1999

A Dynamic Parameter Tuning Algorithm For Rbf Neural Networks

Junxu Li

A DYNAMIC PARAMETER TUNING ALGORITHM FOR RBF NEURAL NETWORKS

By

Junxu Li

B.S. Zhejiang University, China, 1996

A THESIS

Submitted in Partial Fulfillment of the
Requirements for the Degree of
Master of Science
(in Computer Engineering)

The Graduate School
University of Maine
May, 1999

A DYNAMIC PARAMETER TUNING ALGORITHM FOR RBF NEURAL NETWORKS

By Junxu Li

Thesis Advisor: Dr. Mohamad T. Musavi

An Abstract of the Thesis Presented

in Partial Fulfillment of the Requirements for the

Degree of Master of Science

(in Computer Engineering)

May, 1999

The objective of this thesis is to present a methodology for fine-tuning the parameters of radial basis function (RBF) neural networks, thus improving their performance. Three main parameters affect the performance of an RBF network. They are the centers and widths of the RBF nodes and the weights associated with each node. A gridded center and orthogonal search algorithm have been used to initially determine the parameters of the RBF network. A parameter tuning algorithm has been developed to optimize these parameters and improve the performance of the RBF network. When necessary, the recursive least square solution may be used to include new nodes to the network architecture.

To study the behavior of the proposed network, six months of real data at fifteen-minute intervals has been collected from a North American pulp and paper company. The data has been used to evaluate the performance of the proposed network in the approximation of the relationship between the optical properties of base sheet paper and the process variables. The experiments have been very successful and Pearson correlation coefficients of up to 0.98 have been obtained for the approximation.

ACKNOWLEDGEMENTS

I am grateful to my advisor, Dr. Mohamad Musavi, for providing me with the opportunity to pursue my Master's degree at UMaine. I would like to thank him for all his time, encouragement and guidance during my two-year graduate study. I would also like to thank Dr. John Vetelino for his assistance in my application for the graduate study at the University of Maine. I wish to thank my thesis committee members, Dr. Richard Eason, Dr. Bruce Segee, and also graduate coordinator Dr. Donald Hummels, and Dr. Duane Hanselman for all of their time and assistance with this thesis and the various courses I have taken. I would also thank Dr. Ron Bryant and his family for their kindness and help. Thanks also go to Ms. Padma Natarajan for her care and encouragement. I want to thank all the other faculty members in the department and other lab members in the Intelligent Systems Laboratory who have given me help during my graduate study at UMaine. Finally, I would like to thank my family for their support and care, without which this thesis couldn't have been done. I also extend my thanks to all my friends for their encouragement and assistance.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

1.1 Background

The radial basis function (RBF network) offers a viable alternative to the multi-layer feedforward neural network in many applications of signal processing. The original RBF [1,2,3,4] method requires that there be as many RBF centers as data points, which is rarely practical in signal processing applications, as the number of data points is usually very large. While a large number of centers will be computationally difficult and make the neural network sensitive to noise, a low number of centers will make the network local approximation properties poor. Therefore, it is critical to provide a good set of nodes for the approximation of any function using RBF networks. In an industrial application, the available data is usually redundant and contains some noise, and the data distribution is not readily available. It is therefore necessary to approximate the distribution of the input data and specify some good points as the network training data. To address these issues, several suggestions have been proposed. Chen et al. [5] have proposed the use of gridded centers in which the centers are found according to a predetermined grid in the input space. Others have used a clustering algorithm [6, 7]. More recently, several authors have applied the multi-resolution analysis to a gridded approach [5] or a clustering approach [7] to provide a good set of training data to be used as the RBF centers.

The problem of identifying appropriate centers is tied to the selection of training data. There are several suggestions that are based primarily on the closest neighbor criteria. A good discussion of width selection has been presented in [7].

The last parameter is the calculation of weights. While a matrix inversion process is the straightforward solution, it suffers from the singularity issue as well as the time consuming process of inverting large matrices. Another solution is the use of a simple delta rule and gradient descent. A more elegant approach is the use of an orthogonal search (OS) [8] or fast orthogonal search (FOS) [9] technique. Note that both OS and FOS will not only provide an efficient way of finding the weights, but they also reduce the initial number of nodes to meet an error or number of node criteria.

1.2 Objective

The objective of the thesis is to create an RBF network model with fine-tuning and optimization abilities. In this research, the RBF network is initialized with a good set of initial parameters and then based on the minimization of an error function, the parameters are tuned to achieve the best network performance. The network tuning is based on updating the network parameters on an iterative basis. A gradient method has been used to derive the update equations for the RBF network. In our experiments, the orthogonal search algorithm has been used for the initial node selection. When necessary, a recursive least squares solution may be used to add or remove certain nodes while avoiding extra computation.

To study the behavior of the proposed network, six months of real data at fifteen-minute intervals has been collected from a North American pulp and paper company. The data has been used to evaluate the performance of the proposed network for the approximation of the relationship between the optical properties of the base sheet paper and the process variables.

1.3 Thesis Organization

This thesis is comprised of six chapters. Chapter 2 provides an overview of RBF neural networks, the orthogonal least square solution, the recursive least squares solution, and the gradient descent method. Chapter 3 provides the proposed methodology for tuning RBF neural networks. This includes the network construction methods, training processes, and parameter tuning algorithms, along with some discussions of popular neural network training methods. Chapter 4 details the paper making process under investigation and Chapter 5 shows the effect of the proposed tuning algorithm compared with FOS and provides the results of the proposed RBF neural network as applied to the optical property data. Chapter 6 offers concluding remarks and provides suggestions for future work.

2. RBF NEURAL NETWORKS

This chapter provides background information on the RBF formulation, the orthogonal least square (OLS) solution, the recursive least square solution, and the gradient descent method.

2.1 RBF Network Model

Radial basis function (RBF) networks can be described as

$$f(\boldsymbol{x}) = \sum_{i=1}^{M} w_i \varphi(\|\boldsymbol{x} - \boldsymbol{c_i}\|) \qquad (2\text{-}1)$$

where $\{\varphi(\|\boldsymbol{x} - \boldsymbol{c_i}\|) \mid \boldsymbol{i} = 1, 2, ...\boldsymbol{M}\}$ is a set of $\boldsymbol{M}$ arbitrary (generally nonlinear) local functions, known as radial basis functions and also called nodes. The function is normally based on a norm, or Euclidean distance, of a point $\boldsymbol{x}$ from a center $\boldsymbol{c}$. Usually the function is defined as a Gaussian function as given below.

$$\varphi(\|\mathbf{x} - \mathbf{c}_i\|) = e^{-\frac{1}{2}(\mathbf{x}-\mathbf{c}_i)\ \boldsymbol{\Sigma}_i^{-1}(\mathbf{x}-\mathbf{c}_i)^T} \qquad (2\text{-}2)$$

where $\boldsymbol{c_i}$ is the center, $\boldsymbol{x}$ is the input vector, and $_{\boldsymbol{i}}$ is the covariance matrix. For the following discussion, we assume that the covariance matrix, $_{\boldsymbol{i},}$ is diagonal.

$_{\boldsymbol{i}} = diag[\sigma^2_{i1}, \sigma^2_{i2}, \ldots, \sigma^2_{ij,} \ldots, \sigma^2_{iN}]$, where N is the number of input variables, i.e., the dimension of vector of $\boldsymbol{x}$, $\boldsymbol{\sigma_i} = [\sigma_{i1}, \sigma_{i2}, \ldots, \sigma_{iN}]$ is the width vector for the ith RBF node.

As we can see, the RBF network is a linear combination of nonlinear basis functions, which map the input to the output. Once the basis functions are identified, the radial basis function performs a fixed nonlinear transformation with no adjustable parameters and it maps the input space onto a new space. Then by implementing a linear combiner on this new space we can get the output of the network. The only adjustable parameters are the weights of the linear combiner, once we fix the set of radial basis functions. These weights can be determined using a linear least square (LS) solution, which is an important advantage of using fixed centers. Actually, the nonlinear feature within an RBF network can be chosen from a few typical nonlinear functions. A general consensus is that the choice of the nonlinear function for the hidden layer is not crucial for performance, and this opinion can also be justified using the results of a theoretical investigation [4]. However the performance of an RBF network critically depends upon the chosen nodes. In practice the centers of the nodes are often chosen to be a subset of the data. Although researchers are well aware that the fixed centers should suitably sample the input domain, most published results simply assume that the centers are arbitrarily selected from data points. Such a mechanism usually turns out unsatisfactory for building RBF networks. The resulting RBF networks often either perform poorly or have a large size. Alternatively the orthogonal least squares (OLS) method [8] can be employed as a forward regression procedure to select a suitable set of centers from a large set of candidates.

2.2 Orthogonal Least Square(OLS) Learning Algorithm

We can view the RBF network as a special case of the linear regression model

$$d(t) = \sum_{i=1}^{M} p_i(t)\theta_i + e(t) \tag{2-3}$$

where *d(t)* is the desired output, the θ_i are the parameters, and the $p_i(t)$ are known as the regressors, e.g., nodes, which are some fixed functions of ***x**(t)*:

$$\boldsymbol{p_i}(\boldsymbol{t}) = \boldsymbol{p_i}(\boldsymbol{x}(\boldsymbol{t})) \tag{2-4}$$

The error signal *e(t)* is assumed to be uncorrelated with the regressors $p_i(t)$. A constant term can be included in *Equation (2-3)* by setting assuming $p_0(t) = 1$ and changing the summation to start from 0. For the sake of simplicity, this constant term has been omitted from the following formulation. We can arrange *Equation (2-3)* in the following matrix form:

$$\boldsymbol{d} = \boldsymbol{P\theta} + \boldsymbol{e} \tag{2-5}$$

where

$\boldsymbol{d} = [d(1) \cdots d(N)]^T$

$\boldsymbol{P} = [\boldsymbol{p}_1 \cdots \boldsymbol{p}_M],$

$\boldsymbol{p}_i = [p_i(1) \cdots p_i(N)]^T, \quad 1 \leq i \leq M$

$\boldsymbol{\theta} = [\theta_i \cdots \theta_M]^T$

$\boldsymbol{e} = [e(1) \cdots e(N)]^T$

and *N* is the number of input samples, e.g. ***x**(1)*, ***x**(2)*, ... ***x**(N)*

The regressor vectors $\boldsymbol{p_i}$ form a set of basis vectors, and the LS solution $\boldsymbol{\theta}$ satisfies the condition that $\boldsymbol{P\theta}$ is the projection of ***d*** onto the space spanned by these basis vectors. Because different regressors are generally correlated, it is not clear how much each individual regressor contributes to the output. We can use the OLS method to transform the set of $\boldsymbol{p_i}$ into a set of orthogonal basis vectors. This makes it possible to calculate the individual contribution to the desired output from each basis vector. The classical *Gram-Schmidt* can be used to solve for $\boldsymbol{\theta}$. The *Gram-Schmidt* method computes orthogonal basis vectors as follows. At the *kth* stage make the *kth* column orthogonal to each of the *k-1* previously orthogonalized columns and repeat the operation for *k=2, … , M*. The computational procedure can be represented as

$$\begin{aligned} w_1 &= p_1 \\ \alpha_{ik} &= \frac{w_i^T p_k}{(w_i^T w_i)} \\ w_k &= p_k - \sum_{i=1}^{k-1} \alpha_{ik} w_i \quad k = 2, \cdots M . \\ & 1 \le i < k \end{aligned} \tag{2-6}$$

We can use the OLS method for subset selection of all the candidate regressors. Each time we select only one regressor to maximize the normalized projection over the same desired vector ***d***. Because w_i and w_j are orthogonal for $i \neq j$, the sum of squares of $d(t)$ is

$$d^T d = \sum_{i=1}^{M} g_i^{\,2} w_i^T w_i + e^T e \tag{2-7}$$

where g_i is the coefficient for *i*th regressor.

Using the classical *Gram-Schmidt* scheme, the regressor selection procedure is summarized as follows:

Step1. At the first step, for $1 \le i \le M$, compute

$$\begin{aligned} w_1^{(i)} &= p_i \\ corr_1^{(i)} &= \left| \frac{(w_1^{(i)})^T d}{\sqrt{[((w_1^{(i)})^T w_1^{(i)})(d^T d)]}} \right| \end{aligned} \tag{2-8}$$

Find i_1 such that

$corr^{(i1)} = \max(corr^{(i)})$, where $1 \le i \le M$

and select $w_1 = w_1^{(i1)} = p_{i1}$

Step 2. At the *kth* step where $k \ge 2$, for $1 \le i \le M,\ i \neq i_1,\ \cdots,\ i \neq i_{k-1}$, compute

$$\alpha_{jk}^{(i)} = \frac{w_j^T p_i}{(w_j^T w_j)}, \quad 1 \le j < k$$

$$w_k = p_i - \sum_{j=1}^{k-1} \alpha_{jk}^{(i)} w_j \tag{2-9}$$

$$corr_k^{(i)} = \left| \frac{(w_k^{(i)})^T d}{\sqrt{[((w_k^{(i)})^T w_k^{(i)})(d^T d)]}} \right|$$

Find i_k such that

$corr^{(i_k)} = \max(corr^{(i)})$, where $1 \le i \le M, i \ne i_1, \cdots, i \ne i_{k-1}$

and select

$$w_k = w_k^{(i_k)} = p_i - \sum_{j=1}^{k-1} \alpha_{jk}^{(i)} w_j \tag{2-10}$$

The coefficient for each regressor is

$$g_k = \frac{w_k^T d}{(w_k^T w_k)} \tag{2-11}$$

In fact, those coefficients g_k are the optimal weights associated with each regressor.

Step 3. The selection procedure can be terminated when the error has been reduced to an acceptable level.

It's not difficult to see the maximum of step is M, which is the number of candidate regressors. Usually M is quite large. The use of OLS can reduce the number of regressors significantly according to the behavior the network while avoiding an ill-conditioned matrix computation. A similar Fast Orthogonal Search (FOS) algorithm, which has a fast computational nature, has been developed at University of Maine [9].

We can see from the above analysis that OLS uses a set of vectors to approximate the features of each regressor, which is nonlinear. And then, uses a linear analysis method to evaluate the correlation between these vectors and their respective contributions to the output. One important point worth mentioning is that each nonlinear regressor should be sufficiently represented by the vectors, otherwise the linear relation derived from a set of vectors won't be valid over the whole input space. This can be partly solved by selecting enough data to sample the functional input space sufficiently.

2.3 Recursive Least Squares Solution

A real system can evolve over time. Fixing a system model based on limited data samples is not always reliable. Incorporating certain adaptive mechanisms can be very helpful to accurately track the changes in the system behavior while avoiding the need for complete retraining of the network.

Assume a system can be described as:

$$\boldsymbol{y}(t) = \boldsymbol{X}(t)\,\boldsymbol{\theta}(t) + \boldsymbol{e}(t) \tag{2-12}$$

where $\boldsymbol{X}(t)=[\boldsymbol{x}(1), \boldsymbol{x}(2), \ldots, \boldsymbol{x}(t)]^T$

$\boldsymbol{x}^T(t)= [x_1(t), x_2(t), \ldots, x_n(t)]$

$\boldsymbol{y}(t)=[y(1), y(2), \ldots, y(t)]^T$

$\boldsymbol{e}(t)=[e(1), e(2), \ldots, e(t)]^T$

$\boldsymbol{\theta}(t)=[\boldsymbol{\theta}_1(t), \boldsymbol{\theta}_2(t), \ldots, \boldsymbol{\theta}_n(t)]^T$

The least squares solution for the parameter vector is,

$$\boldsymbol{\theta}(t) = [\boldsymbol{X}^T(t)\,\boldsymbol{X}(t)]^{-1}\,\boldsymbol{X}^T(t)\boldsymbol{y}(t) \tag{2-13}$$

Using recursive least squares, when a new sample comes in, $\boldsymbol{\theta}$ will be updated based on the new sample only, instead of storing all previous data and repeating the pseudo-inverse *Equation (2-13)*. This allows significant saving in computation [10].

