the development of the neural network retrieval system.

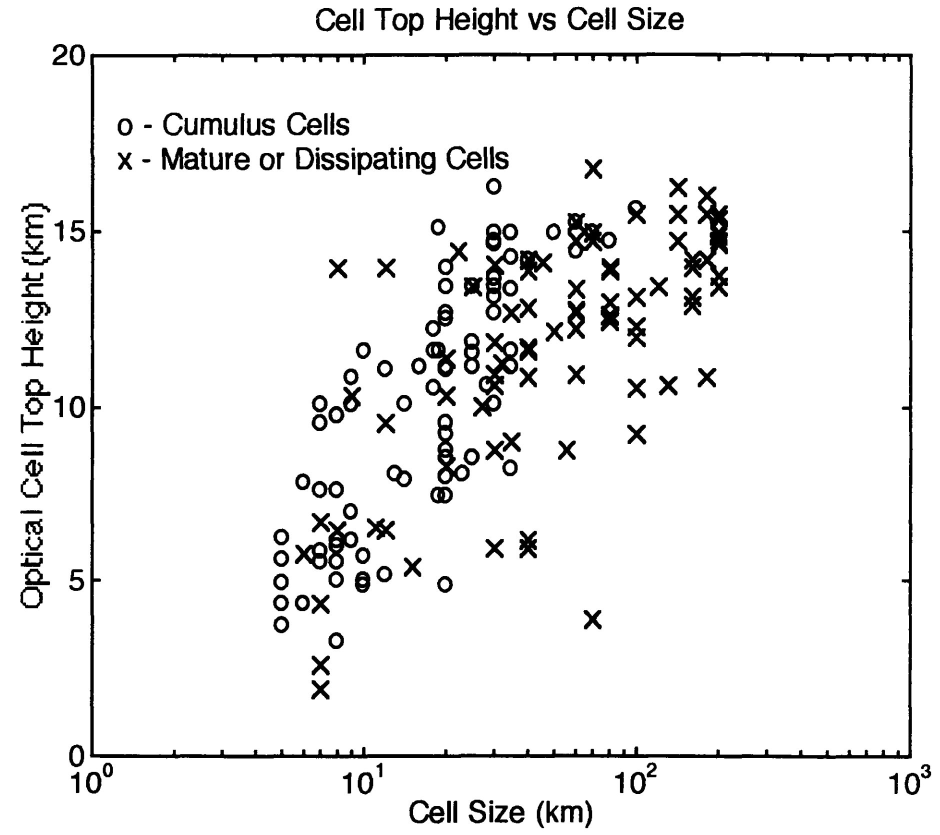

Figure 2-4: *Scatter plot of the optical cell-top altitude versus cell size for the 118-GHz rain cell observations.*

2.3 Discussion of Previous Methods

To determine the relationship between 118-GHz spectral data and cell-top altitude, a function approximation system must be developed. In addition to delta brightness temperature, clear-air reference temperature and cell size may contain information about the cell-top altitude. Figure 2-5 illustrates the three possible estimation approaches to the cell-top altitude retrieval problem.

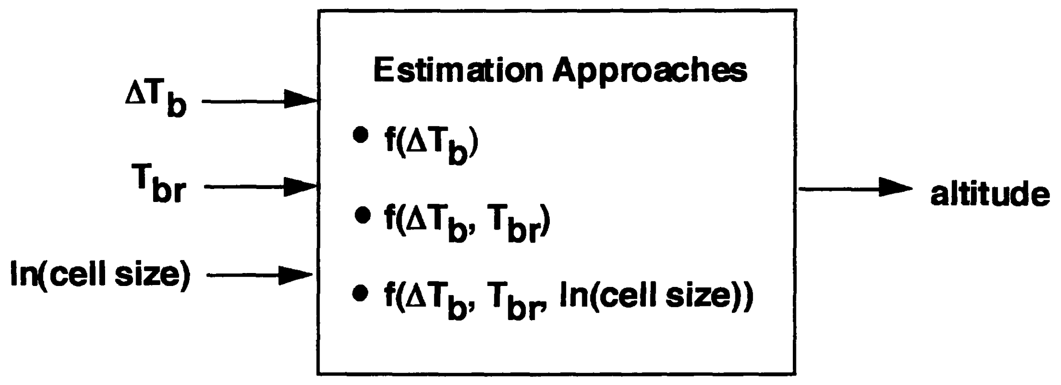

Figure 2-5: *Illustration of cell-top altitude estimation problem.*

To estimate cell-top altitude from 118-GHz spectral data, the unknown function f(*) must be approximated. Approximation theory deals with approximating or interpolating a continuous, multivariate function (f(*)) with an approximation function, $\hat{f}(\overline{W},*)$, that has a fixed number of parameters $\overline{W}$ [12]. To solve this approximation problem the approximation system to use, as well as the algorithm to use to solve for the unknown parameters ($\overline{W}$), must be determined. The determination of the approximation system to use will determine the form of the approximation function ($\hat{f}(\overline{W},*)$). To rate the performance of different approximation functions ($\hat{f}(\overline{W},*)$) and parameters ($\overline{W}$), a measure of the uncertainty of the estimate must be established. Generally this is quantified as the mean-squared error between the actual and estimated output value.

2.3.1 Linear Regression

A simple approach to this problem is to perform linear regression to solve for the unknown parameters $\overline{W}$. In this case, the general form of the approximate function is a linear combination of the 118-GHz spectra;

$$\begin{aligned}\hat{a}_1 &= C + \Delta\overline{T}_{B1}\cdot\overline{D}\\ \hat{a}_2 &= C + \Delta\overline{T}_{B2}\cdot\overline{D}\\ &\vdots\\ \hat{a}_n &= C + \Delta\overline{T}_{Bn}\cdot\overline{D}\end{aligned} \tag{2.6}$$

where $\hat{a}_i$ is the cell-top altitude estimate, $\Delta\overline{T}_{Bi}$ is the ith brightness temperature perturbation spectra, and C and $\overline{D}$ are the unknown parameters to be determined. Equations (2.6) can be rewritten in matrix form as $\overline{a} = T\cdot\overline{W}$, where $\overline{a}$ is the vector of actual cell-top altitudes, $T = \begin{bmatrix}1 & \Delta\overline{T}_{B1}\\ \vdots & \vdots\\ 1 & \Delta\overline{T}_n\end{bmatrix}$, and $\overline{W} = \begin{bmatrix}C\\ \overline{D}\end{bmatrix}$. The vector $\overline{W}$ which will minimize the mean-square error is calculated by;

$$\overline{W} = (T'T)^{-1}T'\overline{a} \tag{2.7}$$

where (') is the transpose operator, and ($^{-1}$) is the inverse operator [13]. This approach assumes a linear relationship between the 118-GHz spectra and cell-top altitude, which may not be a valid assumption. This approach can be made more complicated by adding known or suspected non-linear terms into the regression analysis.

2.3.2 Non-Linear Statistical Estimator

A non-linear statistical estimator operating on the perturbation spectra ($\Delta\overline{T}_B$) was developed by Gasiewski and Staelin [14] to estimate cell-top altitude. The estimator consisted of an orthogonal Karhunen-Loève Transformation (KLT) [15] followed by a rank reduction operation, a non-linear operator and a linear-statistical estimator. The KLT and rank reduction operations were used to reduce the complexity of the perturbation spectra by removing any redundancy that may exist between channels. The non-linear operator linearized the reduced spectra with respect to altitude, after which a linear estimator was applied.

The complexity reduction operation consists of a KLT which rotates the eight-dimensional perturbation spectra space into an orthogonal, eight-dimensional KL space. This is followed by a rank reduction operator which retains the statistically significant KLT modes. The details of this operation are as follows. The KLT is performed by diagonalizing the covariance matrix of the perturbation spectra;

$$\mathfrak{R}_{\Delta\bar{T}_B\Delta\bar{T}_B} = \mathcal{E}'\begin{bmatrix} \lambda_1 & & & 0 \\ & \lambda_2 & & \\ & & \ddots & \\ 0 & & & \lambda_8 \end{bmatrix}\mathcal{E} \tag{2.8}$$

where $\mathfrak{R}_{\Delta\bar{T}_B\Delta\bar{T}_B}$ is the covariance matrix, $\mathcal{E}$ is the row-matrix consisting of the eigenvectors of $\mathfrak{R}_{\Delta\bar{T}_B\Delta\bar{T}_B}$, λ_i are the associated eigenvalues and also a measure of the variance of the i^{th} component of the decomposed spectra, and (') is the transpose operator. The KL transformation is performed by; $k_i = \mathcal{E}\Delta T_{Bi}$.

Figure 2-6 is a plot of the eigenvalues λ_i, and illustrates that only the first and second KLT modes contain any statistically relevant information.

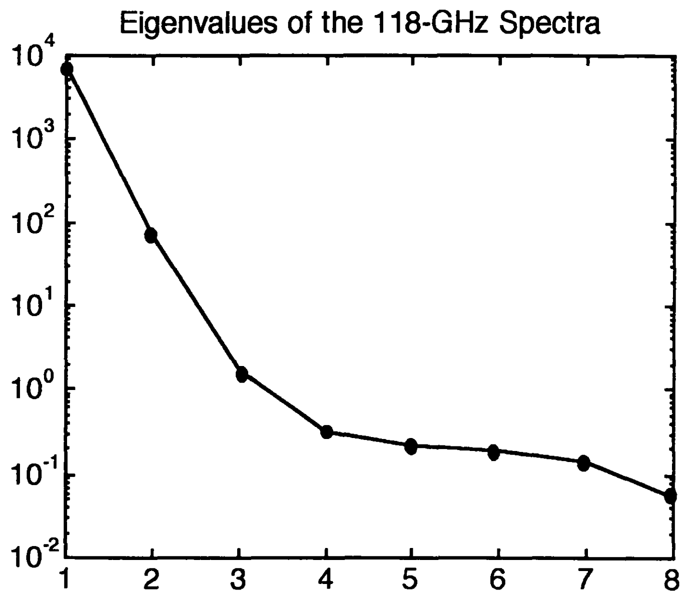

Figure 2-6: *Eigenvalues of covariance matrix of the perturbation spectra.*

The two most dominant KLT coefficients were then linearized with respect to the cell-top altitude. A linear statistical estimator was then used to minimize the mean-squared error between the estimated and actual cell-top altitude.

This estimator produces better results with respect to the mean-squared-error criterion than the linear estimator for the cell-top altitude retrieval problem. However, the use of the KLT to reduce the complexity of the spectra assumes that the information in the spectra can be fully described by second-order statistics. This would completely capture the statistical behavior of the spectra if the spectra also had a jointly Gaussian probability distribution. If this were the case, an elliptical relationship would exist between the KLT mode 1 and KLT mode 2 variables. As shown by Figure 2-7, the relationship between KLT mode 1 and KLT mode 2 is not elliptical. This indicates that information may exist in higher-order statistics, and improvement in cell-top altitude retrievals may be achieved with other methods that can capture this information.