At time step t,

$\boldsymbol{X}(t)= [\boldsymbol{x}^T(1), \boldsymbol{x}^T(2), \ldots, \boldsymbol{x}^T(t)]^T$

$\boldsymbol{y}(t)= [y(1), y(2), \ldots, y(t)]^T$

$\boldsymbol{\theta}(t) = [\boldsymbol{X}^T(t)\,\boldsymbol{X}(t)]^{-1}\,\boldsymbol{X}^T(t)\boldsymbol{y}(t)$

At time step t+1, new data becomes available. So the input and output data matrices are updated as in (2-14) and (2-15).

$$\boldsymbol{X}(t+1) = \begin{bmatrix} \boldsymbol{X}(t) \\ x^T(t+1) \end{bmatrix} \tag{2-14}$$

$$\boldsymbol{y}(t+1) = \begin{bmatrix} \boldsymbol{y}(t) \\ y(t+1) \end{bmatrix} \tag{2-15}$$

Using (2-13) and the update vectors, the estimate for $\boldsymbol{\theta}$ at step *t+1* is then given by

$$\boldsymbol{\theta}(t+1) = [\boldsymbol{X}^T(t+1)\,\boldsymbol{X}(t+1)]^{-1}\,\boldsymbol{X}^T(t+1)\boldsymbol{y}(t+1) \tag{2-16}$$

Now

$$\boldsymbol{X}^T(t+1)\,\boldsymbol{X}(t+1) = \boldsymbol{X}^T(t)\,\boldsymbol{X}(t) + \boldsymbol{x}(t+1)\boldsymbol{x}^T(t+1) \tag{2-17}$$

Thus given $\boldsymbol{x}(t+1)$ we can easily update the old correlation matrix, $\boldsymbol{X}^T(t)\,\boldsymbol{X}(t)$, to obtain new matrix, $\boldsymbol{X}^T(t+1)\,\boldsymbol{X}(t+1)$. Next we try to update the inverse of $\boldsymbol{X}^T(t)\,\boldsymbol{X}(t)$ directly without requiring a matrix inversion at each time step. In addition, we also need to update the term $\boldsymbol{X}^T(t+1)\boldsymbol{y}(t+1)$.

$$\boldsymbol{X}^T(t+1)\boldsymbol{y}(t+1) = \boldsymbol{X}^T(t)\boldsymbol{y}(t) + \boldsymbol{x}(t+1)y(t+1) \tag{2-18}$$

Introducing the following notations

$$\boldsymbol{P}(t) = [\boldsymbol{X}^T(t)\,\boldsymbol{X}(t)]^{-1}$$

$$\boldsymbol{B}(t) = \boldsymbol{X}^T(t)\boldsymbol{y}(t)$$

we get

$$\boldsymbol{\theta}(t+1) = \boldsymbol{P}(t+1)\,\boldsymbol{B}(t+1) \tag{2-19}$$

$$\boldsymbol{\theta}(t) = \boldsymbol{P}(t)\,\boldsymbol{B}(t) \tag{2-20}$$

also

$$\boldsymbol{P}^{-1}(t+1) = \boldsymbol{P}^{-1}(t) + \boldsymbol{x}(t+1)\boldsymbol{x}^T(t+1)$$

$$\boldsymbol{B}(t+1) = \boldsymbol{B}(t) + \boldsymbol{x}(t+1)y(t+1) \tag{2-21}$$

by applying the Matrix Inversion Lemma:

$$(A + BCD)^{-1} = A^{-1} - A^{-1}B(C^{-1} + DA^{-1}B)^{-1}DA^{-1}$$

and assigning

$$\boldsymbol{A} = \boldsymbol{P}^{-1}(t),\ \boldsymbol{C} = 1,\ \boldsymbol{B} = \boldsymbol{x}(t+1),\ \boldsymbol{D} = \boldsymbol{x}^T(t+1)$$

we get

$$\boldsymbol{P}(t+1) = \boldsymbol{P}(t)[\boldsymbol{I}_m - \boldsymbol{x}(t+1)\ (\ 1+ \boldsymbol{x}^T(t+1)\boldsymbol{P}(t)\boldsymbol{x}(t+1)\ )^{-1}\ \boldsymbol{x}^T(t+1)\boldsymbol{P}(t)] \tag{2-22}$$

Equation (2-22) gives us the means to update $\boldsymbol{P}(t)$ to $\boldsymbol{P}(t+1)$ without inverting a matrix. The only inversion is the scalar term $(\ 1+ \boldsymbol{x}^T(t+1)\boldsymbol{P}(t)\boldsymbol{x}(t+1)\ )$. Using *Equation (2-21)* and *Equation (2-22)*, we can get $\boldsymbol{\theta}(t+1)$. In summary, the full recursive least squares (RLS) algorithm for updating $\boldsymbol{\theta}(t)$ is as follows:

At time step *t+1*:

1. Form $\boldsymbol{x}(t+1)$ using the new data
2. Update $\boldsymbol{B}(t)$ using

$$\boldsymbol{B}(t+1) = \boldsymbol{B}(t) + \boldsymbol{x}(t+1)y(t+1)$$

3. Form $\boldsymbol{P}(t+1)$ using

$$\boldsymbol{P}(t+1) = \boldsymbol{P}(t)[\boldsymbol{I}_m - \boldsymbol{x}(t+1)\ (\ 1+ \boldsymbol{x}^T(t+1)\boldsymbol{P}(t)\boldsymbol{x}(t+1)\ )^{-1}\ \boldsymbol{x}^T(t+1)\boldsymbol{P}(t)]$$

4. Update $\boldsymbol{\theta}(t)$

$$\boldsymbol{\theta}(t+1) = \boldsymbol{P}(t+1)\ \boldsymbol{B}(t+1)$$

5. Go back to (1) when new data come in.

Of course, we also need to initialize parameters $\boldsymbol{\theta}$, and $\boldsymbol{P}(0)$, with prior knowledge or initial estimation by using a few samples at the beginning. There could be some trial and error processes, especially for $\boldsymbol{P}(0)$, and its updates. To ensure that $\boldsymbol{P}$ is not ill-conditioned during the recursive process, it is important to start with good initial conditions.

In summary, RLS is a linear regression method that has advantages over traditional linear regression methods. It can be implemented on-line for its adaptive updating of the whole set of parameters, provided the linear model is suitable for the intended system.

2.4 The Gradient Method

Gradient method has been shown to suitably parameterize nonlinear dynamical systems [11]. The method commonly needs to evaluate the gradient of a performance function V with respect to variable $\boldsymbol{\theta}$.

2.4.1 The Gradient of a Function

Let $V(\boldsymbol{\theta})=V(\theta_1, \cdots, \theta_n)$ be a scalar function of n variables $\theta_1, \cdots, \theta_n$, where $\theta_i\, (i=1, ..., n)$ are the elements of the vector $\boldsymbol{\theta}$. The gradient of $V(\boldsymbol{\theta})$ with respect to vector $\boldsymbol{\theta}$ is defined as the row vector,

$$V_\theta = \left[\frac{\partial V}{\partial \theta_1}, \cdots, \frac{\partial V}{\partial \theta_n}\right] \qquad (2\text{-}23)$$

The value of the gradient depends upon the point $\boldsymbol{\theta}_{nom} \in \boldsymbol{R}^n$ (denoting the nominal value of $\boldsymbol{\theta}$) at which f_θ is evaluated. If the operating point is changed from $\boldsymbol{\theta}_{\boldsymbol{nom}}$ to $\boldsymbol{\theta}_{\boldsymbol{nom}} + \boldsymbol{\Delta\theta}$, where $\boldsymbol{\Delta\theta} = -\eta\, \boldsymbol{V}_{\boldsymbol{\theta}}^{\boldsymbol{T}}$, $\eta<<1$ is a positive constant, it follows that $V(\boldsymbol{\theta}_{\boldsymbol{nom}} + \boldsymbol{\Delta\theta}) \le V(\boldsymbol{\theta}_{\boldsymbol{nom}})$, since $V(\boldsymbol{\theta}_{\boldsymbol{nom}} + \boldsymbol{\Delta\theta}) \approx V(\boldsymbol{\theta}_{\boldsymbol{nom}}) - \eta\, \boldsymbol{V}_{\boldsymbol{\theta}} \boldsymbol{V}_{\boldsymbol{\theta}}^{T}$. It is this concept that is used in all gradient methods for the optimization of static and dynamical systems.

2.4.2 Optimization Using the Gradient Method

Let $V(\boldsymbol{\theta})$, a performance index which has to be optimized with respect to a parameter vector $\boldsymbol{\theta}$, be defined as

$$V(\theta) = \frac{1}{T} {}_{0}^{T} L[e(\theta,\tau)]d\tau \qquad (2\text{-}24)$$

in the continuous case and as

$$V(\theta) = \frac{1}{N} {}_{i=1}^{N} L[e(\theta,i)] \qquad (2\text{-}25)$$

in the discrete case. The error *e* in *Equation (2-24)* and *Equation (2-25)* is assumed to be a function of time as well as the parameter $\boldsymbol{\theta}$. *L* is some function to calculate the overall error. If the gradient method is used to minimize the performance index, $\partial V / \partial \theta$ is determined, by the chain rule, as

$$\frac{\partial V}{\partial \theta} = \frac{1}{T} {}_{0}^{T} \frac{\partial L[\, e(\theta,\tau\,)]}{\partial e(\,\theta,\tau\,)} \cdot \frac{\partial e(\,\theta,\tau\,)}{\partial \theta} d\tau \qquad \text{(2-26)}$$

or in the discrete case

$$\frac{1}{N} {}_{i=1}^{N} \left[\frac{\partial L[\, e(\,\theta,i\,)]}{\partial e(\,\theta,i\,)} \cdot \frac{\partial e(\,\theta,i\,)}{\partial \theta} \right] \qquad \text{(2-27)}$$

In both cases, $\boldsymbol{\theta}$ is adjusted as $\boldsymbol{\theta} = \boldsymbol{\theta}_{\boldsymbol{nom}} - \eta \cdot \partial V / \partial \theta$ and the process is repeated.

In practice, however, the parameter $\boldsymbol{\theta}$ in a dynamical system is adjusted on-line and the system cannot be reinitialized. The performance index is then defined over a finite interval *[t-T, t]* for continuous-time systems or a finite interval *[k-T+1, k]* for discrete-time systems. Further, a commonly used function $L[e(\boldsymbol{\theta},\tau)]$ has the form $\| \, e(\boldsymbol{\theta},\tau) \|^2$, where $e(\boldsymbol{\theta},\tau)$ is an output error vector. For the discrete case, $J(\boldsymbol{\theta})$ has the form

$$V(\,\theta\,) = \frac{1}{T} {}_{i=k-T+1}^{k} e^T e \qquad \text{(2-28)}$$

and the gradient is computed over the interval *[k-T+1, k]* as

$$\frac{\partial V}{\partial \theta} = \frac{2}{T} {}_{i=k-T+1}^{k} \left[\frac{\partial e}{\partial \theta} \right]^T e \qquad \text{(2-29)}$$

Then the parameter $\boldsymbol{\theta}$ is adjusted as

$$\theta(k+1)=\theta(k-T+1)-\eta\cdot\frac{2}{T}\cdot\mathop{}_{i=k-T+1}^{k}\left[\frac{\partial e}{\partial\theta}\right]^{T}e \qquad (2\text{-}30)$$

where $\eta<<1$, i.e. the step size, is a suitably chosen constant. As we see from the above, $\boldsymbol{\theta}$ is no longer a constant parameter but a function of time, so that the concept of a partial derivative described earlier has to be replaced by the concept of a functional derivative. However, if η is sufficiently small, we shall assume that the concept of a partial derivative (or gradient in parameter space) can still be applied.

There is a more general form for the gradient method, which is called Descent Algorithms [12].

2.4.3 Descent Algorithms

Definition: Descent Algorithm. Assume $\boldsymbol{V}$: $\boldsymbol{R^d} \rightarrow \boldsymbol{R}$ has continuous second partial derivatives on $\boldsymbol{R^d}$. Let $\boldsymbol{g}$: $\boldsymbol{R^d} \rightarrow \boldsymbol{R^d}$ be the gradient of $\boldsymbol{V}$. Assume there exists a sequence of d-dimensional real vectors $\boldsymbol{\theta}(0)$, $\boldsymbol{\theta}(1)$, … such that for $t=0, 1, 2, \ldots$

$$\boldsymbol{\theta}(t+1)=\boldsymbol{\theta}(t)+\eta_t\,\boldsymbol{f}(t) \qquad (2\text{-}31)$$

where step size $\eta_t \in (0, \infty)$ and the descent direction $\boldsymbol{f}(t) \in \boldsymbol{R^d}$ satisfies either

(1) $\boldsymbol{g}(\boldsymbol{\theta}(t))^T\boldsymbol{f}(t)<0$

or

(2) $|\,\boldsymbol{g}(\boldsymbol{\theta}(t))| = |\,\boldsymbol{f}(t)\,| = 0.$

The difference equation *(2-31)* is a descent algorithm defined with respect to the *objective function V* generating the sequence $\boldsymbol{\theta}(0), \boldsymbol{\theta}(1), \ldots$.

Proposition: Descent Algorithm Proposition. Assume $\boldsymbol{V}$: $\boldsymbol{R^d} \rightarrow \boldsymbol{R}$ has: (i) continuous second partial derivatives on $\boldsymbol{R^d}$. Let $\boldsymbol{g}$: $\boldsymbol{R^d} \rightarrow \boldsymbol{R^d}$ be the gradient of $\boldsymbol{V}$. Assume there exists a sequence of d-dimensional real vectors $\boldsymbol{\theta}(0)$, $\boldsymbol{\theta}(1)$, … such that for $t=0, 1, 2, \ldots$

$$\boldsymbol{\theta}(t+1)=\boldsymbol{\theta}(t)+\eta_t\,\boldsymbol{f}(t)$$

where $\eta_t \in (0, \infty)$ and $\boldsymbol{f}(t) \in \boldsymbol{R^d}$ are defined such that either

(1) $\boldsymbol{g}(\boldsymbol{\theta}(t))^T\boldsymbol{f}(t)<0$

or

$$(2)\quad |\,\boldsymbol{g}(\boldsymbol{\theta}(t))| = |\,\boldsymbol{f}(t)\,| = 0.$$

Then for each $t=0, 1, 2, \ldots$, there exists a η_t such that if $\boldsymbol{\theta}(t+1) \neq \boldsymbol{\theta}(t)$:

$$V(\boldsymbol{\theta}(t+1)) < V(\boldsymbol{\theta}(t)).$$

Proof: Expand V in a Taylor expansion about the point $\boldsymbol{\theta}$(t) and evaluate at $\boldsymbol{\theta}$(t+1) to obtain for some sufficiently small strictly positive real number η_t:

$$V(\boldsymbol{\theta}(t+1)) = V(\boldsymbol{\theta}(t)) + \boldsymbol{g}(\boldsymbol{\theta}(t))^T(\boldsymbol{\theta}(t+1) - \boldsymbol{\theta}(t)) + \boldsymbol{O}(\eta_t^{\,2}),$$

where $\boldsymbol{g}(\boldsymbol{\theta}(t))$ is the gradient of V evaluated at $\boldsymbol{\theta}$(t). For the case $\boldsymbol{\theta}(t+1) \neq \boldsymbol{\theta}(t)$, it is possible to choose a η_t such that

$$V(\boldsymbol{\theta}(t+1)) < V(\boldsymbol{\theta}(t))$$

since

$$\boldsymbol{g}(\boldsymbol{\theta}(t))^T(\boldsymbol{\theta}(t+1) - \boldsymbol{\theta}(t)) = \eta_t\,\boldsymbol{g}(\boldsymbol{\theta}(t))^T\boldsymbol{f}(t) < 0.$$

2.4.4 Step Size Selection Strategies

Equation (2-31) may be viewed as a heuristic search algorithm designed to update $\boldsymbol{\theta}$(t) such that the revised estimate $\boldsymbol{\theta}$(t+1) is closer to a global minimum, where the distance to the global minimum at point $\boldsymbol{\theta}$(t) is given by $V(\boldsymbol{\theta}(t))$. There are some problems, however, with this heuristic search algorithm. These problems arise because the restrictions on the step size sequence $\eta_1, \eta_2, \ldots$ are relatively weak and must be at least slightly strengthened. If η_t is chosen to be too large for some iteration *t*, then $V(\boldsymbol{\theta})$ may increase in value. On the other hand, if $\eta_1, \eta_2, \ldots$ are too small, then the algorithm may take a long time to converge, and it may not even converge to a critical point of the function $V(\boldsymbol{\theta})$. Let's take a look at the definition for properly chosen step size first.

Definition: Properly chosen step size. Assume $\boldsymbol{V}: \boldsymbol{R}^d \to \boldsymbol{R}$ has continuous second partial derivatives on $\boldsymbol{R}^d$. Let $\boldsymbol{g}: \boldsymbol{R}^d \to \boldsymbol{R}^d$ be the gradient of $\boldsymbol{V}$. Assume there exists a sequence of *d*-dimensional real vector $\boldsymbol{\theta}(0), \boldsymbol{\theta}(1), \ldots$ with the property that for $t=0, 1, 2, \ldots$

$$\boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t) + \eta_t \boldsymbol{f}(t)$$

where $\eta_t \in (0, \infty)$. and $\boldsymbol{f}(t) \in \boldsymbol{R}^d$ is defined such that $\boldsymbol{g}(\boldsymbol{\theta}(t))^T \boldsymbol{f}(t) \le 0$. The step size η_t is properly chosen at iteration *t* for $t \in \{0, 1, \ldots\}$ if given $\boldsymbol{\theta}(t)$ and $\boldsymbol{f}(t)$:

$$V(\theta(t) + \eta_t \boldsymbol{f}(t)) \le V(\theta(t)) + \alpha\eta_t \boldsymbol{g}(\theta(t))^T \boldsymbol{f}(t) \qquad (2\text{-}32)$$

and

$$\boldsymbol{g}(\boldsymbol{\theta}(t) + \eta_t \boldsymbol{f}(t))^T \boldsymbol{f}(t) \ge \beta\, \boldsymbol{g}(\boldsymbol{\theta}(t))^T \boldsymbol{f}(t) \qquad (2\text{-}33)$$

are satisfied, where α, β are real numbers that satisfy

$$0<\alpha<\beta<1$$

A properly chosen step size η_t satisfies both of the constraints in *Equation (2-32)* and *Equation (2-33)*. It has been shown that these constraints are sufficient to permit a fairly detailed convergence analysis of a very large class of gradient algorithms. An important advantage of the concept of a properly chosen step size is that it gives us considerable flexibility in the choice of the step size at each step of the algorithm [12].

A *Properly Chosen Step Size Existence Theorem* is given in [9].

Assume $\boldsymbol{V}: \boldsymbol{R}^d \rightarrow \boldsymbol{R}$ has: (i) continuous second partial derivatives on $\boldsymbol{R}^d$, and (ii) a lower bound on $\boldsymbol{R}^d$. Let $\boldsymbol{g}: \boldsymbol{R}^d \rightarrow \boldsymbol{R}^d$ be the gradient of $\boldsymbol{V}$, and let $t \in \{0, 1, \ldots\}$. Let $\boldsymbol{\theta}(\mathrm{t}) \in \boldsymbol{R}^d$ and $\boldsymbol{\theta}(\mathrm{t}+1) \in \boldsymbol{R}^d$. Let $\boldsymbol{\theta}(\mathrm{t}+1) = \boldsymbol{\theta}(\mathrm{t}) + \eta_t \boldsymbol{f}(t)$ where $\eta_t \in (0, \infty)$ and $\boldsymbol{f}(t) \in \boldsymbol{R}^d$ is defined such that $\boldsymbol{g}(\boldsymbol{\theta}(t))^T \boldsymbol{f}(t) \leq 0$. Let α, β be positive numbers defined as in *Equation (2-32)* and *Equation (2-33)* so that $0<\alpha<\beta<1$. Then there exists strictly positive real numbers η_{min} and η_{max} such that every $\eta_t \in (\eta_{min}, \eta_{max})$ satisfies *Equation (2-32)* and *Equation(2-33).*

It has been shown that the gradient method converges based on the same assumption for *Properly Chosen Step Size Existence Theorem* [9].

2.4.5 Summary

From the above analysis, the convergence and stability of the gradient method are highly dependent on how the step size or learning rate is chosen and how it is adjusted during the global optimization process. Especially for the global minimum problem of an unknown function, it is a process of trial and error to identify the fitness of a critical point as close as possible to the global minimum. It is very important to avoid being trapped in a local minimum that is far away from the global minimum. Therefore construction of an appropriate optimization process using proper step size can improve the solution.

3. NETWORK CONSTRUCTION METHOD AND TUNING ALGORITHM

The important issues in the implementation of RBF neural networks are the selection of important variables to be used in the network, the selection of a good set of training data, and the selection of initial network parameters such as the centers, widths and weights. In addition to presenting the RBF network formulation, the following sections provide our methodologies for dealing with these issues.

3.1 Network Model

The RBF network equation can be formulated as:

$$f(\boldsymbol{x}) = \sum_i w_i f_i(x) + \sum_j w_j x_j + c \qquad (3\text{-}1)$$

Equation (3-1) contains both linear and nonlinear terms. For the following derivation, it can be written in a more expressive way.

$$f(\boldsymbol{x}) = \sum_i w_i e^{-\frac{1}{2}{}_{i}(x_j - c_{ij})^2/\sigma_{ij}^2} + \sum_j w_j x_j + c \qquad (3\text{-}2)$$

where *i* is the index of the centers, and *j* is the index of the input variables.

3.2 Sensitivity Analysis

For a real system, there are usually a number of measured variables available. The importance of these variables to the final output is often not known in advance. Analyzing the significance of each variable in a statistical sense can help shed some light on important variables. Neural networks often have difficulty mapping a high number of inputs to an output [16]. The problem is sometimes referred to the "curse of dimensionality" in neural networks. In addition to potentially poor mapping, the network computational time for large input spaces can become enormous. A method is therefore called for to determine which subset of inputs, out of many, are most significant for functional mapping. Such a process is commonly referred to as a sensitivity analysis.

A sensitivity analysis is a means of ranking all input variables relative to their importance in determining the functional mapping. The result of such an analysis is a knowledgeable premise of how to appropriately reduce input space to only the most relevant variables. There are known linear statistical methods to evaluate the relative correlation of one variable with respect to another. While the actual mapping between each input variable and the output may be nonlinear, the use of linear statistical techniques can still be useful in estimating the significance of each variable. Some researchers have in fact a neural network, trained on all variables, to find a nonlinear mapping and then use it for identifying the most important variables. However, this approach might be misleading because the mapping function might not represent the actual nonlinear relationship between the input and output.

In this thesis, the Pearson correlation coefficient [13] has been applied to initially find the significance of each variable to the output. Each coefficient, usually ranging from +1 to –1, expresses the extent of correlation between the two variables. The Pearson correlation coefficient between two quantitative variables, ***X*** and ***Y***, assuming that their averages are $\bar{\boldsymbol{X}}$ and $\bar{\boldsymbol{Y}}$, is defined as:

$$r_p = \frac{(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{(X_i - \bar{X})^2 \quad (Y_i - \bar{Y})^2}} \tag{3-3}$$

The coefficient, r_p, shows the strength of linear association between input vector ***X*** and output vector ***Y***. The interpretation of the correlation coefficient must be done appropriately. A variable with a higher correlation coefficient does not necessarily mean that it has higher significance than other variables. Looking at the nature of each variable and eliminating as much noise as possible is an essential step in getting a good set of correlation coefficients. Because of the highly nonlinear nature of a complex system, the correlation analysis only gives us partial information. Any other helpful information, such as the input from plant engineers, should be combined to analyze the significance of each variable. Using the RBF network model may partly verify the results of sensitivity analysis through a trial and error procedure.

3.3 Initial Center Selection And Training Data Selection

There are basically two phases of artificial neural network construction. One is the training process and the other is the testing process. The training process parameterizes the network in an optimal fashion, the testing process uses samples not used during the training process to compare the network output with the desired output and then evaluate the applicability of the network.

To successfully train any neural network means that the main characteristics of the underlying system are captured despite the limited training samples for the system, and that the network provides good generalization over the entire system input domain. There are two questions that need to be answered. One is the selection of a subset of the entire data for the training process. The idea is to select as small a subset as possible such that the subset spreads “uniformly” over the input space and the network can yield a satisfactory performance over all other data samples. The other question is to assign centers for the network nodes. This question is closely related to the selection of training data. Researchers have found out that it’s not feasible to have as many centers as data points available because of the computation difficulty and over-redundant nature. In order to reduce the computation while maintaining the representative features of the data, random training data selection, clustering algorithms and gridded center methods are commonly used. However, the random selection procedures don’t guarantee that the selected data is a good representative subset of the whole data pool. Also, because of its random nature, the process is usually not repeatable. Clustering algorithms use distance measures to adjust the center distribution such that the centers will represent the input space to a certain degree. However cluster analysis is highly empirical. Different methods can lead to different groupings, both in number and in content. Furthermore, since the groups are not known *a priori*, it is usually difficult to judge whether the results make sense in the context of the problem being studied [14]. Gridded center methods look at the range of all inputs then divides the range into a grid with a reasonable number of cells [15]. A node is then assigned to each cell.

However, not all grid cells contain data samples, which makes the center assignment over-redundant. More importantly, the characteristics of the input domain are not represented well.

Multi-resolution analysis (MRA) can be combined with a clustering approach or gridded approach to create a multi-resolution clustering or multi-resolution gridded approach [15]. The multi-resolution approaches assign centers at more than one level to account for low frequency (smooth) and high frequency (sharp) components of the mapping function. They have been shown to be quite attractive in terms of reducing error in the training process by adding more and more centers. However the computation cost may become a problem as the number of centers increases. It should be noted that better performance on the training data doesn't necessarily mean improved generalization of the network. In fact, network over-fitting may make the generalization worse. So the solution to over-fitting problem is to use a limited number of centers to achieve good generalization.

Given a set of data describing a system, there is usually some distribution characteristics for most data samples. Investigating the distributions of the data samples at different ranges and selecting a good subset of the data according to the homogeneity of the data has been found to be worthwhile. Here homogeneity means defining certain criteria to evaluate the association between data samples. More promising centers can then be created to map the nonlinearity of the system under investigation.

Our approach here is similar to the gridded approach in that it divides the input space into a certain number of regions. We then consider how many samples fall in each region and pick those regions with a sufficient number of samples present. To each selected region, we assign one node at the center of the region. There is no reason to assign a center for a region to which virtually no samples belong. Here we implicitly use the region area to stand for the homogeneity of the data samples. Figure 1 gives an example of our initial center selection technique. There are two points worth mentioning. First, that the entire data set is used in this process. Second, if the data is normalized, the spacing along each dimension is the same, i.e. the cells are square; otherwise, the cells are rectangular.

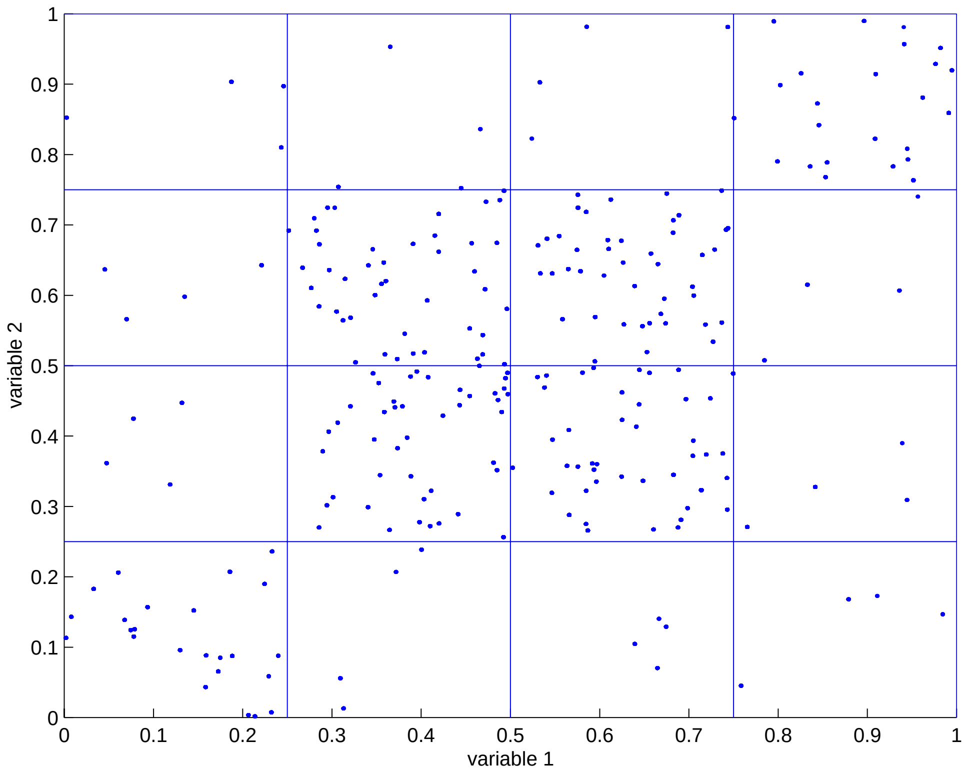


Figure 1: An example for initial center selection

After selecting the centers, the width for each center should be initialized. This is done according to a predefined overlapping between neighbor centers in the grid. Here, the predefined overlapping is the response of the Gaussian function at its neighbor's center, denoted by *neighborhood*. Also from the center selection procedure, we know the distance between two neighboring Gaussian functions denoted by *distance*. Here we only need to consider the one dimensional Gaussian function

$$y = e^{-\frac{(x-c)^2}{2\sigma^2}} \tag{3-4}$$

where c is the center, σ is the width, and y is the response of the Gaussian function. Based on the known parameters and given the overlapping value, we can derive the initial widths for the Gaussian function

$$\sigma_{initial} = distance \cdot \sqrt{\frac{1}{-2\log_e(neighborhood)}} \tag{3-5}$$

Figure 2 shows four Gaussian functions on one axis with *neighborhood = 0.3*. An initial set of centers with initial widths is selected according to the above procedure. Then for each selected region, a certain percentage of the data is selected such that the distribution is fairly uniformly over the region. The selected data is used for training the network, and the rest in the region is used for testing.

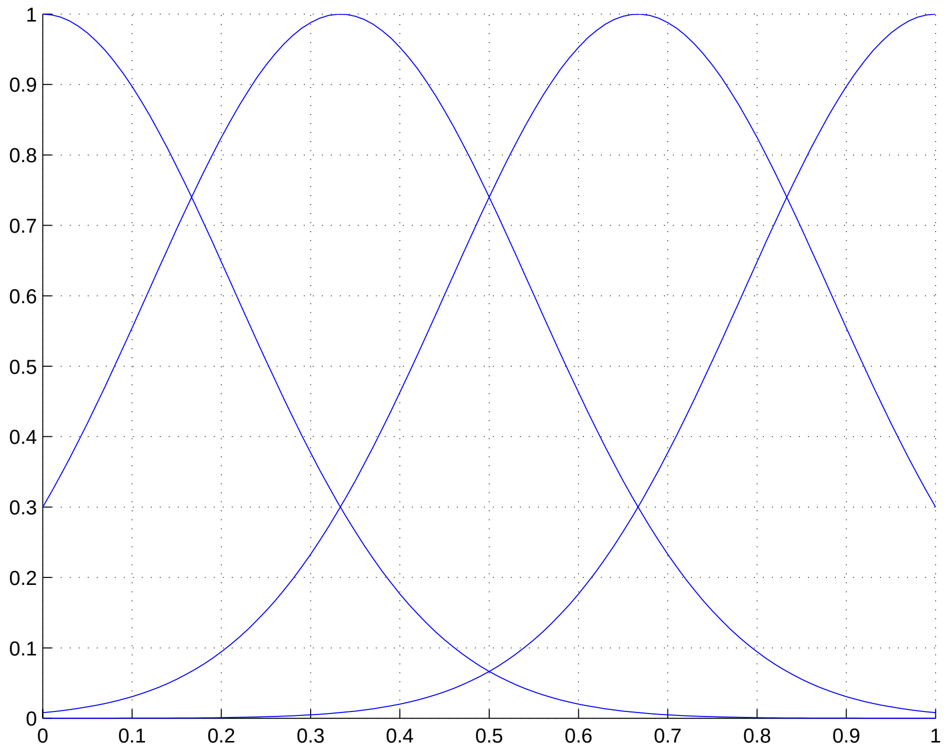


Figure 2: Initial width determination

After selecting the initial centers and widths, the OLS or FOS algorithms can be used to simultaneously reduce the number of initial centers and to find a set of corresponding weights. In reducing the number of centers, attention should be paid to the fact that there should be enough data samples for the evaluation of fitness for each center such that the features of the function associated with that center can be characterized. Ideally we should be able to completely determine the behavior of a continuous function by sampling theory, given enough discrete points from that function.

After the initial parameters, centers, widths and weights are found, the gradient method, described below, is used to optimize them according to the performance of the network. This is different from the traditional approaches in which the centers are kept fixed once they are selected. Haykin [10] discussed the supervised selection of centers in his book “Neural Networks-A Comprehensive Foundation”. The work done by Lowe (1989) on speech recognition using RBF networks indicates that nonlinear optimization of the parameters that define the activation functions of the hidden layer is beneficial when a minimal network configuration is required [16]. However, the issue of how the optimization procedure should be initialized was not addressed. Based on the center selection procedure, we can have a set of good parameters for the global optimization of the RBF network. These values are a set of good starting points for the gradient algorithm. Therefore the difficulty of finding a good set of starting points is greatly reduced, and the stability and convergent nature of the following tuning algorithm is ensured to some extent.

3.4 Tuning Algorithms

The parameter tuning in the training process can be divided into two distinct approaches: instantaneous and batch training modes. The instantaneous training mode uses only the information provided by a single training sample $\{\mathbf{x}(t), y(t)\}$, when the parameter vector is updated, whereas the batch training mode generally uses all the training data to adapt the parameters.

3.4.1 Instantaneous Mode Parameter Tuning

We define the performance function as,

$$e(\mathbf{x}^k) = (f(\mathbf{x}^k) - y^k)^2 \tag{3-6}$$

where $f(\mathbf{x}^k)$ is the network output for input $\mathbf{x}^k$, and y is the desired output for input $\mathbf{x}^k$. Using the chain rule, the derivative of the performance function with respect to different parameters can be given as:

$$\frac{\partial e}{\partial c_{ij}} = \frac{\partial e}{\partial f}\frac{\partial f}{\partial c_{ij}} = \frac{\partial e}{\partial f} w_i e^{-\frac{1}{2}\sum_j \frac{(x_j^k - c_{ij})^2}{\sigma_{ij}^2}} \frac{(x_j^k - c_{ij})}{\sigma_{ij}^2} \tag{3-7}$$

$$\frac{\partial e}{\partial \sigma_{ij}} = \frac{\partial e}{\partial f}\frac{\partial f}{\partial \sigma_{ij}} = \frac{\partial e}{\partial f} w_i e^{-\frac{1}{2}\sum_j \frac{(x_j^k - c_{ij})^2}{\sigma_{ij}^2}} \frac{(x_j^k - c_{ij})^2}{\sigma_{ij}^3} \tag{3-8}$$

$$\frac{\partial e}{\partial w_i} = \frac{\partial e}{\partial f}\frac{\partial f}{\partial w_i} = \frac{\partial e}{\partial f} e^{-\frac{1}{2}\sum_j \frac{(x_j^k - c_{ij})^2}{\sigma_{ij}^2}} \tag{3-9}$$

$$\frac{\partial e}{\partial w_j} = \frac{\partial e}{\partial f} x_j^k \tag{3-10}$$

where,

$$\partial e/\partial f = 2(f - y) \tag{3-11}$$

The update of the parameters is as follows,

$$\Delta c_{ij} = -\eta_{c_{ij}} \frac{\partial e}{\partial c_{ij}} \tag{3-12}$$

$$\Delta\sigma_{ij} = -\eta_{\sigma_{ij}} \frac{\partial e}{\partial \sigma_{ij}} \tag{3-13}$$

$$\Delta w_i = -\eta_{w_i} \frac{\partial e}{\partial w_i} \tag{3-14}$$

$$\Delta w_j = -\eta_{w_j} \frac{\partial e}{\partial w_j} \tag{3-15}$$

where η is the learning rate assigned during the parameter updating process. Each learning rate can be adjusted in its own way according to its influence on the performance function.

3.4.2 Batch Mode Parameter Tuning

In batch mode, we define the performance function as

$$E = \frac{1}{M} \mathop{}_{k=1}^{M} e_k \ , \tag{3-16}$$

where

$$e_k = e(\mathbf{x}^k) = (\, f(\mathbf{x}^k) - y^k\,)^2$$

and $f(\mathbf{x}^k)$ is the network output and y^k is the desired output, given the input $\mathbf{x}^k$, M is the number of training samples. Then find derivatives of the performance function with respect to all parameters:

$$\frac{\partial E}{\partial c_{ij}} = \frac{2}{M} \mathop{}_{i=1}^{M} \frac{\partial e_i}{\partial c_{ij}} \tag{3-17}$$

$$\frac{\partial E}{\partial \sigma_{ij}} = \frac{2}{M} \mathop{}_{i=1}^{M} \frac{\partial e_i}{\partial \sigma_{ij}} \tag{3-18}$$

$$\frac{\partial E}{\partial w_i} = \frac{2}{M} \mathop{}_{i=1}^{M} \frac{\partial e_i}{\partial w_i} \tag{3-19}$$

$$\frac{\partial E}{\partial w_j} = \frac{2}{M} {}_{i=1}^{M} \frac{\partial e_i}{\partial w_j} \tag{3-20}$$

where $\frac{\partial e_i}{\partial c_{ij}}, \frac{\partial e_i}{\partial \sigma_{ij}}, \frac{\partial e_i}{\partial w_i}, \frac{\partial e_i}{\partial w_j}$ are given by *Equation (3-7), Equation (3-8), Equation (3-9), Equation (3-10)*, respectively.