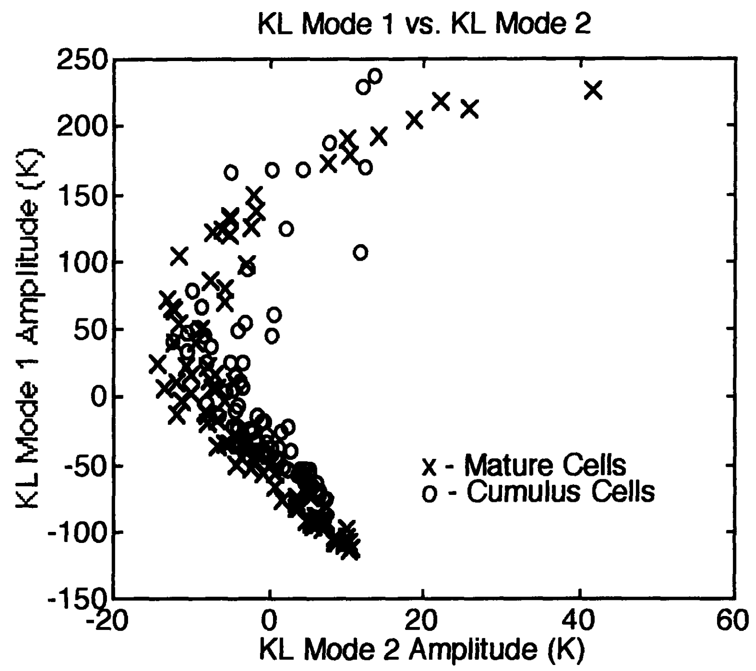

Figure 2-7: *Scatter plot of KLT mode 1 amplitude versus KLT mode 2 amplitude.*

2.4 Why Neural Networks

Classification of remote sensing data has traditionally been performed by Bayesian and other statistically-based classifiers. The drawback to these methods is that the underlying distribution of the data needs to be assumed, and that the classifier is optimal only if these assumptions are correct. A neural network, however, does not require that a particular form of the statistical distribution be assumed, nor does it require that the relationship between the estimator inputs and outputs be known. The neural network system is able to draw its own input-output relationships directly from the data Another important feature of neural networks is their capability to perform function approximation. It has been shown in the literature [16], [17], [18] that a neural network with a single hidden layer having non-linear activation functions is capable of approximating any real-valued continuous function. Multilayer, feedforward neural networks therefore form a class of universal approximators.

The accuracy of the linear regression technique described in Section 2.3.1 is limited by two factors; the noise in the data and the degree to which the relationship between the 118-GHz spectral data and cell-top altitude is non-linear. The non-linear statistical estimator described in Section 2.3.2 is limited by the noise in the data and the degree to which the 118-GHz data can be described by second-order statistics. The accuracy of a neural network retrieval system is limited simply by the noise in the data. The complexity of the input-output relationship and the underlying statistics of the data do not impose additional limits on a neural-network-based retrieval system's accuracy. This fact, combined with the function approximation capabilities of neural networks, indicates that a neural network based retrieval system will produce superior results when compared with linear regression and non-linear statistical estimation systems.

Chapter 3

Introduction to Artificial Neural Networks

3.1 Artificial Neurons

Artificial neural networks, or simply neural nets, are mathematical models which attempt to achieve good performance through interconnections of simple computational elements. In this sense, neural nets are a model of our understanding of biological nervous systems. The computation elements, or nodes, in neural networks perform typically nonlinear and analog computations. The output value of each node is computed in two steps. First, a weighted sum of the node inputs is computed, and a bias term is added. This yields the linear output of the node:

$$s_j = \sum_{k=1}^{N} w_{jk} x_k + w_{j0} \tag{3.1}$$

where s_j is the linear output of the jth node in the network, x_k is the kth input to node j and can either be the actual input to the neural network system, or the output from other neurons in preceding layers, w_{jk} is the weight connecting the kth input and the jth node, w_{j0} is the bias term of node j, and N is the total number of inputs to node j. This output is then passed through a nonlinear function, referred to as the activation function of the node. This quantity can be expressed as:

$$h_j = f(s_j) = f\left(\sum_{k=1}^{N} w_{jk}x_k + w_{j0}\right) \tag{3.2}$$

The output of the activation function (h_j) is referred to as the activation level of node j. Figure 3-1 graphically illustrates a node.

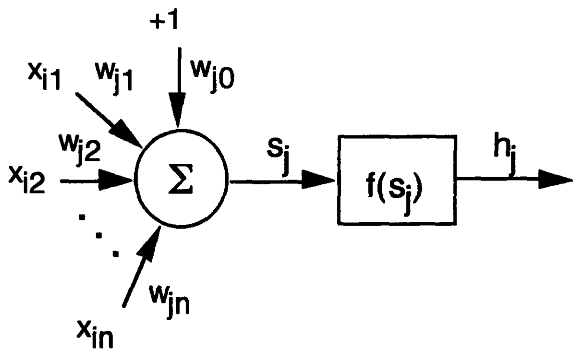

Figure 3-1: *Representation of the j*th *node in a neural network.*

The activation function ($f(s_j)$) can have many forms. Three common types of nonlinearities used for the activation function are illustrated in Figure 3-2; hard limiters, threshold logic elements, and sigmoidal nonlinearities. Nonlinear activation functions used at the output layer yield values in the range [-1,+1]. Since the desired output is generally not limited to this range, linear activation functions are typically used at the output layer.

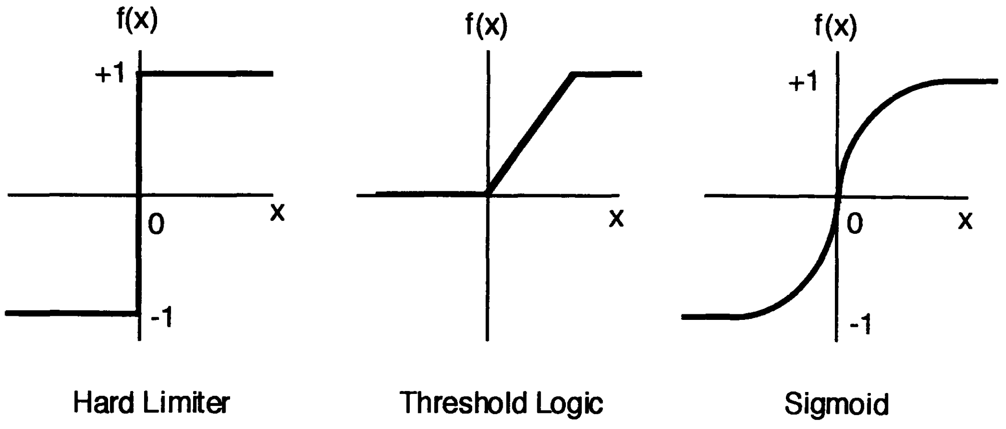

Figure 3-2: *Three common activation function nonlinearities.*

Most of the units in the cell-top altitude retrieval network have a sigmoidal shaped activation function, specifically the hyperbolic tangent ;

$$f(s_j) = \tanh(s_j) \tag{3.3}$$

This function is monotonically increasing and differentiable;

$$\frac{d\tanh(s_j)}{ds_j} = 1 - \tanh^2(s_j) \tag{3.4}$$

This property of the activation function will be exploited in the gradient-based training algorithm described in Section 3.3. Linear activation functions, $f(s_j) = s_j$, are used in the output nodes for the cell-top retrieval network.

3.2 Multilayer Feedforward Neural Networks

Neural network models are specified by the net topology, node characteristics, and the training algorithm used to determine the network weights and biases. The networks used in the cell-top altitude retrieval system are multilayer, feedforward neural networks. The connections between the computational elements are strictly forward. That is, no element can provide input to itself or to any other element that affects its input signals. A network is comprised of layers of these simple computational elements operating in parallel. The output of the nodes of the network make up three different types of layers in a multilayer network. The system input variables comprise the input layer of the network. The output layer of the network is comprised of nodes which compute the output variables. Hidden layers are comprised of variables that are not directly accessible to the outside world (they are neither system input nor output variables). Figure 3-3 illustrates a schematic of a fully connected network. The shaded circles in Figure 3-3 represent a single node in the network. The solid dots represent the output variables of each of the nodes.

Neural networks are able to represent a function by an interconnected set of neurons which have learned the appropriate response to a set of inputs. During the training phase of a network, patterns are sequentially presented to the network. After all patterns have been presented, the interconnecting weights (w_{jk}) of each neuron are adjusted so that the functional approximation created by the network minimizes the squared-error between the desired output and the output produced by the network. The neural network can be thought of as a type of nonlinear, least-mean-square interpolation formula for the set of points in the training set [19].