So, for batch mode the parameter update is as follows:

$$\Delta c_{ij} = -\eta_{c_{ij}} \frac{\partial E}{\partial c_{ij}} \tag{3-21}$$

$$\Delta \sigma_{ij} = -\eta_{\sigma_{ij}} \frac{\partial E}{\partial \sigma_{ij}} \tag{3-22}$$

$$\Delta w_i = -\eta_{w_i} \frac{\partial E}{\partial w_i} \tag{3-23}$$

$$\Delta w_j = -\eta_{w_j} \frac{\partial E}{\partial w_j} \tag{3-24}$$

Like the instantaneous mode, each learning rate can be adjusted according to its influence on the performance function.

It is worth mentioning that the updating of w_j has to be very careful to make sure the tuning process is stable, otherwise the performance of the network could be easily degraded inappropriately. This is because that w_j corresponds to the linear term. Theoretically, all parameters can be updated appropriately if using proper learning rate or step size each time. Therefore there is a problem of proper step size selection (see section 2.4).

4. THE PAPERMAKING PROCESS

4.1 Background

Papermaking is a very complex process. Many parameters affect the product quality. The paper machine is a very important part of the pulp and paper manufacturing process. The most important properties of the paper emerging from a paper machine are brightness and opacity. A major goal of the paper making process control is the reduction of variability of these two properties. Improved paper quality control would not only increase the efficiency of the paper making process, but it would also reduce chemical usage and increase the yield and decrease the downtime of the paper machines, resulting in reduced waste and increased capital savings. Due to the complexity of the process operation and requirements of high quality product, it’s very important to control the process variables such as material feeding flow, temperature, moisture, etc. The knowledge concerning the operation includes complex technologies from different areas. Traditionally the engineers in the paper mill either use their past experiences or follow some simple models to control the paper machine. However, this type of operation doesn’t provide optimal performance, especially for the fast running paper machine. Furthermore, on-line measurement of many important variables is either unreliable or impossible due to sensor technology limitations. In many cases, control is dependent on unreliable, noisy or manually gathered data. Under these conditions, even experienced operators find it difficult to deal with operations such as quality control and operation optimization.

This thesis shows some of the potential of applying the artificial neural networks to the pulp and paper process. In addition to our work in applying the RBF network in the pulp and paper industry, Chen et al [17] have also used RBF networks to overcome the problems of noisy data and modeling of the nonlinear processes in the industry.

4.2 Paper Making Process Description

Paper is a structure formed mainly from wood fibers with or without various additives. By selecting the types of fibers, additives and their treatment in the process, a very wide range of pulp and paper products is made. A typical papermaking process flowchart is shown in Figure 3. As shown, certain additives are mixed with the refined pulp according to the type of paper being made. The mixture is then passed through forming and drying sections and then the coating sections for the addition of more additives to ensure the desired paper quality [18]. Note that measurements are taken at different stages to check the paper quality. Some measurements are on-line and some off-line and in laboratories. For example, the final optical properties, such as brightness and opacity, are measured on-line and are used by operators for quality control, whereas the softwood and hardwood brightness is measured in laboratories. Any on-line modeling of these properties would be very helpful for automatic control.

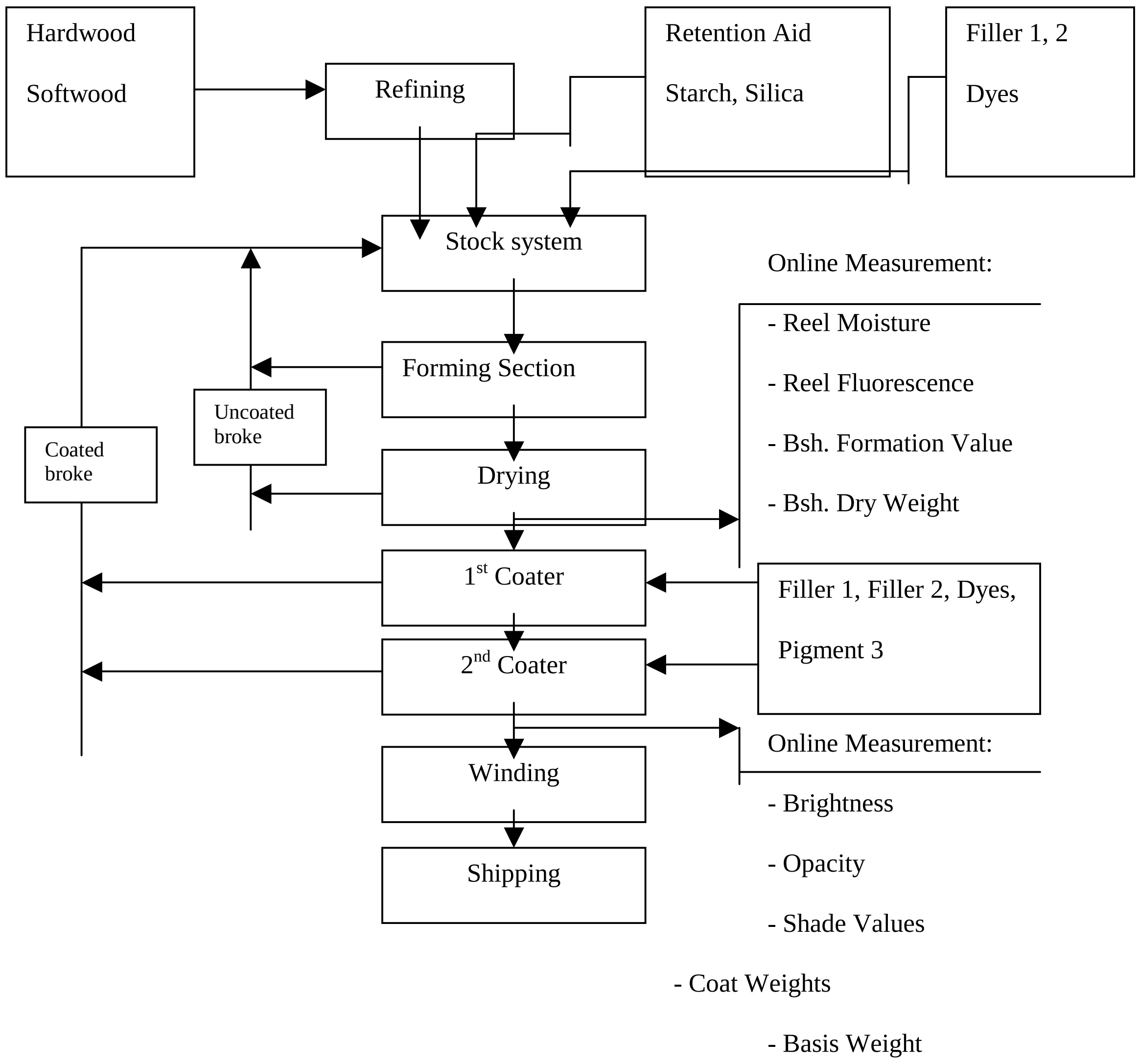


Figure 3: Optical properties process flow chart

4.3 Data Collection

The optical property data was collected in averaged 15-minute intervals from March through September 1996. The averaged 15-minute data is the average of the data over 15 samples because the data is measured every one minute. Sixty-six variables were collected and are listed in Table 1. The first and fifth columns are the variable number, the second and sixth columns are the corresponding variable name, and the third and seventh columns indicate the valid range for the variable. Different variables are measured at different stages along the overall papermaking process, therefore, there are time delays between input process variables at different stages and the outputs of the process, namely brightness and opacity. The time delay means how long it will take for the measured input variable to have an effect in the output variable. Time delays for the process under investigation are also included in the fourth and eighth column of Table 1. The process operators have provided these values.

No.	Variable	Valid Range	Delay	No.	Variable	Valid Range	Delay
1	Brightness	78-100	0	34	Basesheet brightness	75-100	0
2	Opacity	88-100	0	35	Basesheet opacity	80-100	0
3	L-value	90-98	0	36	Basesheet L-value	84-95	0
4	A-value	-0.7-0.6	0	37	Basesheet A-value	-.7-.1	0
5	B-value	-2.25-3	0	38	Basesheet B-value	-.5-3	0
6	Tobias Mottle	10-80	0	39	Basesheet formation value	5-20	0
7	Filler 1	0-85	40 sec	40	Basesheet fluorescence	0-85	0
8	Filler 2	40-300	40 sec	41	Basis weight (target)	50-80	0
9	Total ash %	5-20	0	42	Basis weight (measured)	0-100	0
10	Dye 1	0-2.6	40 sec	43	Retention aid	0-5	0
11	Dye 2	0-4	40 sec	44	Starch	0-60	0
12	Dye 3	0-6	40 sec	45	Silica	0-4	0
13	Dye 4	0-4	40 sec	46	Refining Horse Power (hw)	0-8	0
14	Softwood %	0-40	40 min	47	Hardwood Freeness	100-600	0
15	Hardwood %	0-60	40 min	48	Refining Horse Power (sw)	0-10	0
16	Coated broke %	0-60	40 min	49	Softwood Freeness	400-700	0
17	Recycle %	0-10	40 min	50	Softwood Dirt	0-1	0
18	Uncoated broke %	0-36	40 min	51	Softwood Fines	3-6	0
19	Hardwood brightness	85-90	0	52	Softwood Drylap	0-62	0
20	Softwood brightness	86-90	0	53	Softwood Purchased	0-52	0
21	Coating clay1	0-810	4 hours	54	Softwood Slush	0-100	0
22	Coating clay2	0-800	4 hours	55	Softwood Welap	0-60	0
23	Coating pigment 1	200-400	4 hours	56	Softwood Fiberlength	1.5-2.5	0
24	Coating pigment 2	0-55	4 hours	57	Softwood pH	5.5-7	0
25	Coating pigment 3	0-100	4 hours	58	Hardwood Dirt	0-1	0
26	Coating dye 5	0-.4	4 hours	59	Hardwood Fines	4.5-7.5	0
27	Coating dye 6	0-40	4 hours	60	Hardwood Drylap	0-30	0
28	Coating dye 7	0-.4	4 hours	61	Hardwood Purchased	0-80	0
29	Coating weight top	0-10	4 hours	62	Hardwood Slush	40-100	0
30	Coating weight bottom	0-10	4 hours	63	Hardwood Welap	0-60	0
31	Reel fluorescence	0-6.2	4 hours	64	Hardwood Fiberlength	.5-1	0
32	Reel moisture	0-6	0	65	Hardwood pH	5-7	0
33	Basesheet dry wt.	0-70	0	66	Paper grade	N/A	0

Table 1: Optical property variables

4.4 Data Filtering

Once data is obtained it is important to understand when it can be considered valid. If invalid data is found, it must be removed. The goal is to obtain a valid database that can be used as a foundation for experiments to follow.

The data collection system at the company with whom we are cooperating on this project has an internal filter for identifying obvious invalid data. This filter fills a column of the data file and provides numbers ranging from 0 to 100. A value of the filter variable less than 99 indicates that all the values in that row are invalid. Therefore, rows within the database having a filter value less than 99 were deleted. There are other times during plant operation when erroneous data is collected. A few examples are when the plant is in a shutdown, when the reel of paper is being unloaded, or when the stream of paper has broken. The plant's data collection system collects all variables during these invalid times of operation. Fortunately, such regions of data result is database elements that are text strings briefly explaining the type of malfunction that occurred and are therefore easily recognizable. The approach taken was to replace such database elements with a negative integer that represents the error. This is useful because it avoids simply deleting the whole row of data, which could contain some valid data, and yet allows the data to be entered into Matlab (text strings are not allowed), which has been used for building prototype of our models. Therefore, the continuity of the data can be kept.

In addition to the above-mentioned filtering, other cleaning techniques were investigated in attempt to remove noise from the data. These techniques are described below.

4.4.1 Discrete Fourier Transform (DFT) Method

The Fourier transform technique allows analysis of the data frequency spectrum. Applying a low pass filter to the Discrete Fourier Transform (DFT) removes high frequency components, which can be attributed to noise. Then, by applying the discrete inverse Fourier Transform, cleaned data is obtained. For the data of this process, network response degraded after applying this technique. Therefore, it is not appropriate to use DFT as the filtering for the data. This could be partly because we don't know exactly what part of frequency should be kept, and because of discontinuity of the data at hand.

4.4.2 Moving Average Method

The moving average method is based on the idea that the value of a variable measured at time *t* should be similar to the value of the variable measured at *t-1* and *t+1*. To be simple, we just take the average of the three values as the value at time *t*, and repeat the calculation for all data samples. This method also did not work because the neural network performance was degraded after applying this technique.

4.4.3 Adjacent Sample Method

The adjacent sample technique is based on the idea that variables should change, on a sample by sample basis, in a relatively consistent manner. That is to say there are limits to what can be considered valid step sizes. For instance, *dye 1* has been stated as ranging from zero to about two and a half in Table 1. It is therefore reasonable to assume that any adjacent samples of *dye 1* with a step size greater than two and half are invalid. Further, any variable data that exceeds its valid range is also erroneous. Such data makes no physical sense, and should not be considered by the network; removing this type of data is the result of the adjacent sample method.

This technique is most suitable for the data of this project, because it keeps the original data as intact as possible, further data corruption resulting from many unknown factors is avoided. A list of each variable's valid range, as given by plant engineers, is listed in Table 1.

4.5 Data Processing

4.5.1 Paper Grade

There are twenty-seven different grades of paper in the collected data. It is informative to conduct the analysis with two types of data, namely data for one particular grade of paper and data for all grades of paper combined. The data for one grade of paper has the advantage of considering only one state of operation, and the disadvantage of having discontinuities. Because all twenty-seven grades of paper are produced in a continuous manner, there are many switches between different grades during the continuous papermaking process. The data for all grades of paper has the advantage of being continuous, and the disadvantage of considering many states of operation.

The database for all grades of paper is the foundational database. The database for each grade of paper is constructed from the foundational database, where variable number 66 allows regions of different grades of paper to be recognized and divided. The exclusion of a ±4 hour time span, at each grade transition, aids in avoiding non-static effects caused from the transition from one grade of paper to another.

4.5.2 Data Normalizing

Since different input variables use different measurement units, it was decided that they be normalized. For our experiment, we normalized the data during the calculation for the network, mainly between 0 and 1. Each value of a variable will be normalized as follows:

$$normalized\ value = \frac{original\ value - valid\ minimum}{valid\ maximum - valid\ minimum} \qquad (4\text{-}1)$$

Figure 4 shows the six variables chosen for the work in this thesis. These six variables were chosen via a sensitivity analysis as well as input from plant engineers.

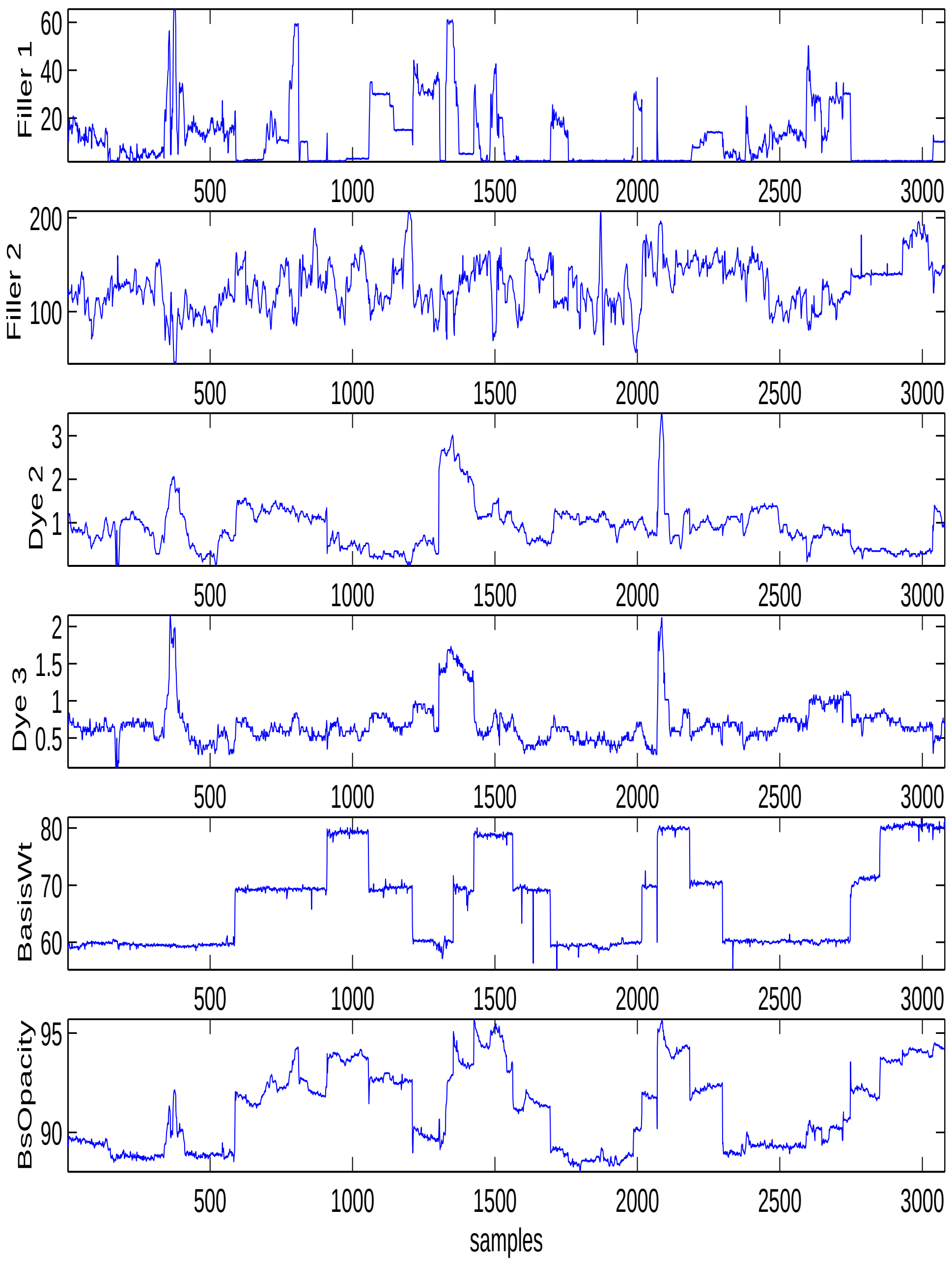


Figure 4: The nature of different variables used in the RBF network modeling

4.6 Pearson Correlation Coefficient Analysis

Table 2 lists the results of the Pearson correlation coefficients (see section 3.2) for the top twenty ranking optical property variables with respect to brightness and opacity. A coefficient of ±1 represents a high positive and negative correlation respectively, whereas a coefficient of 0 represents a total lack of correlation.