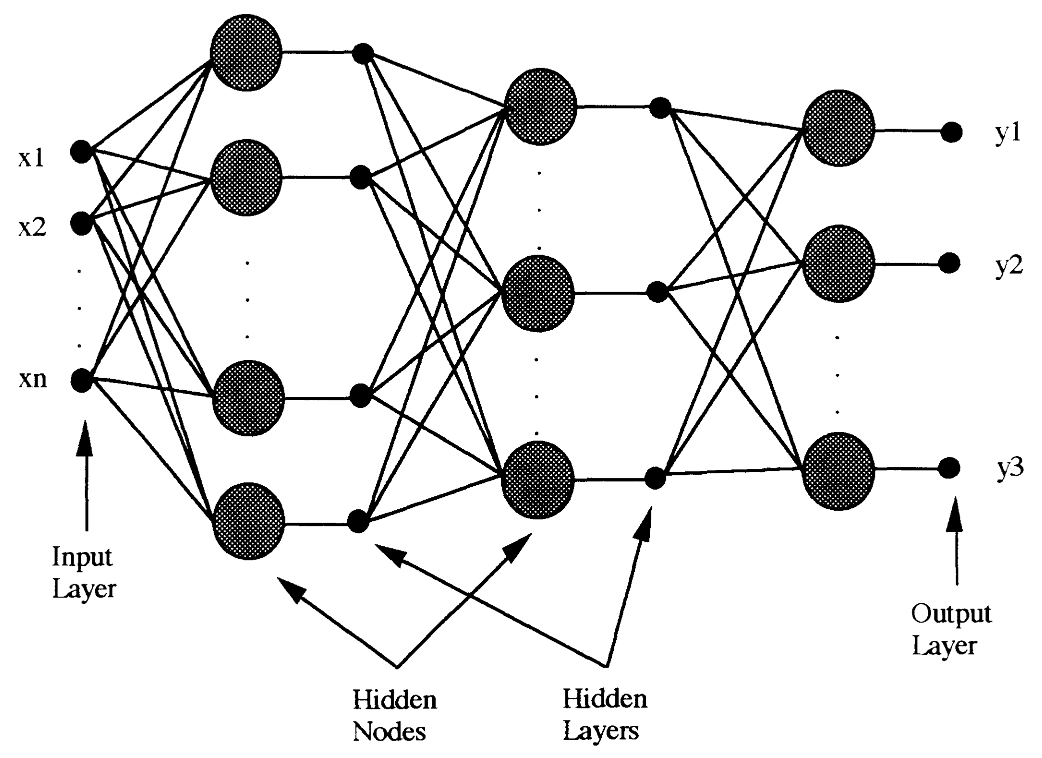

Figure 3-3: *A schematic of a multilayer feedforward neural network.*

3.3 Training of Neural Networks by Backpropagation

Given a neural network topology, the weights and biases must be determined to minimize the mean-squared error between the desired and calculated output variables. The backpropagation algorithm developed by Rumelhart et. al. [20] has proven to be very successful in training multilayer, feedforward neural networks. This algorithm is a generalization of the delta rule and involves the presentation of a set of pairs of input and output patterns to the neural network. The system first uses the input vector to calculate its own output vector, and then compares this with the desired, or target, vector. If the network produces the correct output vector, no learning takes place. Otherwise, the weights and biases are adjusted to reduce the error between the actual and desired output. The rule for changing the weights following the presentation of input/output pair p, for the simple case of a network with no hidden nodes, is given by;

$$\Delta_p w_{ij} = \eta(t_{pj} - o_{pj})i_{pi} = \eta\delta_{pj}i_{pi} \tag{3.5}$$

where t_{pj} is the target output for the jth component of the output pattern for input pattern p, o_{pj} is the jth component of the actual output produced by the network for input pattern p, i_{pi} is the ith component of the input pattern, $\delta_{pj} = t_{pj} - o_{pj}$, η is the learning rate, which determines how large a change can be made to the weight, and $\Delta_p w_{ij}$ is the change made to the weight between the ith and jth unit following the presentation of pattern p.

The rule described by Equation (3.5) minimizes the squared error between the actual and target output variables summed over the output units and all pairs of input/output vectors. To prove this, it can be shown that the derivative of the error with respect to each weight is proportional to the weight change dictated by the delta rule. This corresponds to performing steepest descent on a surface in weight space where the height of the surface is equal to the error. This will be shown by first dealing with linear activation functions. Specifically, let

$$E_p = \frac{1}{2}\sum_j (t_{pj} - o_{pj})^2 \tag{3.6}$$

be the measure of the error of input/output pattern p and let $E = \sum E_p$ be the overall measure of the error. To show that the delta rule implements gradient descent on E, it must be shown that

$$-\frac{\partial E_p}{\partial w_{ij}} = \delta_{pj}i_{pi} \tag{3.7}$$

which is proportional to Δw_{ij} by the delta rule. When there are no hidden units in a network, this derivative can be computed easily. The chain rule can be used to write the derivative as the derivative of the error with respect to the output times the derivative of the output with respect to the weight.

$$-\frac{\partial E_p}{\partial w_{ij}} = \frac{\partial E_p}{\partial o_{pj}} \frac{\partial o_{pj}}{\partial w_{ij}} \tag{3.8}$$

From Equation (3.6);

$$\frac{\partial E_p}{\partial o_{pj}} = -(t_{pj} - o_{pj}) = -\delta_{pj} \tag{3.9}$$

Therefore, the contribution of the jth unit to the error is simply proportional to δ_{pj}. Also, since we are dealing with linear units;

$$o_{pj} = \sum_i w_{ij} i_{pi} \tag{3.10}$$

Therefore;

$$\frac{\partial o_{pj}}{\partial w_{ij}} = i_{pi} \tag{3.11}$$

Substituting into Equation (3.8), we get the desired result;

$$-\frac{\partial E_p}{\partial w_{ij}} = \delta_{pj} i_{pi} \tag{3.12}$$

Given that $\frac{\partial E}{\partial w_{ij}} = \sum_p \frac{\partial E_p}{\partial w_{ij}}$ we can conclude that the net change in w_{ij} is proportional to this derivative and the delta rule therefore implements gradient descent on the error surface E.

Now, networks with nonlinear hidden units must be addressed. The output of a node can be expressed as;

$$net_{pj} = \sum_i w_{ij} o_{pi} \tag{3.13}$$

where $o_{pi} = i_{pi}$ if unit i is an input unit or $o_{pj} = f_j(net_{pj})$ for other units. The backpropagation algorithm requires that $f()$ is differentiable and non decreasing. To determine the generalized delta rule, we must set

$$\Delta_p w_{ij} \propto -\frac{\partial E_p}{\partial w_{ij}} \tag{3.14}$$

where E is the same error function described by Equation (3.6). As done earlier, the derivative in Equation (3.14) will be expressed as the product of two derivatives;

$$\frac{\partial E_p}{\partial w_{ij}} = \frac{\partial E_p}{\partial net_{pj}} \frac{\partial net_{pj}}{\partial w_{ij}} \tag{3.15}$$

Using Equation (3.14) we can write the second partial derivative in Equation (3.15) can be written as;

$$\frac{\partial net_{pj}}{\partial w_{ij}} = \frac{\partial}{\partial w_{ij}} \sum_k w_{kj} o_{pk} = o_{pi} \tag{3.16}$$

Now, if we define $\delta_{pj} = -\frac{\partial E_p}{\partial net_{pj}}$, we can write Equation (3.15) as;

$$-\frac{\partial E_p}{\partial w_{ij}} = \delta_{pj} o_{pi} \tag{3.17}$$

Therefore, to perform gradient descent on E we should make the weight changes according to;

$$\Delta_p w_{ij} = \eta \delta_{pj} o_{pi} \tag{3.18}$$

Now we must determine a way to calculate δ_{pj} for each unit in the network. We will again apply the chain rule to $\delta_{pj} = -\frac{\partial E_p}{\partial net_{pj}}$;

$$\delta_{pj} = -\frac{\partial E_p}{\partial net_{pj}} = -\frac{\partial E_p}{\partial o_{pj}} \frac{\partial o_{pj}}{\partial net_{pj}} \tag{3.19}$$

Given that $o_{pj} = f_j(net_{pj})$, the second factor in the above derivative can be expressed as;

$$\frac{\partial o_{pj}}{\partial net_{pj}} = f_j'(net_{pj}) \tag{3.20}$$

which is simply the derivative of the activation function of unit j evaluated at the network input, net_{pj}, to that unit. To compute the first factor in Equation (3.19) two cases must be considered. First, assume that the unit under consideration is an output unit. In this case, from the definition of E_p we can write;

$$\frac{\partial E_p}{\partial o_{pj}} = -(t_{pj} - o_{pj}) \tag{3.21}$$

Substituting Equations (3.21) and (3.20) into Equation (3.19);

$$\delta_{pj} = (t_{pj} - o_{pj})f_j'(net_{pj}) \tag{3.22}$$

If the unit under consideration is not an output unit, we can again use the chain rule to obtain;

$$\sum_k \frac{\partial E_p}{\partial net_{pk}} \frac{\partial net_{pk}}{\partial o_{pj}} = \sum_k \frac{\partial E_p}{\partial net_{pk}} \frac{\partial}{\partial o_{pj}} \sum_i w_{ik} o_{pi} = \sum_k \frac{\partial E_p}{\partial net_{pk}} w_{jk} = -\sum_k \delta_{pk} w_{jk} \tag{3.23}$$

Using this result, we can substitute into Equation (3.19) to obtain;

$$\delta_{pj} = f_j'(net_{pj}) \sum_k \delta_{pk} w_{jk} \tag{3.24}$$

These results can be summarized by the following three equations which specify how to compute all of the δ's in a network, which are then used to compute the weight changes in a network. This procedure makes up the generalized delta rule for a feedforward neural network.

$$\begin{aligned} &\Delta_p w_{ij} = \eta \delta_{pj} o_{pi} \\ &\delta_{pj} = (t_{pj} - o_{pj})f_j'(net_{pj}) \qquad \text{for output nodes} \\ &\delta_{pj} = f_j'(net_{pj}) \sum_k \delta_{pk} w_{jk} \qquad \text{for non - output nodes} \end{aligned} \tag{3.25}$$

In summary, the application of the generalized delta rule involves two phases. First, the input is presented and propagated through the network to determine the output value for each unit (o_{pj}). This output is compared with the targets resulting in an error signal for each output unit (δ_{pj}). Second, a backward pass is taken through the network to propagate the error signal back to each unit in the network and appropriate weight changes are made.

Chapter 4

Development of the Neural Network Based Altitude Retrieval System

4.1 Problem Description

The precipitation cell-top altitude retrieval system involves the estimation of altitude from 118-GHz spectral data. Due to the uncertainty of the relationship between cell-top altitude and the 118-GHz data, the measurement noise present in the 118-GHz data, and the unknown statistical distribution of the 118-GHz data, traditional Bayesian estimation techniques are not very successful. The estimation system proposed in this thesis utilizes the power of multilayer, feedforward neural networks to provide a more nearly optimal model of the 118-GHz cell-top altitude relationship. Recent research has proven that these networks are capable of such input-output mappings [21].

4.2 System Design

The development of a neural network retrieval system involves the selection of the neural network attributes and the training algorithm. The attributes of the network that need to be determined are the network model, network topology and training algorithm. This thesis explores the use of multilayer, feedforward neural networks trained by the backpropagation algorithm described in Section 3.3. Therefore, only network topology (how many hidden layers, how many nodes in each layer) must be determined. There are no design rules that indicate an optimal network topology for a specific application. The optimal topology must be determined by experimentation. The performance of a number of different network topologies will be explored in following sections. The performance criterion used is the mean-squared error between the target and actual cell-top altitudes. The size of the network will be limited by the size of the training set. A network with a large number of nodes will require a large data set to satisfactorily constrain the parameters of the network. If a limited data set is available for training, the network will overfit to the training data and have trouble generalizing its results to data not used in training.

4.2.1 Description of Neural Network Model

The retrieval system developed in his thesis consists of a multilayer feedforward neural network with one or more hidden layers. Each node in the network computes a weighted sum of its inputs and adds a bias term. Hidden layer nodes pass this weighted sum through a hyperbolic tangent activation function. Output layer nodes are linear. The network input is processed to have zero-mean and is peak-to-peak normalized. The network output is the estimate of cell-top altitude associated with the presented input vector.