For the following experiments, basesheet opacity and brightness have been used as the network outputs. According to the experience from the plant engineer, analysis for one grade and all grades of paper given in the data set, and by trial and error, the 7th, 8th, 11th, 12th, 42th, 35th variables were selected for the modeling of the next chapter. Note that these don't correspond to the ranking of the following table because the ranking of the table is based on linear analysis and that other factors, as mentioned above, have been taken into consideration.

	Final Brightness		Final Opacity		Basesheet Brightness		Basesheet Opacity
variable #	r	variable #	r	variable #	r	variable #	r
31	0.74184	35	0.919576	12	-0.67631	2	0.919576
22	0.512674	33	0.833837	6	0.637531	33	0.919474
27	0.496226	41	0.828846	38	0.54687	42	0.919319
21	-0.40079	42	0.827267	11	-0.52915	41	0.91739
4	0.393296	43	-0.6946	52	-0.48727	43	-0.75101
8	0.322349	9	0.504938	53	0.437898	45	0.586874
16	-0.3114	45	0.482691	23	-0.42135	9	0.494397
46	0.286874	8	0.430896	55	0.416882	8	0.473309
45	0.278748	38	-0.39144	60	-0.41285	46	0.348353
49	-0.26303	6	-0.36977	28	0.397375	38	-0.32167
9	0.256427	40	0.341049	40	-0.37911	39	0.310457
47	-0.24081	39	0.328072	62	0.339241	40	0.301756
11	-0.2344	55	-0.31161	4	0.334677	44	0.277703
53	0.233842	28	-0.28652	25	0.327019	24	-0.27271
34	0.225217	46	0.278851	51	-0.30771	55	-0.26544
5	-0.22109	52	0.275016	31	0.289057	28	-0.26459
30	0.214442	11	0.274023	48	-0.26985	6	-0.25745
3	0.203481	24	-0.24507	26	-0.25742	48	0.252381
12	-0.19461	18	-0.24015	30	-0.25684	18	-0.25143
60	-0.18938	22	-0.23044	15	-0.24387	52	0.205011

Table 2: Ranking of optical properties variables according to Pearson sensitivity analysis (top 20)

5. EXPERIMENTS

This chapter provides the results of the proposed tuning algorithms on two different sets of data. The first one is a one-dimensional mathematically generated data to show the effectiveness of the proposed algorithm. The second set is multi-dimensional data from the papermaking process.

5.1 Effect of Parameter Tuning Algorithm

To show the effectiveness of the proposed parameter tuning algorithm, one needs to compare it with the current RBF techniques. A multi-resolution radial basis function (RBF) network trained with FOS [15] has provided very promising results. Therefore, it has been used here to provide the initial conditions for the parameter tuning algorithm.

The intention is to see if the proposed algorithm can improve upon RBF/FOS. Consider the following function

$$f=5*\sin(x^2)+\sin(x)+x, \qquad x\in[0, 2\pi].$$

The objective is to use a fixed number of nodes and approximate this function once using RBF/FOS alone without any tuning and once with the proposed tuning algorithm. For this example the final number of RBF nodes was set to be 10. Starting with 100 training samples and using a 5-level multi-resolution grid, the FOS algorithm was first used to find the 10 most important RBF nodes, then the parameter tuning algorithm proceeded to tune the network parameters. Using 900 testing samples, the results for both FOS and parameter tuning algorithm are shown in Figures 5 through 10. Figure 5 provides network training, testing and testing error after FOS. The network output and the real value were overlaid in this figure. Figure 6 provides the scatter plot for the network output and actual values. Figures 7 through 10 provide the network performance after applying the parameter tuning algorithm. Figure 7 provides network training, testing and testing error. Figure 8 provides the scatter plot for the network output and actual values. Figure 9 shows how the network error is reduced during the parameter tuning process; the dashed line is for network training error and the solid line is for network testing error. Figure 10 provides the Pearson correlation coefficient for the parameter tuning iterations. The dashed line is for network training data and the solid line is for network testing data. Note that the application of the proposed training algorithm has improved the performance of the RBF/FOS network.

In this experiment the learning rate was set at 0.005. Note that the iterative tuning process is very stable due to 1) the dimensionality of the problem and 2) small step size. In our experiments with high dimensional data, some oscillations have been observed. However, if the initial conditions are good and the step size is appropriate then the procedure has usually converged to a solution. Table 3 is a comparison of errors and Pearson correlation coefficients for the RBF/FOS network before and after the parameter tuning process. Note that the error mentioned above is the Mean Square Error (MSE). The linear regression for the same data set was also conducted for comparison. The results have also been provided in Table 3.

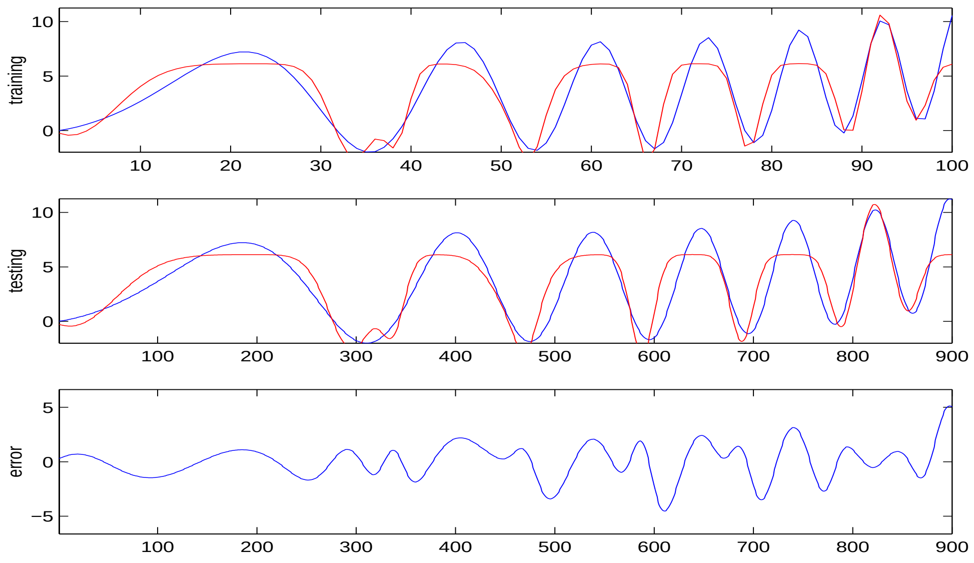


Figure 5: Network training, testing and error (using FOS)

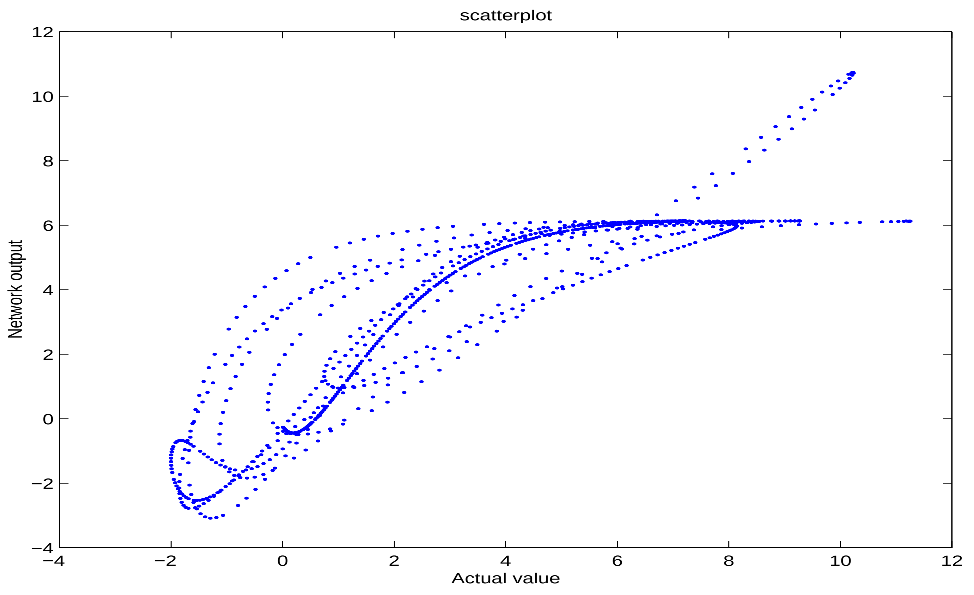


Figure 6: Scatter plot for the actual value and network output (using FOS)

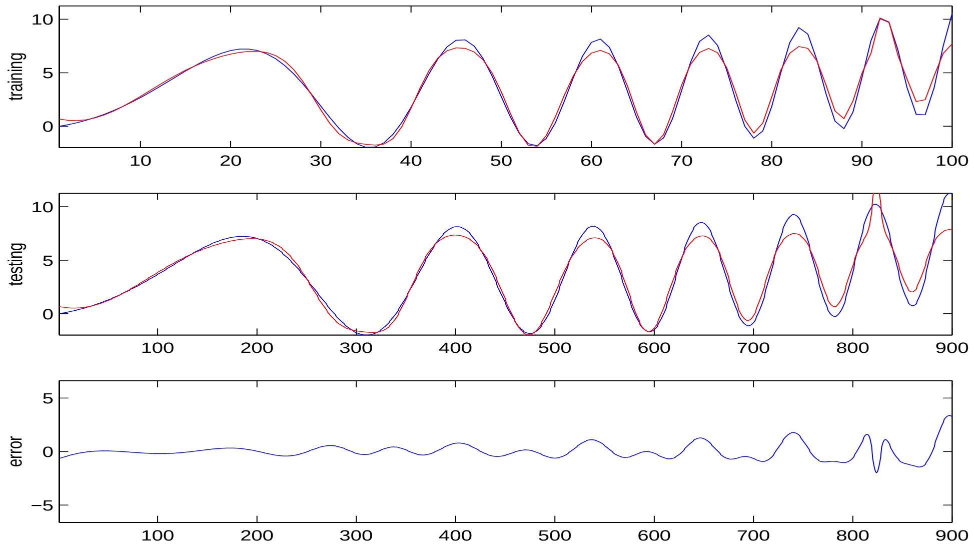


Figure 7: Network training, testing and error (after tuning algorithm)

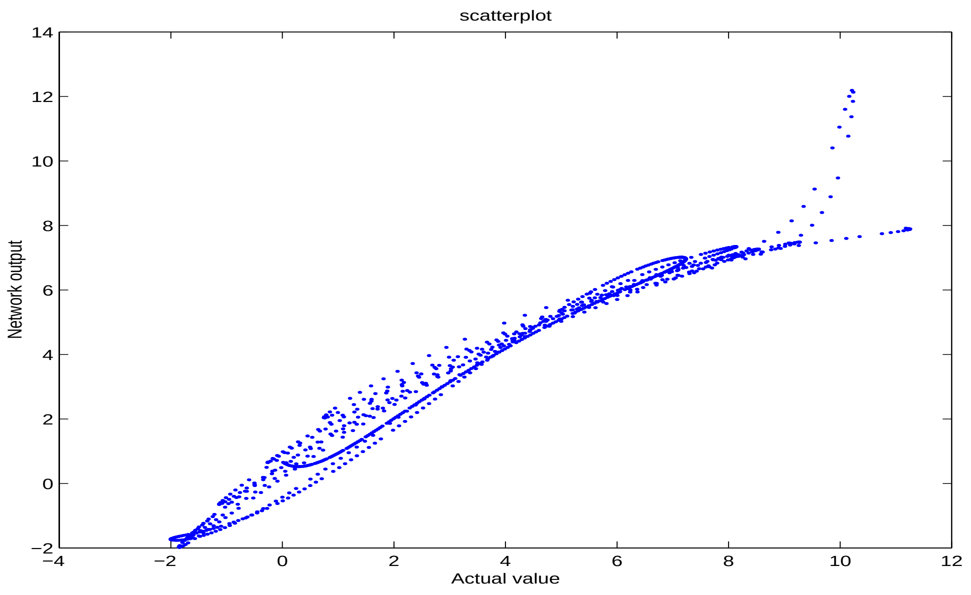


Figure 8: Scatter plot for the actual value and network output (after tuning algorithm)

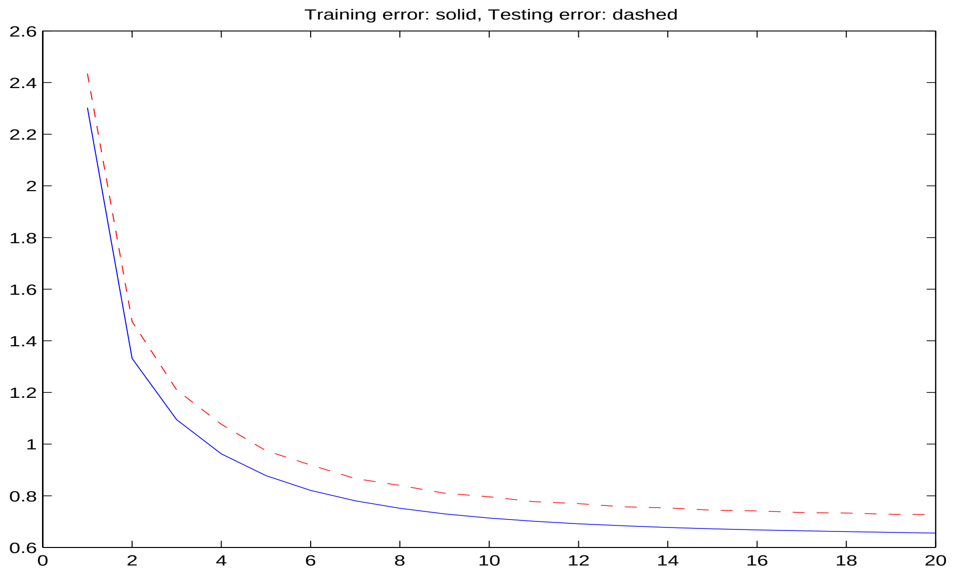


Figure 9: Error for parameter tuning iterations

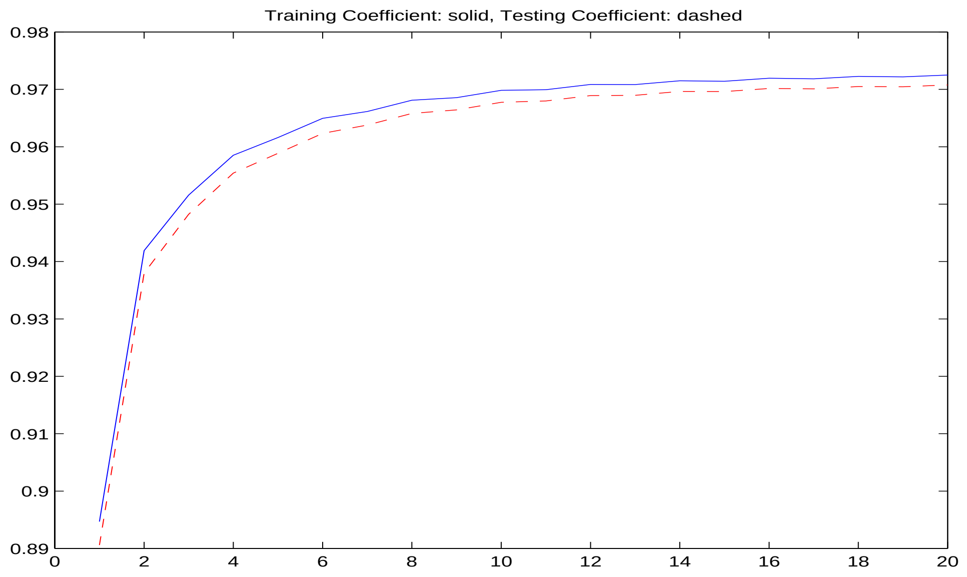


Figure 10: Pearson correlation coefficient for parameter tuning iterations

	Before the tuning process	After the tuning process	Linear regression
Training error	2.3026	0.4062	N/A
Testing error	2.435	0.5029	11.34
Pearson correlation coefficient (training)	0.8947	0.9852	N/A
Pearson correlation coefficient (testing)	0.8906	0.9817	0.19

Table 3: Comparison of errors and Pearson correlation coefficients before and after the parameter tuning using FOS

5.2 Training Data Selection

Now the network is used for the optical properties modeling problem. For the experiments described in this section, the opacity data set for all grades of paper was used. The total number of samples used for training and testing the network is 3079. The sampling period is 15 minutes and each sample is the average over the minute data. In the instantaneous mode of the tuning process, 302 training samples (approximately 10% of the data set) and 2777 testing samples were used. For the batch mode tuning process, 457 samples (approximately 15% of the data set) were selected for training and 2622 samples were used for testing.

In order to see the effect of the training data selection, histograms of the training data set and the whole data set for each of the selected 6 variables are shown in figures 11 through 16. In each figure, the left Y-axis denotes the histogram for the whole data set and the right Y-axis denotes the histogram for the selected training data set. As can be seen that the distribution features of each variable have been represented well by the training set. The X-axis consists of ranges for the variable, from the valid minimum to the valid maximum.

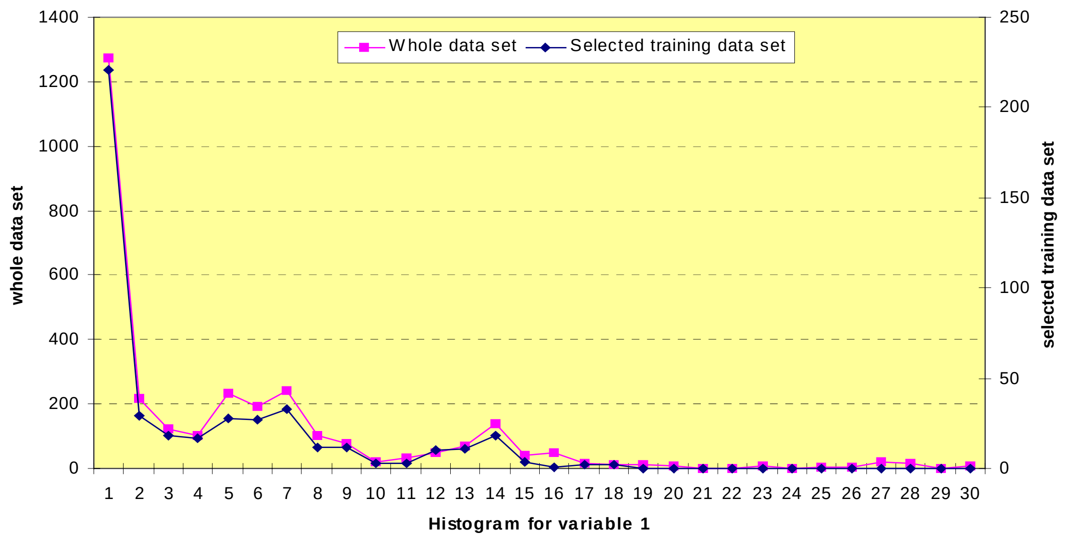
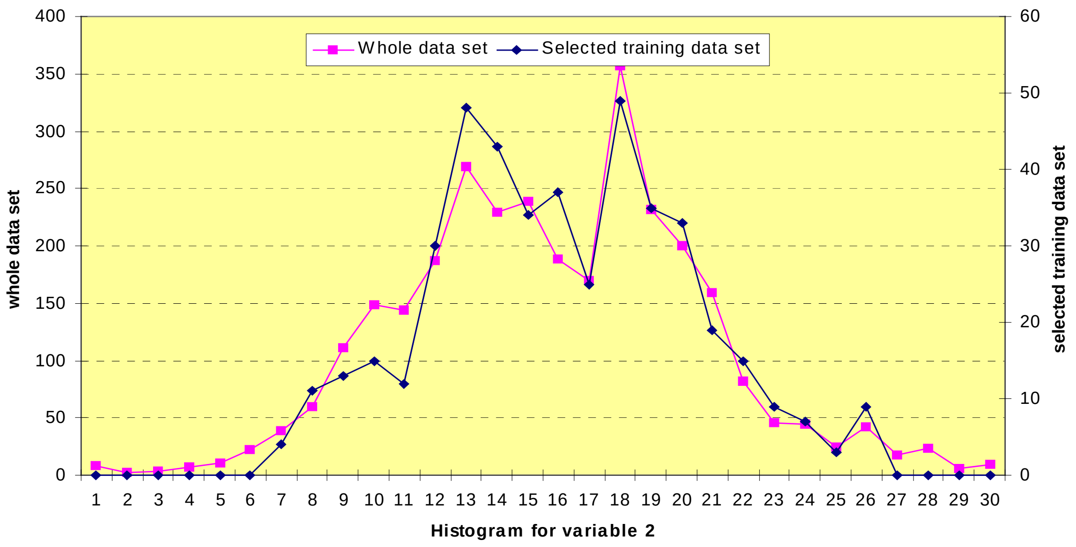


Figure 11: Histogram for variable 1



\F

igure 12: Histogram for variable 2

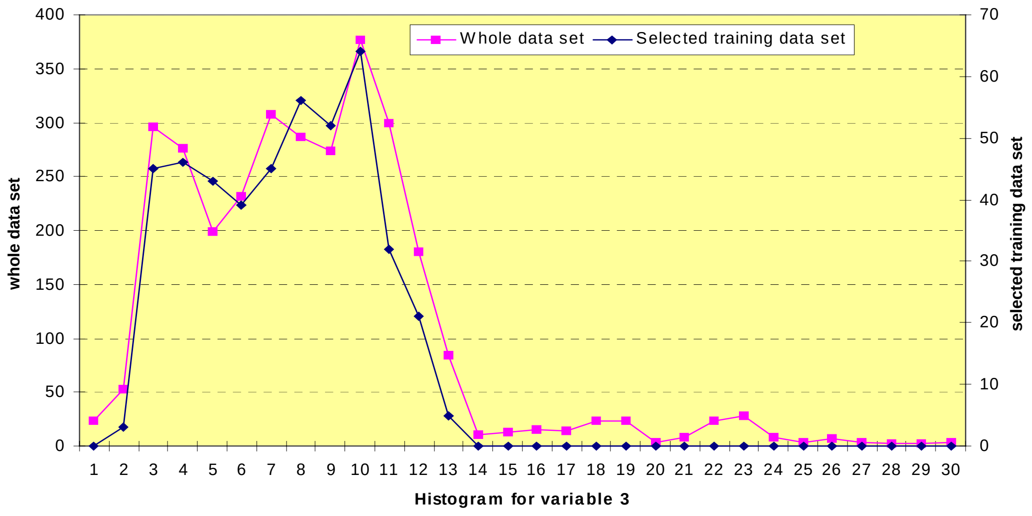


Figure 13: Histogram for variable 3

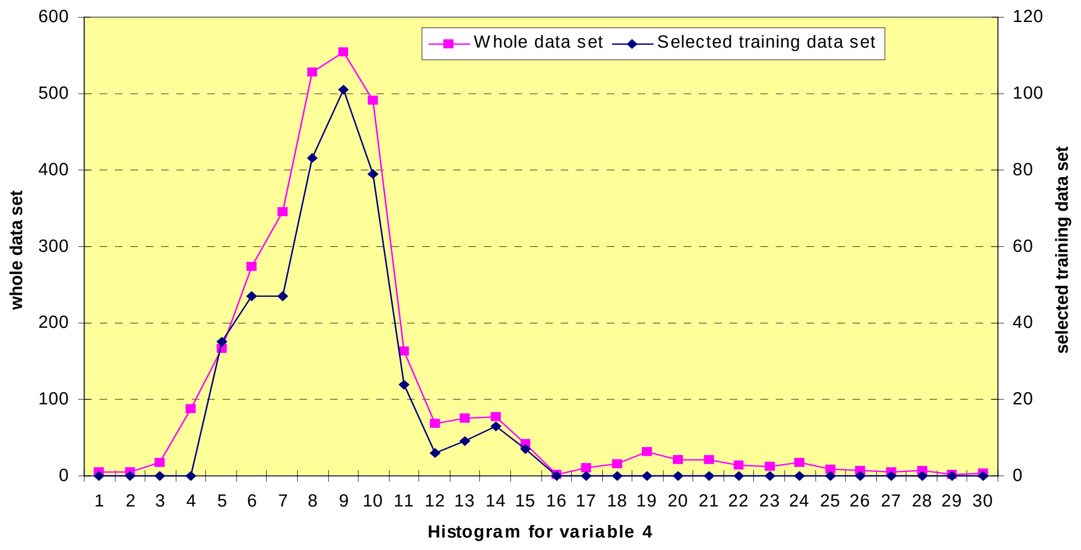


Figure 14: Histogram for variable 4

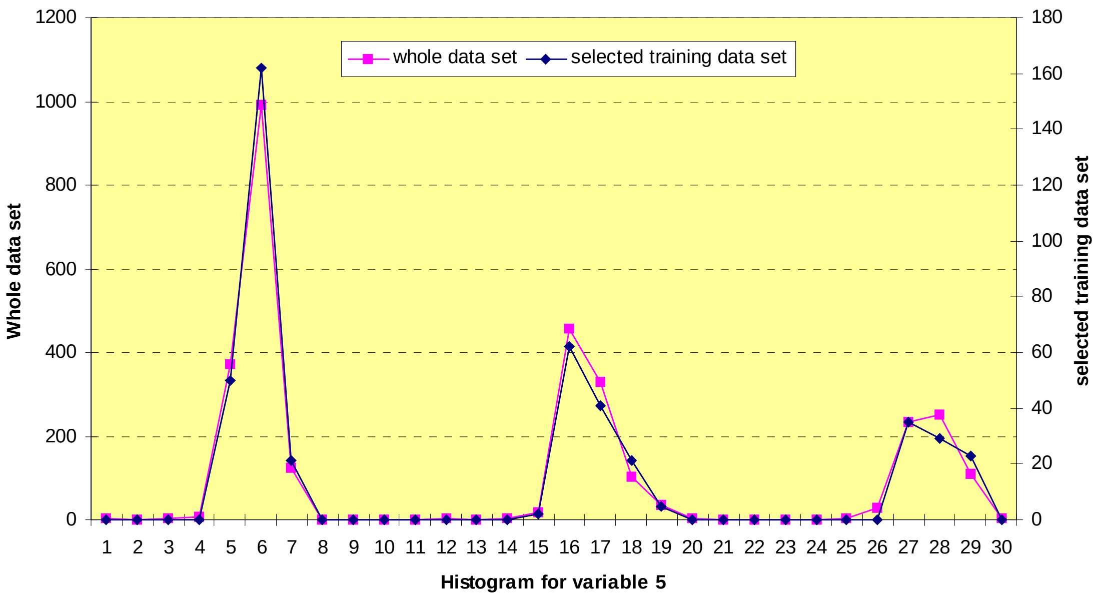


Figure 15: Histogram for variable 5

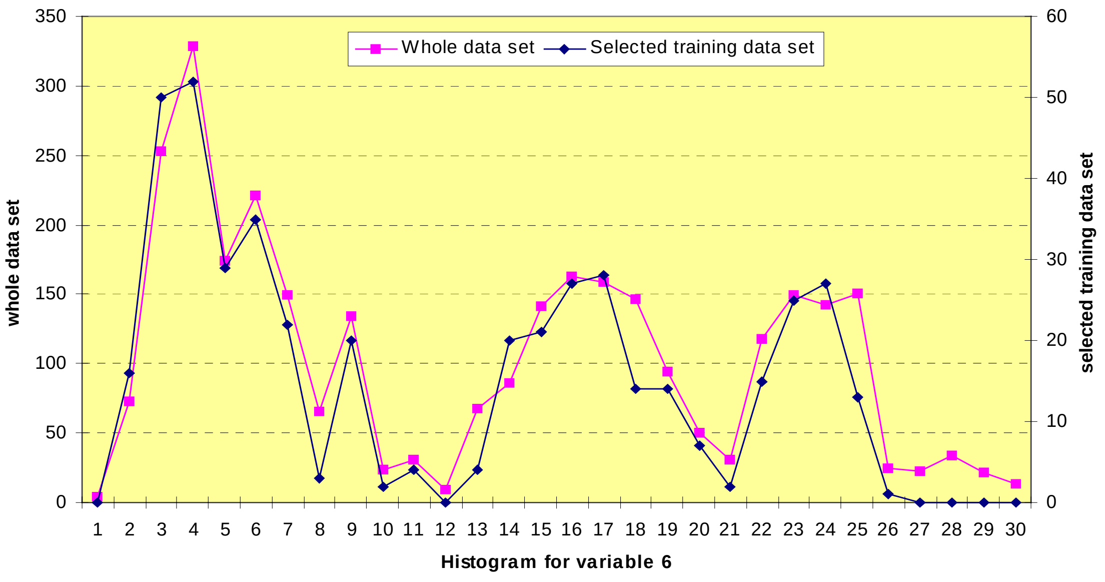


Figure 16: Histogram for variable 6

5.3 Network Training And Testing

5.3.1 Selecting The Training Data From The Entire Data Set

Starting with the training points, as described in section 5.1, we used the initial center assignment procedure and the OLS algorithm to find a set of centers and their associated weights. The *instantaneous mode* (section 3.5.2) and *batch mode* (section 3.5.3) parameter tuning algorithms were used to further optimize the network.

5.3.1.1 Instantaneous Mode For Parameter Tuning

For instantaneous mode parameter tuning, 10 centers were selected. The instantaneous tuning process then modified the network parameters (centers, widths and weights). The results are shown in figures 17 through 20. Figure 17 provides network training, testing and testing error. The network output and the actual value were overlaid in this figure. Figure 18 provides the scatter plot for the network output and actual values. Figure 19 provides the network error for the parameter tuning iterations, the dashed line is for network training error and the solid line is for network testing error. Figure 20 provides the Pearson correlation coefficient for the parameter tuning iterations. The dashed line is for network training data and the solid line is for network testing data. For Figure 17 through Figure 20, the learning rate is set to 0.2 for the first 300 iterations, then changed to 0.1 for the rest. Table 4 is a comparison of errors and Pearson correlation coefficients for the network before and after the parameter tuning process. Note that the dotted line is the network output and the solid line is the actual values in the following network training and testing figures.

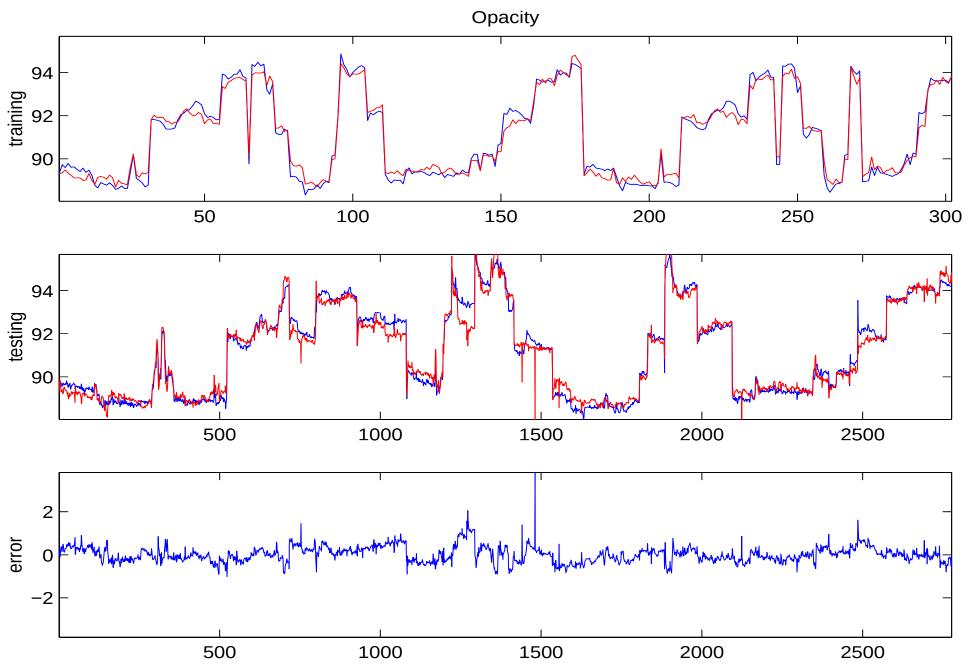


Figure 17: Network training, testing and error

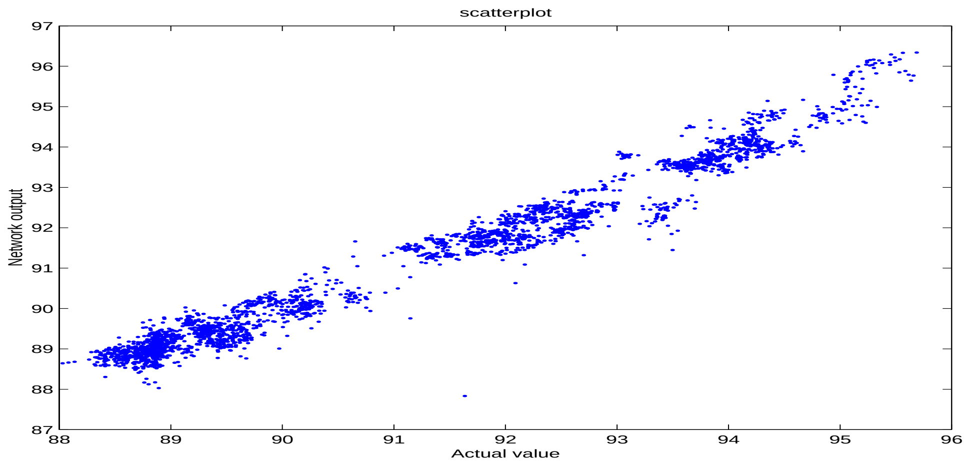


Figure 18: Scatter plot for the actual value and network output

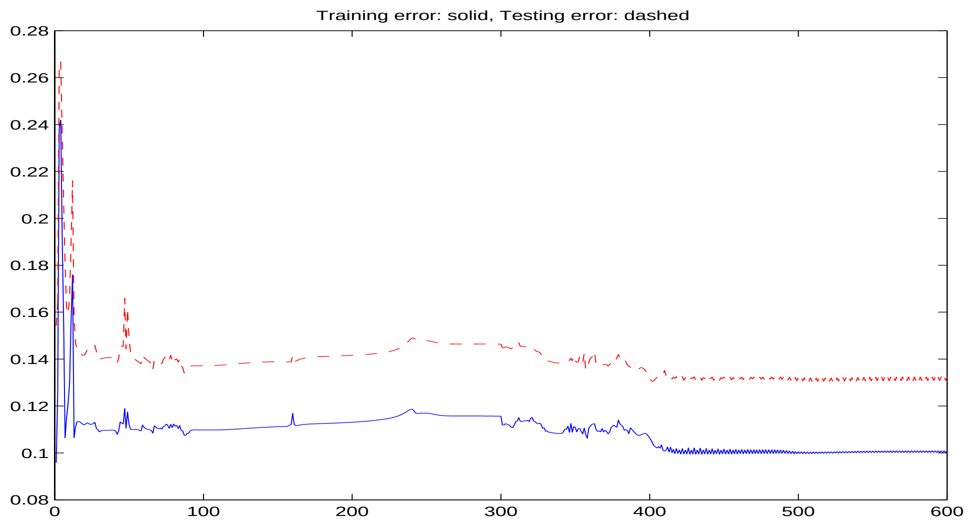


Figure 19: Error for parameter tuning iterations

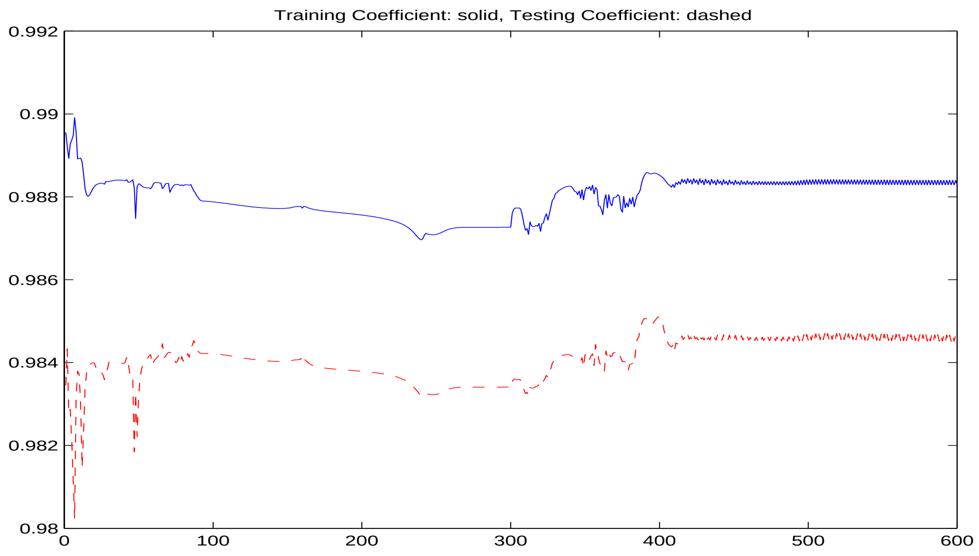


Figure 20: Pearson correlation coefficient for parameter tuning iterations

	Before the tuning process	After the tuning process
Training error	0.0958	0.1009
Testing error	0.1542	0.1324
Pearson correlation coefficient (training)	0.9896	0.9883
Pearson correlation coefficient (testing)	0.9835	0.9845

Table 4: Comparison of errors and Pearson correlation coefficients before and after the parameter tuning

As we noted, before the tuning process began, the training error was 0.0958, the testing error was 0.1542, the Pearson correlation coefficient for the training data set was 0.9896 and for the testing data set was 0.9835. After the 600 iterations of the parameter tuning process, the training error was 0.1009, the testing error was 0.1324, the Pearson correlation coefficient for the training data was 0.9883, and the Pearson correlation coefficient for the testing data was 0.9845.