The backpropagation algorithm was used to modify the weights and biases of the network. To improve upon the backpropagation algorithm, momentum and an adaptive learning rate were implemented. The use of momentum decreases the sensitivity to small details in the error surface, and helps prevent the network from converging to a local rather than global minimum. This is accomplished by allowing the network to respond not only to the local gradient, but also to recent trends in the error surface. Momentum was added to the backpropagation algorithm by making the weight changes equal to the sum of a fraction of the last weight change and the new change suggested by the backpropagation rule described in Section 3.3 by Equations (3.25). This can be expressed as;

$$\Delta w_{ij} = \mu \Delta_{-1} w_{ij} + (1-\mu)\eta \delta_j i_i \tag{4.1}$$

where Δw_{ij} is the suggested current weight change, $\Delta_{-1} w_{ij}$ is the previous weight change, μ is the momentum coefficient and $\eta \delta_j i_i$ is the weight change suggested by the backpropagation rule. The new weights and biases are rejected if they result in too large an increase in mean-square error. This prevents the network from being pushed out of a deep minimum in the error surface.

An adaptive learning rate was used to allow the network to train faster by determining an optimal learning rate for the local terrain in the error surface. When a larger learning rate results in stable learning the learning rate is increased. However, when the learning rate is too high to guarantee a decrease in error it is decreased until stable learning resumes.

4.2.2 Description of Data Sets

The mapping from delta brightness spectra ($\Delta \overline{T}_B$) to cell-top altitude is accomplished by training a neural network with a subset of randomly shuffled $\Delta \overline{T}_B$ and associated cell-top altitudes. This subset of spectra is called the training set. A test set, created from the remaining subset of $\Delta \overline{T}_B$ spectra, was used to validate the network's performance on input-output pairs not previously seen by the network. Four types of data sets were used to train four different neural network cell-top altitude estimators. The data sets were derived from a collection of 279 independent near-nadir brightness temperature spectra compiled during

GALE and COHMEX. The measurement instrument and flights are described in Section 2.2. The first data set consisted of 176 $\Delta\overline{T}_B$ spectra from the full collection of observed cloud types (both cumulus and mature). It has been shown that the 118-GHz channels are insensitive to some dense cirrus anvils [22]. However, the visible spectrum is highly sensitive to the cirrus anvils, and therefore optical estimates of cell-top altitude are higher than the retrieved 118-GHz cell-top altitudes. For this reason, the remaining data sets limit the observations to only cumulus cloud types which typically do not display these cirrus shields. This reduced the data set from 176 spectra to 84 spectra. Data set three consisted of cumulus-only data with both $\Delta\overline{T}_B$ and clear-air reference spectra ($\overline{T}_{Br}$) as input. Data set four incorporated the logarithm of cell-size in addition to $\Delta\overline{T}_B$ and $\overline{T}_{Br}$ as input to the neural network estimator. Table 4.1 summarizes the data sets used in the development of the neural network retrieval system.

Input Data	**# Patterns in Training Set**	**# Patterns in Test Patterns**
1) All Cloud Types; $\Delta\overline{T}_B$	117	59
2) Cumulus-Only Clouds; $\Delta\overline{T}_B$	56	28
3) Cumulus-Only Clouds; $\Delta\overline{T}_B, \overline{T}_{Br}$	56	28
4) Cumulus-Only Clouds; $\Delta\overline{T}_B, \overline{T}_{Br}$,log(cell-size)	56	28

Table 4.1: *Summary of data sets used in development of cell-top altitude retrieval system.*

4.2.3 Topology Comparison

To determine the optimal neural network retrieval system, a number of network topologies had to be investigated. It has been shown that networks with two hidden layers and a sufficient number of nodes are capable of approximating any well behaved function. It has also been shown that neural networks with one hidden layer and a sufficient number of nodes are capable of universal function approximation [23],[24],[25]. Therefore, only networks with one and two hidden layers needed to be investigated. Seven network topologies were trained with a data set consisting of 117 $\Delta\overline{T}_B$ spectra from the combined mature and cumulus cloud types. The performance of the networks was based on the resulting rms error of the training and test sets. The test set consisted of 59 $\Delta\overline{T}_B$ spectra taken from the same collection of data as the training set. Since the training and test sets are mutually exclusive, the test set results show how well the network can generalize given similar data that it has not been trained on.

First, networks with two hidden layers were investigated. A network with six nodes in the first hidden layer and four nodes in the second hidden layer was tested first. The number of nodes in each of the hidden layers was increased until the performance of the test set began to degrade. A single hidden layer network with four hidden nodes was then evaluated. The number of hidden layer nodes was again increased until the performance of the test set began to degrade. The training of each network was stopped when the training error converged to a minimum value. For each network topology, the resulting training and test set rms errors were noted, as well as the number of presentations of the training set that was required for the network to converge to a minimum rms error value. Table 4.2 summarizes the results from the topology comparisons.

Description of Neural Network Topology	Training Set RMS Error	Test Set RMS Error	# of Training Epochs Required
2 hidden layers: 6 and 4 nodes	1.81	1.78	3000
2 hidden layers: 8 and 4 nodes	1.70	1.79	3500
2 hidden layers: 8 and 6 nodes	1.71	1.79	4000
1 hidden layer: 4 nodes	1.77	1.80	2500
1 hidden layer: 5 nodes	1.74	1.75	3500
1 hidden layer: 6 nodes	1.71	1.76	5000
1 hidden layer: 7 nodes	1.72	1.75	5000

Table 4.2: *Comparison of RMS errors (km) from different neural network topologies.*

From these topology experiments, it was determined that a network with one hidden layer, with five nodes, was the optimal network topology for this data set. This network topology yields acceptable rms error results and performs as well on the test set as the training set. Figure 4.1 illustrates the performance of the network on both the training and test sets as it was training. Because the data sets available for training the networks were small, networks with two hidden layers performed well on the training set, but did not generalize well to independent test data sets. Generalization issues will be discussed in Section 4.2.4.

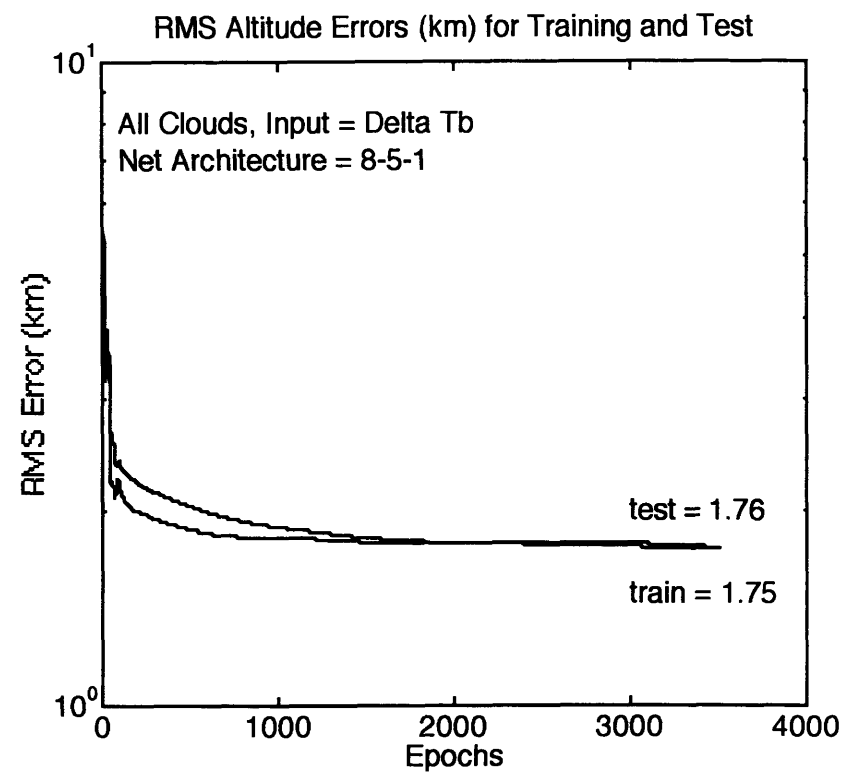

Figure 4-1: *Performance of neural network retrieval system with one hidden layer with five nodes on training and test set.*

4.2.4 Development of the Incremental Neural Network

One typical problem encountered when developing neural network systems is that of overfitting to the training data. This occurs when the network has too much power. We want a network that has enough power to provide a good fit to the data, but not so much that it overfits. The problem of overfitting occurred in the development of the neural network estimator for the reduced data set consisting of cumulus only clouds. When the additional inputs of $\overline{T}_{Br}$ and log(cell-size) were added to the system, the network acquired 45 new weights (8+1 new inputs x 5 hidden nodes). These additional weights provided enough power to enable the network to easily fit very well to the training data. However, the performance of the network on the test data was severely degraded. Figure 4-2 illustrates the performance of a network with 17 inputs (8 $\Delta\overline{T}_B$ + 8 $\overline{T}_{B_r}$ + 1 log(cell-size)) and one hidden layer containing 5 hidden nodes. While the performance of the training set is excellent, the test set results quickly diverge as the network overfits to the training data. As we are most interested in the network's performance on data that it has never seen before, a network system that behaves as shown in Figure 4-2 is unacceptable.

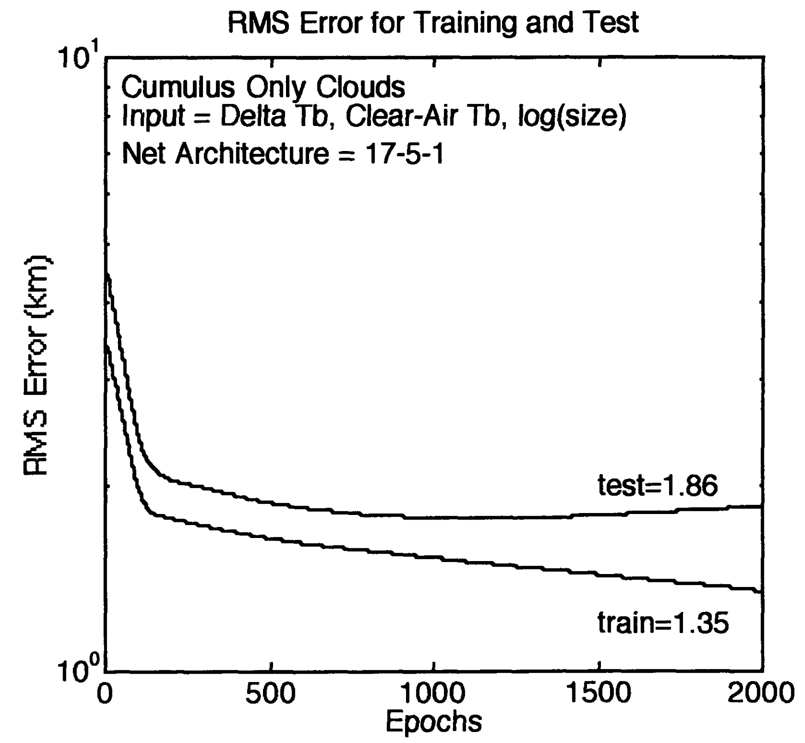

Figure 4-2: *Performance of neural network retrieval system that is overfit to the training data.*

There are a number of options available to overcome overfitting to the training data. Two methods that are typically used are 1) limiting the number of hidden nodes, therefore limiting the number of weights or 2) limiting the number of epochs of training. Both of these methods were investigated to try to prevent overfitting for data sets three and four. Although both of these methods helped with the problem of overfitting, the resulting performance of the network was not increased when compared to systems with only the perturbation brightness temperature spectra ($\Delta\overline{T}_B$) as input. To combat the overfitting problem while increasing the performance of the neural network estimation system an "incremental neural network" training algorithm was developed.