As we can see, both the network testing error and Pearson correlation coefficient for the testing data set have been improved, but the network training error and the Pearson correlation coefficient for the training data set have been degraded a little, after the parameter tuning process. The important point here is the fact that the testing or the network generalization has improved. The training degradation could be due to the step size or the learning rate of the process, and only the error for one sample was considered for the parameter. Because of the extensive computational effort of this mode, only 302 samples were selected for this mode and the process ended at 600 iterations.

5.3.1.2 Batch Mode for Parameter Tuning

For the first experiment of batch mode parameter tuning, 10 centers were selected. The results are shown in Figure 21 through Figure 24. Figure 21 provides network training, testing, and testing error. The network output and the real value were overlaid in this figure. Figure 22 provides the scatter plot for the network output and actual values. Figure 23 provides the network error for the parameter tuning iterations; the dashed line is for network training error and the solid line is for network testing error. Figure 24 provides the Pearson correlation coefficient for the parameter tuning iterations; the dashed line is for network training data and the solid line is for network testing data. For figures 21 through 24, the learning rate was selected to be 0.15 for the first 220 iterations, then changed to 0.05 for iterations 221 to 1134, and to 0.02 for the rest. Table 5 is a comparison of errors and Pearson correlation coefficients for the network before and after the parameter tuning process.

	Before the tuning process	After the tuning process
Training error	0.0801	0.0454
Testing error	0.1359	0.1253
Pearson correlation coefficient (training)	0.9897	0.9942
Pearson correlation coefficient (testing)	0.9844	0.9862

Table 5: Comparison of errors and Pearson correlation coefficients before and after the parameter tuning

It can been seen that from the above table, before the tuning process began the training error was 0.0801, the testing error was 0.1359, the Pearson correlation coefficient for the training data set was 0.9897, and the Pearson correlation coefficient for the testing data set was 0.9844. After the 2134 parameter tuning iterations, the training error was 0.0454, the testing error was 0.1253, the Pearson correlation coefficient for the training data was 0.9942, and the Pearson correlation coefficient for the testing data was 0.9862.

As we can see, both the network training and testing error and Pearson correlation coefficient for the training data set and testing data set have been improved. We can also see the testing error and Pearson correlation coefficient follow closely with the training process, which indicates the effectiveness of the tuning algorithm in improving the network generalization ability.

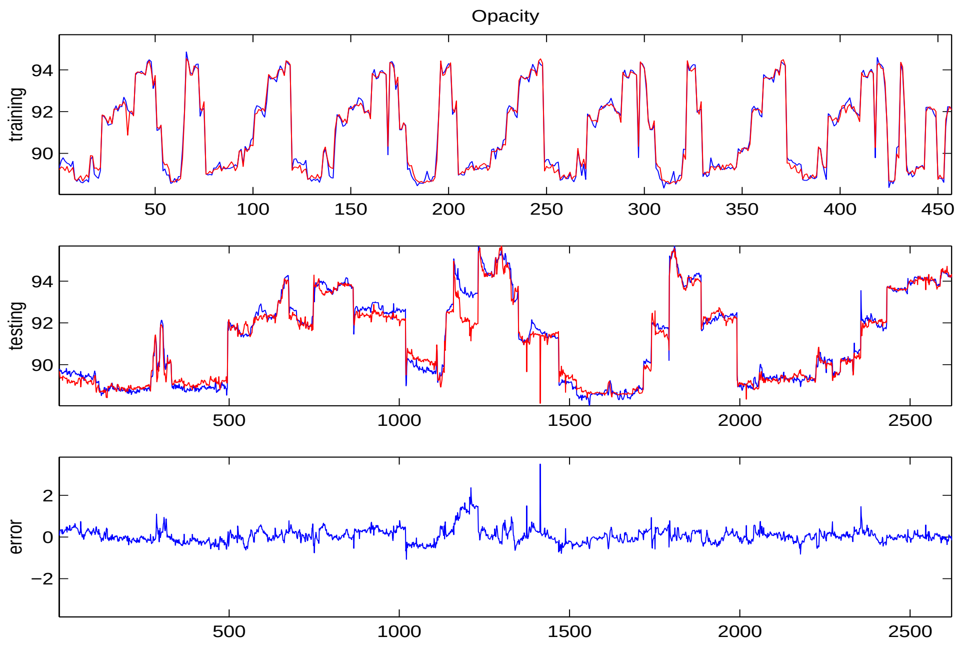


Figure 21: Network training, testing and error

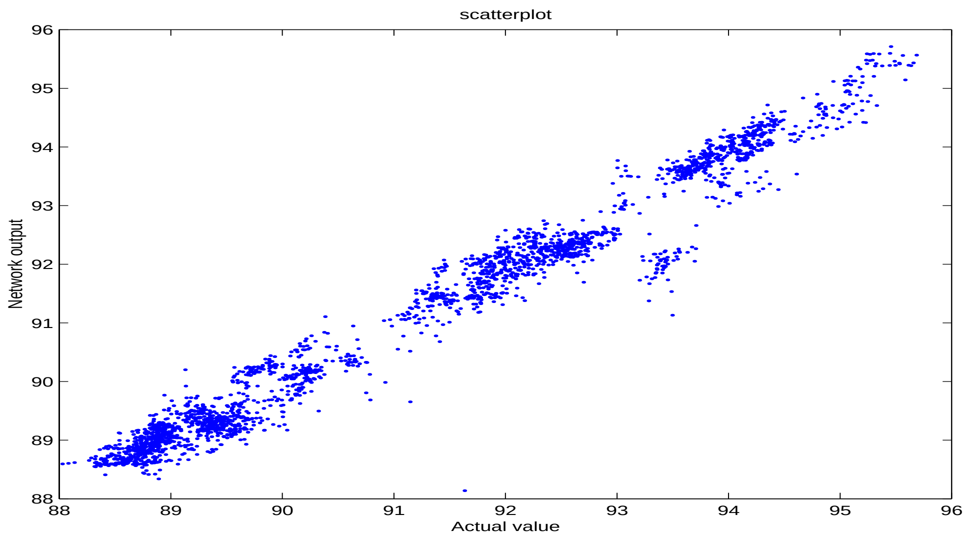


Figure 22: Scatter plot for the actual value and network output

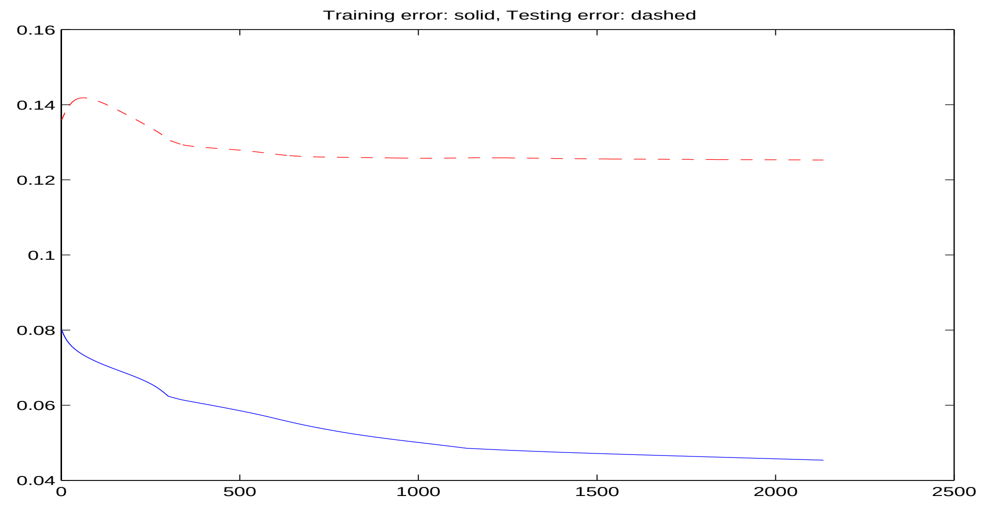


Figure 23: Error for parameter tuning iterations

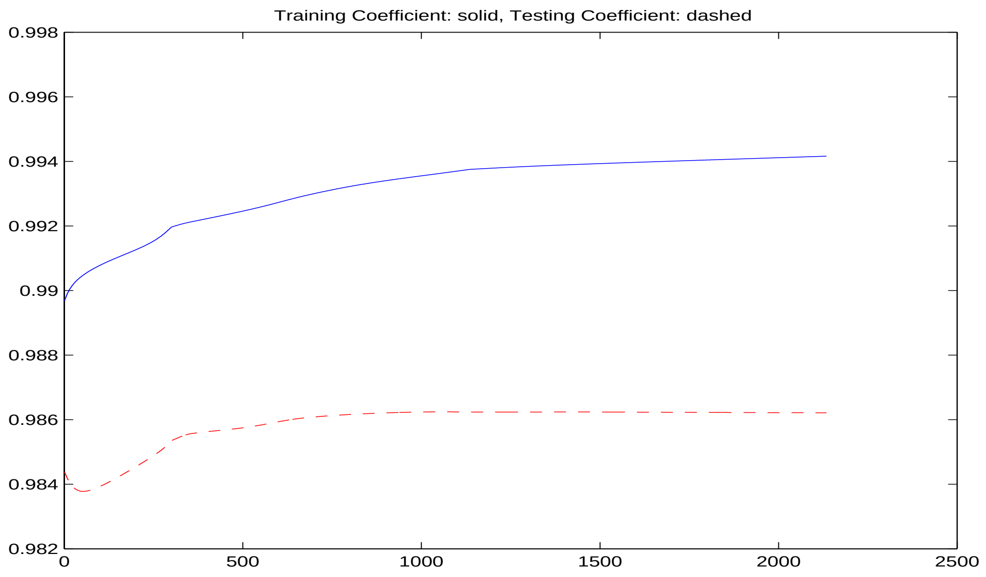


Figure 24: Pearson correlation coefficient for parameter tuning iterations

For the second experiment of batch mode parameter tuning, 20 centers were selected. The results are shown by Figure 25 through Figure 28. Figure 25 provides network training, testing and testing error, the network output and the real value were overlaid in this figure. Figure 26 provides the scatter plot for the network output and actual values. Figure 27 provides the network error for the parameter tuning iterations, the dashed line is for network training error and the solid line is for network testing error. Figure 28 provides the Pearson correlation coefficient for the parameter tuning iterations, the dashed line is for network training data and the solid line is for network testing data. For figures 25 through 28, the learning rate is set to 0.15 for the first 300 iterations, then changed to 0.09 for the rest. Table 6 is a comparison of errors and Pearson correlation coefficients for the network before and after the parameter tuning process.

	Before the tuning process	After the tuning process
Training error	0.065	0.0281
Testing error	0.1483	0.1319
Pearson correlation coefficient (training)	0.9915	0.9964
Pearson correlation coefficient (testing)	0.9827	0.9846

Table 6: Comparison of errors and Pearson correlation coefficients before and after the parameter tuning

We can see from the above table that before the tuning process began, the training error was 0.0650, the testing error was 0.1483, the Pearson correlation coefficient for the training data set was 0.9915, and the Pearson correlation coefficient for the testing data set was 0.9827. After the 1900 parameter tuning iterations, the training error was 0.0281, the testing error was 0.1319, the Pearson correlation coefficient for the training data was 0.9964, and the Pearson correlation coefficient for the testing data was 0.9846.

As we can also see, both the network training and testing error and the Pearson correlation coefficient for the training data set and testing data set has been improved. We can also see the testing error and Pearson correlation coefficient follow closely with the training process.