It was postulated that most of the cell-top altitude information was contained in the $\Delta\overline{T}_B$ spectra. Therefore, when the clear-air reference spectra ($\overline{T}_{Br}$) and log(cell-size) were added as input to the neural network, the minimum acceptable performance would be that given by the network with only $\Delta\overline{T}_B$ as input. To accomplish this, a network using $\Delta\overline{T}_B$ as input was initially trained using a network with one hidden layer with four nodes until the network rms error converged to a minimum. The additional inputs ($\overline{T}_{Br}$, log(cell-size)) were then added to the network and connected to one additional hidden node, making a total of five hidden nodes. Connections were made from the original $\Delta\overline{T}_B$ inputs to the new hidden node. However, the new inputs were *not* connected to the original four nodes. The weights and biases associated with the initially trained network were held constant, while the backpropagation algorithm was used to train the new weights and biases. The incremental neural network approach is illustrated graphically in Figure 4-3. The original weight matrix is shown with a white background. The additional weight matrix is shown with a shaded background.

The topology illustrated in Figure 4-3 has a total of 54 weights. If a fully connected network with five hidden nodes was used, the network would have a total of 90 weights. By dramatically decreasing the number of weights the network has to work with and by training the network in this incremental fashion, the problem of overfitting is solved while increasing the network performance.

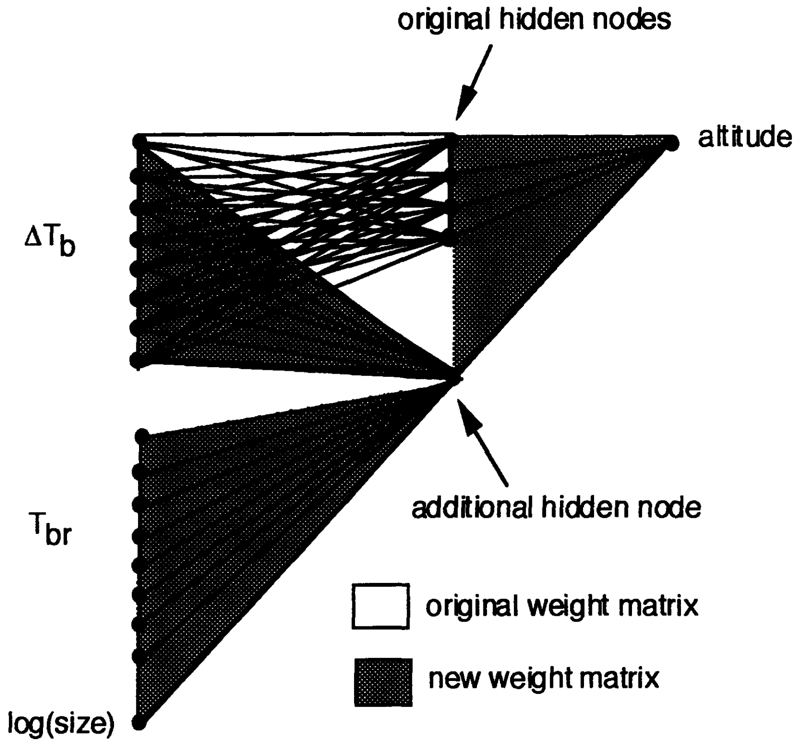

Figure 4-3: *Illustration of incremental neural network system.*

4.3 Summary of Results

To rate the performance of the neural network estimator it was compared with the simple linear regression analysis described in Section 2.3.1 and the non-linear statistical estimator described in Section 2.3.2. The neural network estimator used the optimal architectures determined in Sections 4.2.3 and 4.2.4. A single hidden layer network with five hidden nodes was used for the data sets with $\Delta\overline{T}_B$ only as input. The incremental neural network was used for the data sets with additional inputs. Table 4.3 shows the resulting rms altitude errors (km) for the three methods. Additionally, the apriori variance in the altitude data is listed. The error reported for the neural network estimator is the rms error associated with the test set as described in Section 4.2.2. The error reported for the linear regression estimator is the rms error associated with the entire data set. The non-linear statistical estimator error shown was computed by Gasiewski et al. [26], and also is the rms error associated with the entire data set.

Data Set Used	Apriori Variance	Linear Regression	NonLinear Statistical Estimator	Neural Network Estimator
1) All Cloud Types; $\Delta\bar{T}_B$	3.53	2.03	1.97	1.76
2) Cumulus Only Clouds; $\Delta\bar{T}_B$	3.53	1.82	1.63	1.44
3) Cumulus Only Clouds; $\Delta\bar{T}_B, \bar{T}_{Br}$	3.53	1.66	1.53	1.41
4) Cumulus Only Clouds; $\Delta\bar{T}_B, \bar{T}_{Br}$,log(cell-size)	3.53	1.58	1.50	1.36

Table 4.3: *Comparison of rms errors (km) for cell-top altitude estimators.*

As illustrated by Table 4.3, the neural network estimator outperforms both the linear regression and nonlinear statistical estimators by ~10.7-13.3% for data set 1, ~11.7-20.9% for data set 2, ~7.8-15.1% for data set 3 and ~9.3-13.9% for data set 4.

4.3.1 Images of Retrieved Cell-Top Altitude

Cell-top altitude imagery was created from the output of the neural network estimator. 118-GHz CAMEX spectral data gathered September-October 1993 was evaluated by the neural network cell-top estimator and the results were plotted. The altitudes produced by the network show expected cell morphology. Figure 4-4 shows two samples of imagery of the retrieved cell-top altitudes.

IMAGES OF PRECIPITATING CELL RETRIEVED ALTITUDES (km)

CAMEX 118-GHz data collected October 5, 1993

cloud 1: latitude: 27.25 longitude: -80.12

cloud 2: latitude: 26.21 longitude: -80.64

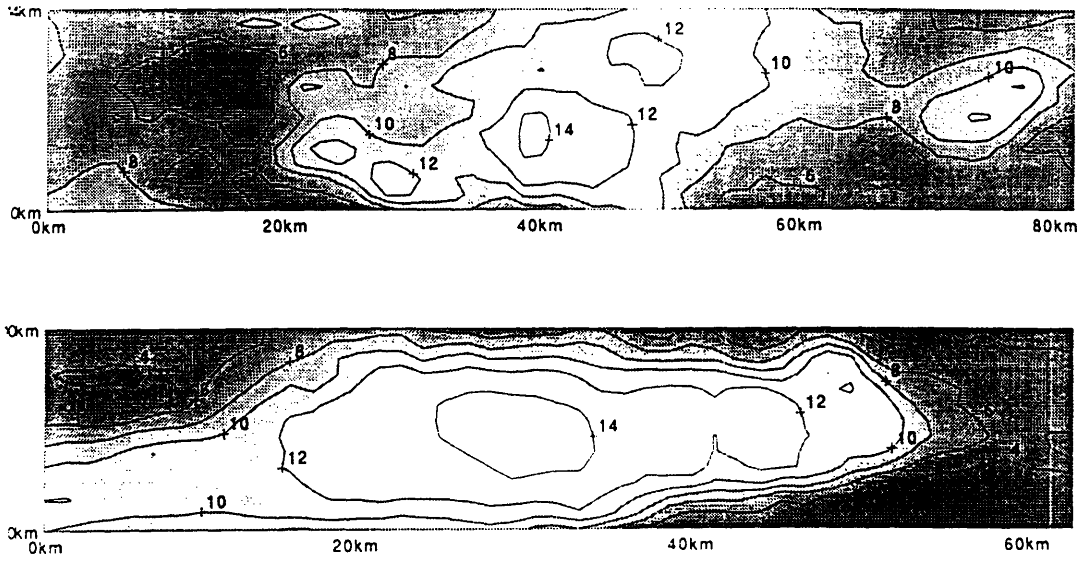

Figure 4-4: *Images of retrieved precipitating cell-top altitudes.*

The accuracy of the images shown in Figure 4-4 was determined by optical estimation of cell-top altitude from the video imagery. For near-nadir scan angles, the variance in the cell-top altitude data that was optically estimated from the video imagery is 1.35 km. The network produced an error of 0.84 km which is an improvement of ~37.8%. For off-nadir scan angles, the variance in the optically estimated cell-top altitude data is 1.29 km. The network produced an error of 1.19 km which is an improvement of ~7.8%. Figure 4-5 shows the estimation error as a function of scan angle. The decrease in performance of the neural network estimator for off-nadir scan angles can be explained in two ways. First, the neural network was trained on near-nadir data. Therefore, it is expected that the network will produce more accurate results with similar data. Second, the video imagery of the CAMEX flight showed that the distinct cell-top peaks occurred at near-nadir scan angles, and the off-nadir scan angles showed an increased amount of cirrus cover. Therefore, as explained in Section 4.2.2, the off-nadir cell-top optical estimates may be higher than indicated by the 118-GHz spectra data.

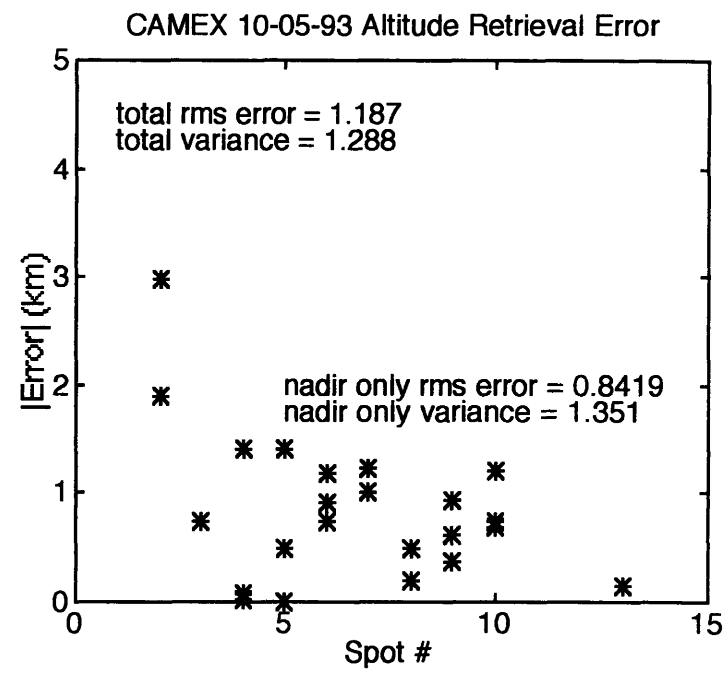

Figure 4-5: *Cell-top altitude retrieval error versus scan angle.*

A second example of the utility of the neural network cell-top estimator is shown in Figure 4-6. Cell-top altitude retrievals of Cyclone Oliver (February 7, 1993) were created using the neural network estimator developed for data set 1. Although the retrievals can not be verified with the video imagery because the 118-GHz data was gathered at night, the retrieved images are as expected. However, the retrievals from Cyclone Oliver will be less accurate than the CAMEX retrievals because the training data did not consist of any flights over tropical areas. These considerations aside, the retrieved images are still quite useful. The morphology of the eyewall and the surrounding precipitating cells are clearly visible.

IMAGES OF RETRIEVED CELL TOP ALTITUDES (km)
FROM HURRICANE OLIVER (Feb. 7, 1993)

latitude: -17.50 longitude: 151.88

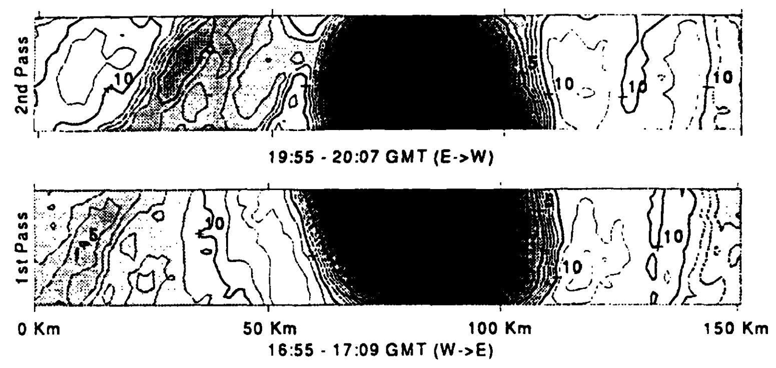

Figure 4-6: *Retrieved cell-top altitudes from Cyclone Oliver.*

Chapter 5

Conclusions / Suggestions for Further Research

Multilayer feedforward neural networks have proven to be a viable method for estimating precipitating cell-top altitudes from 118-GHz imagery. When compared to linear and nonlinear retrieval methods, the neural network yielded superior results. This may be attributed to the fact that the neural network is able to capture not only the nonlinear relationship that exists between the 118-GHz brightness temperatures and cell-top altitude, but also the complex statistics of the 118-GHz data. In addition, the output of the cell-top estimator can be used to produce imagery that displays cell morphology in a useful way.

Additional research can be done to build upon the neural network cell-top estimator developed in this thesis. The use of an expanded training set would produce more robust networks that would perform well on a number of different data sets. In addition, the error results reported for the neural network estimator have been total rms error, and may contain a bias produced by the estimator. If this bias was identified and removed, the neural network system may be additionally improved.

The neural network retrieval system could be expanded to use the relationship that exists between cell-top altitude and rainfall rate to predict not only the cell-top altitude, but also the rainfall rate. Finally, to help discover why the neural network system outperforms the nonlinear statistical estimator described in Section 2.3.2, experiments could be performed to determine an optimal algorithm to compress the 118-GHz spectral data using a neural network.

Appendix A

Development Code for the Neural Network Estimator

The neural network cell-top estimation system was developed on a 486-based PC using MATLAB, version 4.0, and the MATLAB Neural Network Toolbox, version 1.0. The code developed for the system is listed below.

GETINPUT.M

```
% load the data file containing the desired info and parse thru to gather input of interest

% This file needs only to be run ONCE for full data and ONCE for cumulus data - once set
up, keep the same shuffle....

% Use the following 2 lines for full data
load cloud.dat
clouddata = cloud;

% Use the following 2 lines for cumulus only data
%load cumulus.dat
%clouddata = cumulus;

[R,C] = size(clouddata);
altitude = clouddata(:,1);
diameter = clouddata(:,2);
deltat = clouddata(:,4:11);
cleart = clouddata(:,13:20);
clear clouddata;
clear cloud;

logdia = log(diameter);
index = 1:1:R;
input = [index' deltat cleart logdia];
```

```
% Shuffle data
altitude = [index' altitude];
[input,altitude] = shuffle(input',altitude');

% Now, input contains delta Tb's, clear air Tb's
% and ln(diameter)'s

% In parsecld, strip off the input data you actually want to use and prepare for net...

parsecld;
```

SHUFFLE.M

```
function [newA,newB] = Shuffle(A,B)

[R,C] = size(A);

% index will tell which columns have already been shuffled
index = 1:C;
index = index*0;

newA = [];
newB = [];

i=0;
while i<C
  x= floor(rand(1) * C + 1);
  while (index(x) == 1)
    x = floor(rand(1) * C + 1);
  end
  index(x) = index(x) + 1;
  newA = [newA(:,:),A(:,x)];
  newB = [newB(:,:),B(:,x)];
  i= i + 1;
end
```

PARSECLD.M

```
% Get the desired input patterns and prepare data sets for net

% Determine which inputs you want

% Delta Tb only
netinput = input(2:9,:);

% Delta Tb and Clear Tb
%netinput = input(2:17,:);

% Delta Tb, Clear Tb, ln(diameter)
%netinput = input(2:18,:);
```

```
n = 8;
P = netinput;

MeanOutVec = zeros(1,1);
NormOutVec = ones(1,1);
MeanInVec = zeros(1,n);
NormInVec = ones(1,n);

% Split set into test and training, and process data so zero mean and normalized

num = 4;
den = 5;

[R,C] = size(P);
[TrainPatterns,MeanInVec,NormInVec] = meannorm(P(1:n,1:floor(num*C/den))');
TestIn = P(1:n,(floor(num*C/den)+1):C);
[T,MeanOutVec,NormOutVec] = meannorm(altitude(2,1:floor(num*C/den))');
TestOut = altitude(2,(floor(num*C/den)+1):C);

% Must compensate test vectors for mean and normalization procedures...

[row,col] = size(TestIn);
for i = 1:row
 TestPatterns(i,:) = TestIn(i,:)-MeanInVec(i);
 TestPatterns(i,:) = TestPatterns(i,:)/NormInVec(i);
end

[row,col] = size(TestOut);
for i = 1:row
 T2(i,:) = TestOut(i,:)-MeanOutVec(i);
 T2(i,:) = T2(i,:)/NormOutVec(i);
end

clear row col i n TestIn TestOut num den P R C;
disp('Done parsing cloud data')
```

MEANNORM.M

```
function [x,meanvec,normvec] = meannorm(A)

% This function subtracts the mean from the columns of A
% then p-p normalizes the columns

% Need to subtract the mean of each channel out

meanvec = mean(A);
[r,c] = size(A);
M = meanvec'*ones(1,r);
A = A-M';

% Let's try just p-p normalizing the data

for i = 1:c
```

```
 normvec(i) = max(abs(A(:,i)));
 A(:,i) = A(:,i)./normvec(i);
end

x = A';
```

BEP2PLS.M

```
% This file trains a two layer network

P = TrainPatterns;

% Initialize RMS Error record

RMSTrainTotal = [];
RMSTestTotal = [];
tepoch = 0;

% INITIALIZE NETWORK ARCHITECTURE
%===============================
% Set input vector size R, layer sizes S1 & S2, batch size Q.
disp('Initializing Weight Matrix')
[R,Q] = size(P);[S2,Q] = size(T); S1 = 7;

% Initialize weights and biases.
[W1,B1] = nwtan(S1,R);
W2 = rands(S2,S1)*0.5;
B2 = rands(S2,1)*0.5;

% TRAINING PARAMETERS
disp_freq = 50;
max_epoch = 6000;
err_goal = 0.02;
lr = 0.001;
lr_inc = 1.001;
lr_dec = 0.999;
momentum = 0.95;
err_ratio = 1.04;

f1 = 'tansig';
f2 = 'purelin';

disp('Starting to train')
TP = [disp_freq max_epoch err_goal lr lr_inc lr_dec momentum err_ratio];
[ W 1 , B 1 , W 2 , B 2 , e p o c h , T R , T R T ]          =
trainbps(W1,B1,f1,W2,B2,f2,P,T,TestPatterns,T2,MeanOutVec,NormOutVec,TP);

pack;

disp('Done Training')
tepoch = tepoch + epoch;
[row,col] = size(T);
RMSTrain = ((TR(1,:)./col)./row).^0.5;
```

```
[row,col] = size(T2);
RMSTest = ((TRT(1,:)./col)./row).^0.5;
RMSTrainTotal = [RMSTrainTotal RMSTrain];
RMSTestTotal = [RMSTestTotal RMSTest];

save bep.mat
clg;

% PLOT ERROR CURVE
%================
semilogy(1:tepoch,RMSTrainTotal(1,1:tepoch),'r',1:tepoch,RMSTestTotal(1,1:tepoch),'g--
')
title('RMS Error for Training and Test');
xlabel('Epochs');
ylabel('RMS Error');
pause

% PRESENTATION PHASE of Normalized vectors
[DesiredOut,NetOut] = present2(W1,B1,f1,W2,B2,f2,P,T,MeanOutVec,NormOutVec);

E = DesiredOut-NetOut;
disp('The RMS Error for Training Patterns is ');
SSE = sumsqr(E);
[row,col] = size(T);
RMS = ((SSE/col)/row)^0.5


% PLOT ERRORS ASSOCIATED WITH EACH OUTPUT VECTOR
%==========================================
barerr(NetOut,DesiredOut);
pause

% PRESENTATION PHASE of Test vectors

[ T D e s i r e d O u t , T N e t O u t ]                    =
present2(W1,B1,f1,W2,B2,f2,TestPatterns,T2,MeanOutVec,NormOutVec);

E = TDesiredOut-TNetOut;
disp('The RMS Error for Test Patterns is ');
SSE = sumsqr(E);
[row,col] = size(T2);
RMS = ((SSE/col)/row)^0.5


% PLOT ERRORS ASSOCIATED WITH EACH OUTPUT VECTOR
%==========================================
barerr(TNetOut,TDesiredOut);
pause

cmptrai2;
pause
cmptest2;
```