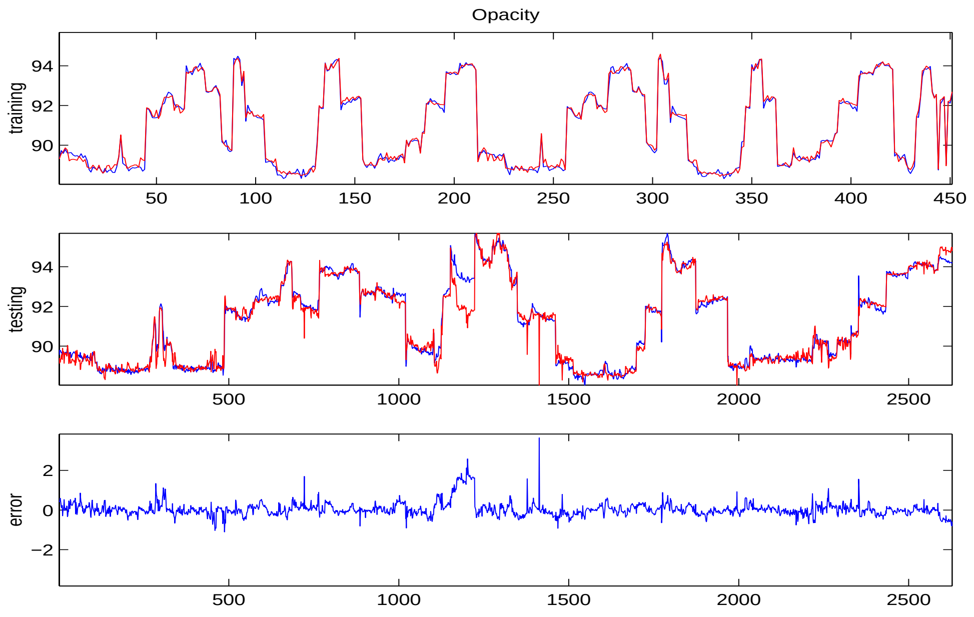


Figure 25: Network training, testing and error

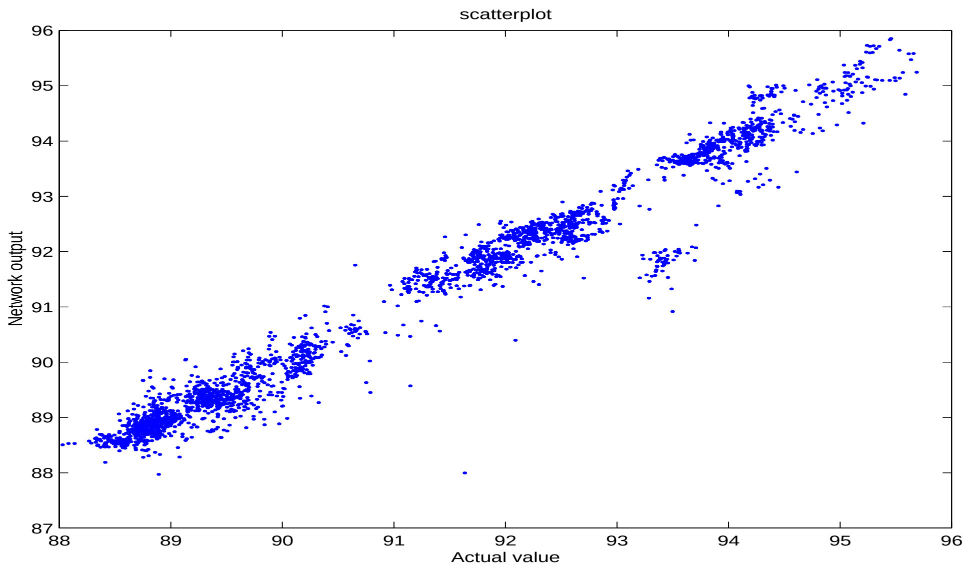


Figure 26: Scatter plot for the actual value and network output

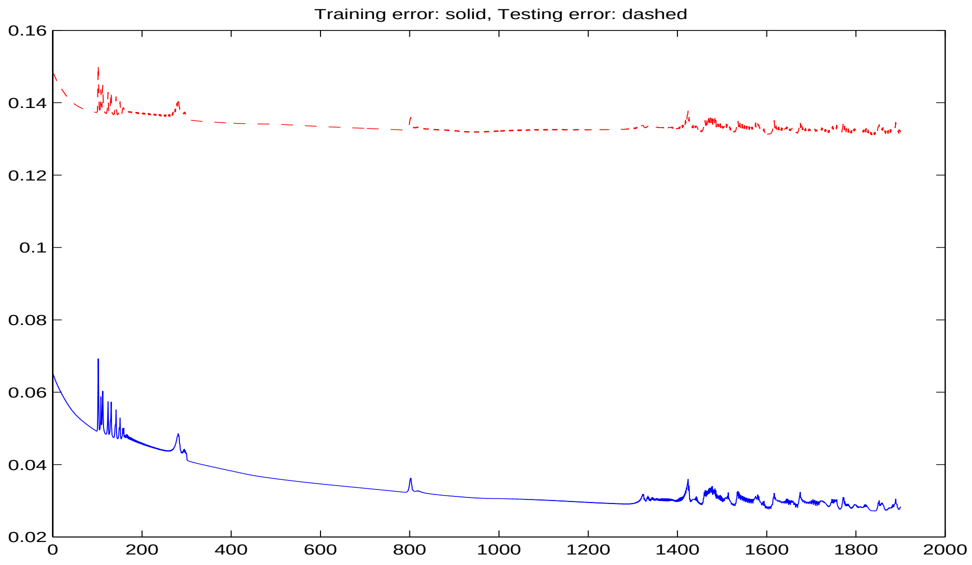


Figure 27: Error for parameter tuning iterations

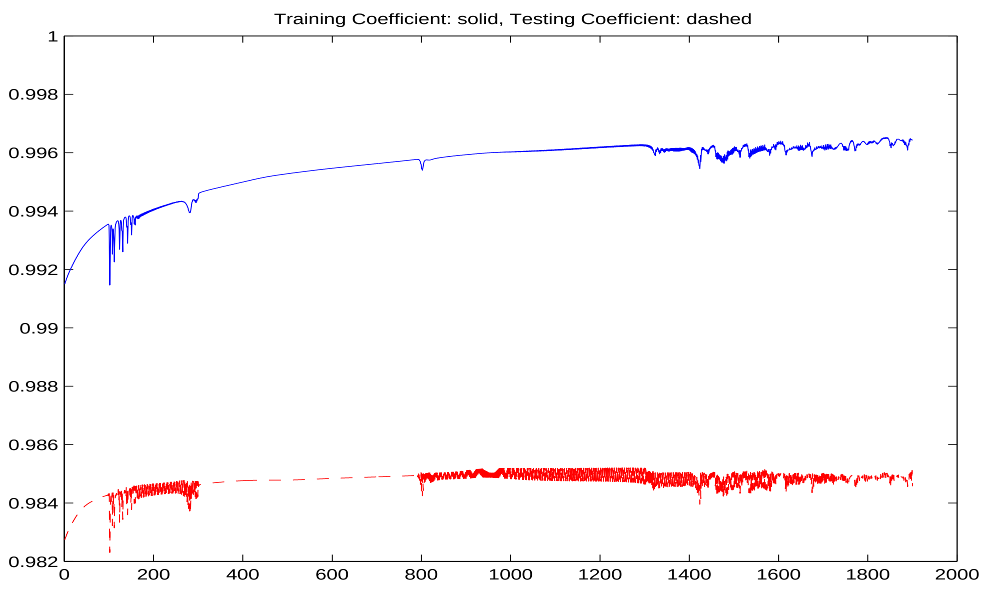


Figure 28: Pearson correlation coefficient for parameter tuning iterations

5.3.2 Selecting The Training Data From 50% of The Data Set

In this experiment the first 50% of the data set was used for selection of the training data. Then, the remaining 50% were used to test the network performance. Note that the same methodology that was used to select the training data in the previous experiment was applied here. The batch mode parameter tuning was used. Figure 29 provides network training, testing and testing error. The network output and the real value were overlaid in this figure. Note that the testing is for the data that hasn't been used for training data selection. Figure 30 shows the scatter plot for the network output and actual values for the testing data set. Figure 31 shows the network error for the parameter tuning iterations, the dashed line is for network training error and the solid line is for network testing error. Figure 32 provides the Pearson correlation coefficients for the parameter tuning iterations, the dashed line is for network training data and the solid line is for network testing data. For figures 29 through 32, the learning rate is set to 0.1 for the parameter tuning iterations. The error is 0.20, and the Pearson correlation coefficient is 0.984 for the network testing.

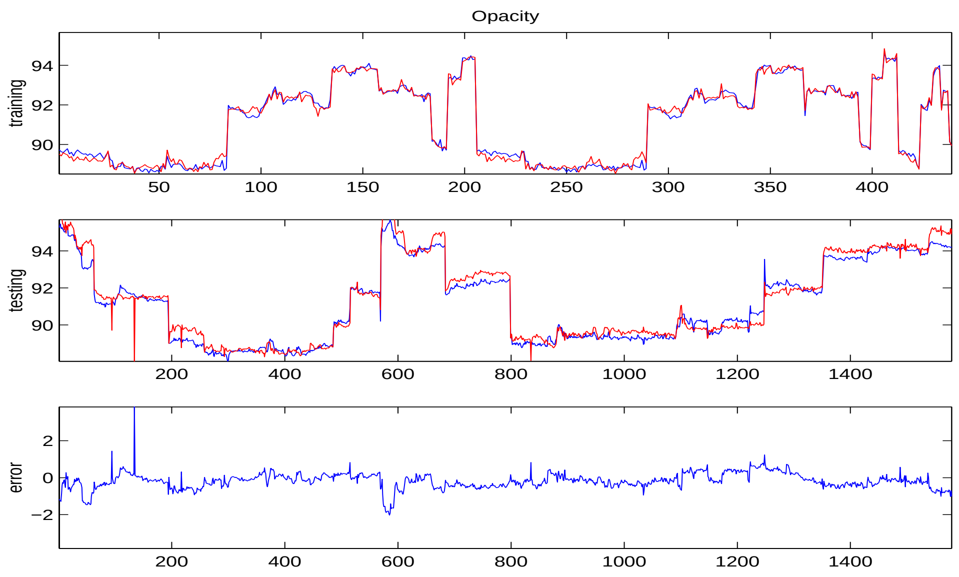


Figure 29: Network training, testing and error

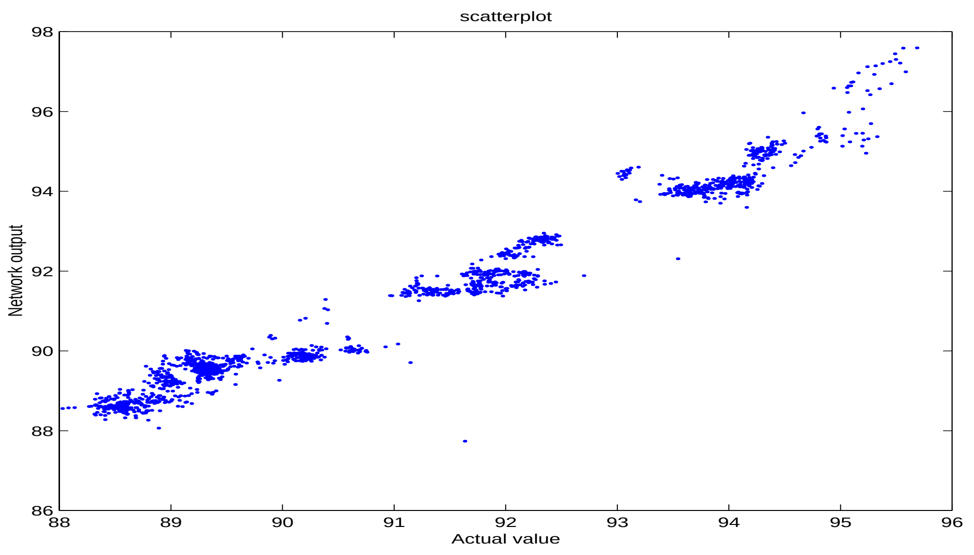


Figure 30: Scatter plot for the actual value and network output

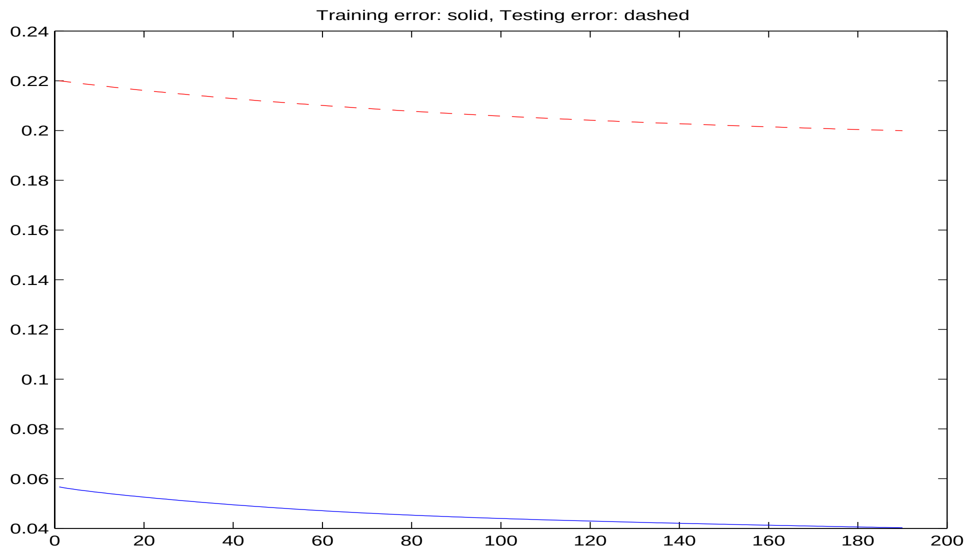


Figure 31: Error for parameter tuning iterations

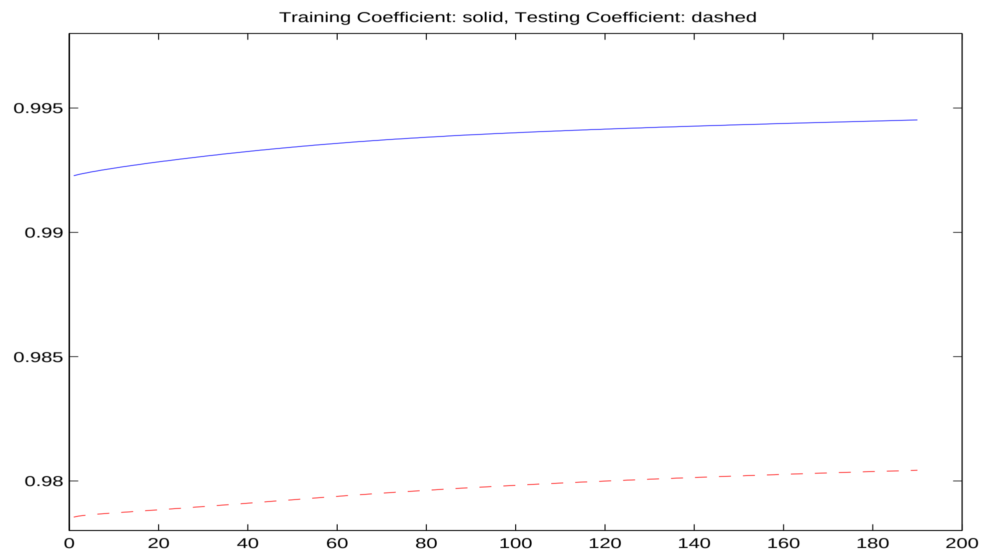


Figure 32: Pearson correlation coefficient for parameter tuning iterations

As can be seen, the network approximation is good over the totally unseen data. This is because the training data selected from the first 50% data contains representative information for the whole data set, which makes the network have good generalization ability.

5.4 Summary of The Results

From the above experiments, we can conclude that the overall performance of the network has improved due to the tuning process. Furthermore, batch mode parameter tuning outperforms instantaneous mode. The change of testing error roughly follows the change of the training error, indicating that the generalization ability provided by the training process is good. It was noted that the batch mode needs much less computation and hence it is faster than the instantaneous mode. The experiments also showed that increasing the number of nodes beyond certain level only slightly improves the network performance. From the above results we can also conclude that the training data selection and RBF node selection schemes work well.

6. CONCLUSION AND POSSIBLE FUTURE WORK

6.1 Conclusion

An RBF neural network model with parameter fine-tuning has been proposed. It has been shown that the proposed technique can improve upon the performance of the conventional RBF techniques. This improvement can especially be observed when the conventional techniques fail to provide a good generalization. For the fine-tuning process, it has been shown that the batch mode is better than the instantaneous mode for the experiments reported in this thesis. The parameter tuning algorithm showed a good stability when the learning rate (step size) was selected appropriately.

For the network model, a sensitivity analysis method using linear correlation was used here to select the important variables. A scheme, which is based on the distribution characteristics of the data, was developed for automatically selecting representative training data. An initial center assignment procedure is applied based on the distribution characteristic analysis. In order to select RBF nodes appropriately, enough data samples should be available to characterize each RBF node. The OLS or FOS algorithm can be used to initialize the network with a set of good values for the parameters of all selected nodes, so that the fine-tuning process using the gradient method will have a good starting point. It was also concluded that the behavior of the RBF network highly depends on how the parameters are chosen and not so much on the number of nodes, when the number of nodes reaches certain level.

6.2 Possible Future Work

This thesis dealt with the qualitative nature of the training data. It would be beneficial to use information theory to define a quantitative measure for expressing how representative the training data is. When using the OLS or FOS algorithm to evaluate the contribution of a node, it’s important to make sure that the functional input space is sufficiently sampled. This is worth further investigation. Due to the number of parameters to be updated, the tuning process could take long time to achieve a good solution if too many nodes are used for the network or the learning rate is not appropriately selected and updated.

Therefore it is very important to appropriately adjust the learning rate in terms of efficiency, stability and convergence. Some momentum and dynamic adjustment of step size could improve the behavior of the gradient method [16]. Note that for the RBF network model discussed in this thesis, the covariance matrix is diagonal. This greatly simplifies the mathematical derivation and notation, especially for high dimensional input spaces. Nevertheless, it may be helpful to investigate the effect of a full covariance matrix.

The proposed algorithm was applied to a mathematical example for validation of the technique. It was also applied to an industrial process. Further test and evaluation with other industrial processes are needed to fully realize the advantage and disadvantages of the proposed technique.

REFERENCES

[1] Broomhead, D. S., and D. Lowe, 1988. “Multivariable functional interpolation and adaptive networks.” Complex Systems 2, pp. 321-355

[2] Poggio, T. and F. Girosi, 1990b. “Networks for approximation and learning.” Proceedings of the IEEE 78, pp. 1481-1497

[3] Renals, S., 1989. “Radial basis function network for speech pattern classification.” Electronics Letters 25, pp. 437-439

[4] Powell, M. J. D., “Radial Basis Functions Approximations to polynomials,” in Proc. 12th Biennial Numerical Analysis Conf. (Dundee), 1987, pp. 223-241

[5] Chen, J. and D. Bruns. “WaveARX neural network development for system identification using a systematic design synthesis, Ind. Eng. Chem. Res., vol. 34, vol. 55, no. 5, pp. 1051-1070, 1992

[6] Moody, J. and C. J. Darken, “Fast learning in networks of locally-tuned processing units,” Neural Computation, vol. 1, pp. 281-294, 1989

[7] Musavi, M. T., W. Ahmed, K. H. Chan, K. B. Faris, and D. M. Hummels, “On the training of radial basis function classifiers”, Neural Networks, vol. 5, pp. 595-603, 1992

[8] Chen, S., S. A. Billings, and W. Luo, “Orthogonal Least Squares methods and their application to non-linear system identification,” Int. J. Contr., vol. 50, no. 5, pp.1873-1896, 1989

[9] Ahmed, W., Fast orthogonal search for training radial basis function neural networks, M.S. Thesis, Department of Electrical and Computer Engineering, University of Maine, 1994

[10] Wellstead, P. E., M. B. Zarrop, “Self-Tuning Systems, Control and Signal Processing”, John Wiley & Sons Ltd. 1991, pp. 85-89

[11] Narendra, K. S., K. Parthasarathy, “Gradient Methods for the Optimization of Dynamical Systems Containing Neural Networks”, IEEE Trans. On Neural Networks, March 1991, pp. 252-262

[12] Golden, R. M., “Mathematical Methods for Neural Network Analysis and Design”, MIT 1996, pp.198-206

[13] Jobson, J. D., “Applied Multivariate Data Analysis Volume I,” Springer-Verlag, 1991

[14] Afifi, A. A. and V. Clark, “Computer-Aided Multivariate Analysis”, Chapman & Hall, 1996, pp.382

[15] Miranda, Jon, Prediction of Wood Pulp Digester Level Using Artificial Neural Networks, M.S. Thesis, Department of Electrical and Computer Engineering, University of Maine, 1997

[16] Haykin, Simon, Neural Networks, Prentice-Hall, Inc. 1994

[17] Chen, S., Billings, S. A., and P. M. Grant. Recursive hybrid algorithm for nonlinear system identification using radial basis function networks, Int. J. Control, vol. 55, no. 5, pp. 1051-1070, 1992

[18] Rao, Ming, Qijun Xia and Yiqun Ying, Modeling and advanced control for process industries, applications to paper making processes, Springer-Verlag London Limited 1994

BIOGRAPHY OF THE AUTHOR

Junxu Li was born in YongKang, Zhejiang Province, the People's Republic of China, on September 22, 1973. After graduation from YongKang 1st High School in 1991, he enrolled at the Zhejiang University, Hangzhou, China, and graduated in 1996 with a Bachelor's degree. He began his graduate study at the University of Maine in January, 1997.

Junxu is a candidate for the Master of Science degree in Computer Engineering from the University of Maine in May, 1999.