MBEP2PLS.M

```
% This function continues to train a network with initial parameters as given in the file
% bep.mat

load bep.mat

% TRAINING PARAMETERS
disp_freq = 50;
max_epoch = 1000;
err_goal = 0.02;
lr = 0.001;
lr_inc = 1.001;
lr_dec = 0.999;
momentum = 0.95;
err_ratio = 1.04;

f1 = 'tansig';
f2 = 'purelin';

disp('Starting to train')
TP = [disp_freq max_epoch err_goal lr lr_inc lr_dec momentum err_ratio];
[W1,B1,W2,B2,epoch,TR,TRT] =
trainbps(W1,B1,f1,W2,B2,f2,P,T,TestPatterns,T2,MeanOutVec,NormOutVec,TP);
%[W1,B1,W2,B2,TR] = trainbpx(W1,B1,f2,W2,B2,f2,P,T,TP);

disp('Done Training')

tepoch = tepoch+epoch;
[row,col] = size(T);
RMSTrain = ((TR(1,:)./col)/row).^0.5;
[row,col] = size(T2);
RMSTest = ((TRT(1,:)./col)/row).^0.5;
RMSTrainTotal = [RMSTrainTotal RMSTrain];
RMSTestTotal = [RMSTestTotal RMSTest];

pack;
save bep.mat

% PLOT LEARNING RATES
%===================
plotlr(TR(2,:));
pause

% PLOT ERROR CURVE
%================
semilogy(1:tepoch,RMSTrainTotal(1,1:tepoch),'r',1:tepoch,RMSTestTotal(1,1:tepoch),'g--
')
title('RMS Error for Training and Test');
xlabel('Epochs');
ylabel('RMS Error');
```

```
% ploterr(TR(1,:));
pause

% PRESENTATION PHASE of Training vectors

[DesiredOut,NetOut] = present2(W1,B1,f1,W2,B2,f2,P,T,MeanOutVec,NormOutVec);
E = DesiredOut-NetOut;
disp('The RMS Error for Training Patterns is ');
SSE = sumsqr(E);
[row,col] = size(T);
RMS = ((SSE/col)/row)^0.5

% PLOT ERRORS ASSOCIATED WITH EACH OUTPUT VECTOR
%=============================================
barerr(NetOut,DesiredOut);
pause

% PRESENTATION PHASE of Test vectors

[TDesiredOut,TNetOut] =
present2(W1,B1,f1,W2,B2,f2,TestPatterns,T2,MeanOutVec,NormOutVec);

E = TDesiredOut-TNetOut;
disp('The RMS Error for Test Patterns is ');
SSE = sumsqr(E);
[row,col] = size(T2);
RMS = ((SSE/col)/row)^0.5


% PLOT ERRORS ASSOCIATED WITH EACH OUTPUT VECTOR
%=============================================
barerr(TNetOut,TDesiredOut);
pause

cmptrai2;
pause
cmptest2;
```

TRAINBPS.M

```
function[a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p]=
trainbps(i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z,aa,bb,cc)
%TRAINBPX Trains a network with 1-3 layers with backpropagation
%     with momentum and an adaptive learning rate until
%     an error goal or a maximum training epoch is reached.
%      (See TRAINBP, TRAINBPA, TRAINBPM)
%
%       [NW1,NB1,TE,TR] = TRAINBMA(W1,B1,F1,P,T,TP)
%       W1 - S1xR weight matrix.
%       B1 - S1x1 bias vector.
%       F1 - the layer's transfer function.
%       P  - RxQ matrix of input vectors.
%        T  - S1xQ matrix of target vectors.
%        TP - row vector of 8 training parameters:
%            [disp_freq max_epoch err_goal lr
%             lr_inc lr_dec mom_const err_ratio]
%       Returns:
%        NW1 - a new weight matrix.
%        NB1 - a new bias vector.
%        TE  - the actual number of epochs trained.
%        TR - training record: [row of errors;
%                         row of learning rates]
%        TRT - test training error
%
%         [NW1,NB1,NW2,NB2,TE,TR] = TRAINBPA(W1,B1,F1,W2,B2,F2,P,T,TP)
%         ...trains a 2-layer network.
%
%         [NW1,NB1,NW2,NB2,NW3,NB3,TE,TR] =
%           TRAINBPA(W1,B1,F1,W2,B2,F2,W3,B3,F3,P,T,TP)
%         ...trains a 3-layer network.
%       M.H. Beale & H.B. Demuth, 1-31-92
%       Copyright (c) 1992 by the MathWorks, Inc.

if nargin == 6
 if length(n) ~= 8
  error('Wrong number of training parameters.')
  end
 if nargout == 3
  [a,b,c] = tbpx1(i,j,k,l,m,n);
 else
  [a,b,c,d] = tbpx1(i,j,k,l,m,n);
 end

elseif nargin == 13
 if length(u) ~= 8
  error('Wrong number of training parameters.')
  end
 if nargout == 5
  [a,b,c,d,e,f] = tbps2(i,j,k,l,m,n,o,p,q,r,s,t,u);
 else
  [a,b,c,d,e,f,g] = tbps2(i,j,k,l,m,n,o,p,q,r,s,t,u);
  end

elseif nargin == 16
 if length(x) ~= 8
  error('Wrong number of training parameters.')
  end
 if nargout == 7
  [a,b,c,d,e,f,g,h] = tbps3(i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x);
 else
  [a,b,c,d,e,f,g,h,i,j] = tbps3(i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x);
  end

elseif nargin == 15
 if length(w) ~= 8
  error('Wrong number of training parameters.')
  end
 if nargout == 9
  [a,b,c,d,e,f,g,h,i] = tbps4(i,j,k,l,m,n,o,p,q,r,s,t,u,v,w);
 else
  [a,b,c,d,e,f,g,h,i,j,k] = tbps4(i,j,k,l,m,n,o,p,q,r,s,t,u,v,w);
  end

elseif nargin == 21
 if length(cc) ~= 8
  error('Wrong number of training parameters.')
  end
 if nargout == 13
  [a,b,c,d,e,f,g,h,i,j,k,l,m] = tbps6(i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z,aa,bb,cc);
 else
  [a,b,c,d,e,f,g,h,i,j,k,l,m,n,o] = tbps6(i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z,aa,bb,cc);
  end
else
 error('Wrong number of arguments.');
 end
```

TBPS2.M

```
function [nw1,nb1,nw2,nb2,te,tr,trt] = ...
  tbps2(w1,b1,f1,w2,b2,f2,p,t,tin,t2,m,n,tp)
%TBPX2  Trains a 2-layer network with backpropagation
%       using momentum and an adaptive learning rate.
%        (Called by TRAINBPX)
%
%         [NW,NB,TE,TR]
%            = TBPX2(W1,B1,F1,W2,B2,F2,P,T,TP)
%         W1 - S1xR layer-1 weight matrix.
%         B1 - S1x1 layer-1 bias vector.
%         F1 - name of layer-1 transfer function (string).
%         W2 - S2xS1 layer-2 weight matrix.
%         B2 - S2x1 layer-2 bias vector.
%         F2 - name of layer-2 transfer function (string).
%         P  - RxQ matrix of input vectors.
%         T  - S2xQ matrix of target vectors.
%         TP - row vector of 8 training parameters:
%             [disp_freq max_epoch err_goal lr
%              lr_inc lr_dec mom_const err_ratio]
%        Returns:
%          NW1,NB1 - new weights & biases for layer 1.
%          NW2,NB2 - new weights & biases for layer 2.
%         TE - the actual number of epochs trained.
%         TR - training record: [row of errors;
%                          row of learning rates]

%        M.H. Beale & H.B. Demuth, 1-31-92
%       Copyright (c) 1992-93 by the MathWorks, Inc.

disp('I am using the new tpbs2');
```

```
TestPatterns = tin;
T2 = t2;
MeanOutVec = m;
NormOutVec = n;

if nargin ~= 13
  error('Wrong number of arguments.');
  end

% TRAINING PARAMETERS
df = tp(1);
me = tp(2);
eg = tp(3);
lr = tp(4);
im = tp(5);
dm = tp(6);
mc = tp(7);
er = tp(8);

% NETWORK PARAMETERS
W1 = w1;
B1 = b1;
DF1 = getdelta(f1);
W2 = w2;
B2 = b2;
DF2 = getdelta(f2);

dW1 = W1*0;
dB1 = B1*0;
dW2 = W2*0;
dB2 = B2*0;
MC = mc;

% PRESENTATION PHASE
A1 = feval(f1,W1*p,B1);
A2 = feval(f2,W2*A1,B2);

% compensate for subtraction of mean and normalization
[row,col] = size(A2);
for i = 1:row
 NetOut(i,:) = A2(i,:)*NormOutVec(i);
 NetOut(i,:) = NetOut(i,:)+MeanOutVec(i);
 DesiredOut(i,:) = t(i,:)*NormOutVec(i);
 DesiredOut(i,:) = DesiredOut(i,:)+MeanOutVec(i);
end

E = DesiredOut-NetOut;
SSE = sumsqr(E);

% TEST PRESENTATION PHASE
A1T = feval(f1,W1*TestPatterns,B1);
A2T = feval(f2,W2*A1T,B2);

% compensate for subtraction of mean and normalization
```

```
[row,col] = size(A2T);
for i = 1:row
 TNetOut(i,:) = A2T(i,:)*NormOutVec(i);
 TNetOut(i,:) = TNetOut(i,:)+MeanOutVec(i);
 TDesiredOut(i,:) = T2(i,:)*NormOutVec(i);
 TDesiredOut(i,:) = TDesiredOut(i,:)+MeanOutVec(i);
end

ET = TDesiredOut-TNetOut;
SSET = sumsqr(ET);

% TRAINING RECORD
TR = zeros(2,me);
TRT = zeros(1,me);
RMSTrain = zeros(2,me);
RMSTest = zeros(1,me);
SSE0 = SSE;
SSE0T = SSET;
[row,col] = size(t);
RMSSSE0 = ((SSE0/col)/row)^0.5;
[row,col] = size(T2);
RMSSSE0T = ((SSE0T/col)/row)^0.5;
lr0 = lr;

% BACKPROPAGATION PHASE
D2 = feval(DF2,A2,t-A2);
D1 = feval(DF1,A1,D2,W2);

for epoch=1:me

  % CHECK PHASE
  if SSE < eg, epoch=epoch-1; break, end

  % LEARNING PHASE
  [dW1,dB1] = learnbpm(p,D1,lr,MC,dW1,dB1);
  [dW2,dB2] = learnbpm(A1,D2,lr,MC,dW2,dB2);
  MC = mc;
  TW1 = W1 + dW1; TB1 = B1 + dB1;
  TW2 = W2 + dW2; TB2 = B2 + dB2;

  % PRESENTATION PHASE
  TA1 = feval(f1,TW1*p,TB1);
  TA2 = feval(f2,TW2*TA1,TB2);

% compensate for subtraction of mean and normalization
  [row,col] = size(TA2);
  for i = 1:row
   NetOut(i,:) = TA2(i,:)*NormOutVec(i);
   NetOut(i,:) = NetOut(i,:)+MeanOutVec(i);
   DesiredOut(i,:) = t(i,:)*NormOutVec(i);
   DesiredOut(i,:) = DesiredOut(i,:)+MeanOutVec(i);
  end

  TE = DesiredOut-NetOut;
 TSSE = sumsqr(TE);

 % TEST PRESENTATION PHASE
 TA1T = feval(f1,TW1*TestPatterns,TB1);
 TA2T = feval(f2,TW2*TA1T,TB2);

 % compensate for subtraction of mean and normalization
 [row,col] = size(TA2T);
 for i = 1:row
  TNetOut(i,:) = TA2T(i,:)*NormOutVec(i);
  TNetOut(i,:) = TNetOut(i,:)+MeanOutVec(i);
  TDesiredOut(i,:) = T2(i,:)*NormOutVec(i);
  TDesiredOut(i,:) = TDesiredOut(i,:)+MeanOutVec(i);
 end

 TET = TDesiredOut-TNetOut;
 TSSET = sumsqr(TET);

 % MOMENTUM & ADAPTIVE LEARNING RATE PHASE
 if TSSE > SSE*er
  lr = lr * dm;
  MC = 0;
 else
  if TSSE < SSE
   lr = lr * im;
  end
  W1=TW1; B1=TB1; W2=TW2; B2=TB2;
  A1=TA1; A2=TA2; E=TE; SSE=TSSE; SSET = TSSET;

  % BACKPROPAGATION PHASE
  D2 = feval(DF2,A2,t-A2);
  D1 = feval(DF1,A1,D2,W2);
  end

 % TRAINING RECORD
 TR(1,epoch) = SSE;
 [row,col] = size(t);
 RMSTrain(1,epoch) = ((SSE/col)/row)^0.5;
 TRT(1,epoch) = SSET;
 [row,col] = size(T2);
 RMSTest(1,epoch) = ((SSET/col)/row)^0.5;
 TR(2,epoch) = lr;
 RMSTrain(2,epoch) = TR(2,epoch);

 % DISPLAY RESULTS
 if rem(epoch,df) == 0
  plottrs([[RMSSSE0;lr0] RMSTrain(:,1:epoch)],[[RMSSSE0T] RMSTest(:,1:epoch)])
  end
 end

if rem(epoch,df)
  plottrs([[RMSSSE0;lr0] RMSTrain(:,1:epoch)],[[RMSSSE0T] RMSTest(:,1:epoch)])
 end
```

```
% RETURN RESULTS
nw1 = W1;
nb1 = B1;
nw2 = W2;
nb2 = B2;
te = epoch;
tr = [[SSE0; lr0] TR(:,1:epoch)];
trt = TRT(:,1:epoch);
```

PLOTTRS.M

```
function plottrs(tr,trt,t)
%PLOTTR  Plots record of network error and (optionally)
%       adaptive learning rate during training.
%        (See BARERR, ERRSURF)
%
%        PLOTTR(TR)
%         TR - row of training network errors with OPTIONAL
%              second row of learning rates.
%         TRT - row of test network errors
%         T  - (Optional) String for graph title.
%              Default is 'Network Training Record'.

%        M.H. Beale & H.B. Demuth, 1-31-92
%       Copyright (c) 1992-93 by the MathWorks, Inc.

if nargin > 3 | nargin < 1
  error('Wrong number of arguments.');
  end

clf reset
[r,c] = size(tr);

if r >= 2
 subplot(211)
 end

semilogy(0:c-1,tr(1,:),'r',0:c-1,trt(1,:),'g--');
xlabel('Epoch')
ylabel('RMS Error')
if nargin == 2
 title('Network RMS Error - Training and Test')
else
 title(t)
 end

if r >= 2
 subplot(212)
 plot(0:c-1,tr(2,:));
 xlabel('Epoch')
 ylabel('Learning Rate')
 if nargin == 2
   title('Network Learning Rate')
  else
   title(t)
    end
  end
drawnow
set(gcf,'NextPlot','replace')
```

CMPTRAI2.M

```
% This code calculates the final rms error for the training set
% PRESENTATION PHASE of Normalized vectors
clg;
A1 = feval(f1,W1*P,B1);
A2 = feval(f2,W2*A1,B2);

% compensate for subtraction of mean and normalization
[row,col] = size(A2);
for i = 1:row
 NetOut(i,:) = A2(i,:)*NormOutVec(i);
 NetOut(i,:) = NetOut(i,:)+MeanOutVec(i);
 DesiredOut(i,:) = T(i,:)*NormOutVec(i);
 DesiredOut(i,:) = DesiredOut(i,:)+MeanOutVec(i);
end

E = DesiredOut-NetOut;

SSE = sumsqr(E);
subplot(211), plot(DesiredOut);
title('Train Targets');
v = axis;
subplot(212), plot(NetOut), axis(v);
title('Train Outputs');
```

CMPTEST2.M

```
% This code computes the final rms error for the test set
% PRESENTATION PHASE of Normalized vectors
%P = getnorm(TestPatterns')';
clg;
A1 = feval(f1,W1*TestPatterns,B1);
A2 = feval(f2,W2*A1,B2);

% compensate for subtraction of mean and normalization
[row,col] = size(A2);
for i = 1:row
 TNetOut(i,:) = A2(i,:)*NormOutVec(i);
 TNetOut(i,:) = TNetOut(i,:)+MeanOutVec(i);
 TDesiredOut(i,:) = T2(i,:)*NormOutVec(i);
 TDesiredOut(i,:) = TDesiredOut(i,:)+MeanOutVec(i);
end

E=TDesiredOut-TNetOut;
```

```
SSE = sumsqr(E);
subplot(211), plot(TDesiredOut);
title('Test Targets');
v = axis;
subplot(212), plot(TNetOut), axis(v);
title('Test Outputs');
```

References

1 Wilheit, T.T., Some Comments on Passive Microwave Measurements of Rain, *Bull. Am. Meteorol. Soc.*, Vol. 67, pp. 1226-1232, 1986.

2 Wilheit, T.T., A.T.C. Chang, M.S.V. Rao, E.B. Rodgers, and J.S. Theon, A Satellite Technique for Quantitatively Mapping Rainfall Rates Over the Oceans, *J. Appl. Meteorol.*, Vol. 16, pp. 551-560, 1977.

3 Spencer, R.W., and D.A. Santek, Measuring the Global Distribution of Intense Convection Over Land with Passive Microwave Radiometry, *J. Clim. Appl. Meteorol.*, Vol. 24, pp. 860-864, 1985.

4 Adler, R.F., and R.A. Mack, Thunderstorm Cloud Height-Rainfall Rate Relation for use with Satellite Rainfall Estimation Techniques, *J. Appl. Meteorol*, Vol. 29 No. 7, pp. 620-632, 1990.

5 Gagin, A., D. Rosenfeld, and R.E. Lopez, The Relationship Between Height and Precipitation Characteristics of Summertime Convective Cells in South Florida, *J. Atmos. Sci.*, Vol. 42, pp. 84-94, 1985.

6 Smith, W.L., and C.M.R. Platt, Comparison of Satellite Deduced Cloud Heights with Indications from Radiosonde and Ground-Based Laser Measurements, *J. Appl. Meteorol.*, Vol. 17, pp. 1796-1802, 1978.

7 Lippmann, R.P., An Introduction to Computing with Neural Nets, *IEEE ASSP Magazine*, Vol. 4, pp. 4-22, April 1987.

8 Hornik, K., M. Stinchcombe, and H. White, Multilayer Feedforward Networks are Universal Approximators, *Neural Networks*, Vol. 2, pp. 359-366, 1989.

9 Staelin, D.H., Passive Microwave Techniques for Geophysical Sensing of the Earth from Satellites, *IEEE Transactions on Antennas and Propagation* Vol. AP-29, No. 4, pp. 863-887, July 1981.

10 Gasiewski, A.J., J.W. Barrett, P.G. Bonanni and D.H. Staelin, Aircraft-based Radiometric Imaging of Tropospheric Temperature and Precipitation Using the 118.75-GHz Oxygen Resonance, *j. Appl. Meteorol*, Vol. 29, No. 7, pp. 620-632, July 1990.

11 Gasiewski, A.J., et. al., July 1990.

12 Poggio, T., and F. Girosi, Networks for Approximation and Learning, *Proc. IEEE,* Vol. 78, No. 9, pp. 1481-1497, 1990.

13 Strang, G., *Introduction to Applied Mathematics,* Chapter 1, Wellesley-Cambridge Press, Wellesley, MA, 1986.

14 Gasiewski, A.J., and D.H. Staelin, Statistical Precipitation Cell Parameter Estimation Using Passive 118-GHz O_2 Observations, *J. Geophysical Research,* Vol. 95, No. D15, pp. 18,367-18,378, 1989.

15 Van Trees, H.L., *Detection, Estimation and Modulation Theory, Part I,* pp. 174-187, John Wiley, New York, 1968.

16 Hornik, K. et.al., 1989.

17 Hartman, E.J., J.D.Keeler, and J.M. Kowalski, Layered Neural Networks with Gaussian Hidden Units as Universal Approximations, *Neural Computation,* Vol. 2, No. 2, pp. 210-215, Summer 1990.

18 Gallant, A., and H. White, There exists a Neural Network That Does not Make Avoidable Mistakes, *Proc. IJCNN 88,* Vol. I, pp. I-657-664, 1988.

19 Dawson, M.S. and A.K. Fung, Neural Networks and Their Applications to Parameter Retrieval and Classification, *IEEE Geo. and Remote Sensing Soc. Newsletter,* pp. 6-14, Sept. 1993.

20 Rumelhart, D.E., G.E. Hinton, and R.J. Williams, Learning Internal Representations by Error Propagation, *Parallel Distributed Processing: Explorations in the Microstructures of Cognition,* pp. 318-362, Vol. 1, D.E. Rumelhart and J.L. McClelland (Eds.), MIT Press, Cambridge, MA, 1986.

21 Lippmann, R.P., 1987.

22 Gasiewski, A.J., et. al., July 1990.

23 Hornik, K., et. al., 1989.

24 Hartman, E.J., et. al., 1990.

25 Gallant, A., et. al., 1988.

26 Gasiewski, A.J., et al., 1989.