A Thesis

entitled

A Comparison of Various Interpolation Techniques for Modeling and Estimation of Radon Concentrations in Ohio

by

Jayaram Gummadi

Submitted to the Graduate Faculty as partial fulfillment of the requirements for the

Master of Science Degree in Engineering

_____

Dr. William Acosta, Committee Chair

_____

Dr. Vijay Devabhaktuni, Co-Committee Chair

_____

Dr. Ashok Kumar, Committee Member

_____

Dr. Rob Green, Committee Member

_____

Dr. Patricia R. Komuniecki, Dean
College of Graduate Studies

The University of Toledo

December 2013

Copyright 2013, Jayaram Gummadi

This document is copyrighted material. Under copyright law, no parts of this document may be reproduced without the expressed permission of the author.

An Abstract of

A Comparison of Various Interpolation Techniques for Modeling and Estimation of Radon Concentrations in Ohio

by

Jayaram Gummadi

Submitted to the Graduate Faculty as partial fulfillment of the requirements for the Master of Science Degree in Engineering

The University of Toledo
December 2013

Radon-222 and its parent Radium-226 are naturally occurring radioactive decay products of Uranium-238. The US Environmental Protection Agency (USEPA) attributes about 10 percent of lung cancer cases that is 'around 21,000 deaths per year' in the United States, caused due to indoor radon. The USEPA has categorized Ohio as a Zone 1 state (i.e. the average indoor radon screening level greater than 4 picocuries per liter). In order to implement preventive measures, it is necessary to know radon concentration levels in all the zip codes of a geographic area. However, it is not possible to survey all the zip codes, owing to reasons such as inapproachability. In such places where radon data are unavailable, several interpolation techniques are used to estimate the radon concentrations. This thesis presents a comparison between recently developed interpolation techniques to new techniques such as Support Vector Regression (SVR), and Random Forest Regression (RFR). Recently developed interpolation techniques include Artificial Neural Network (ANN), Knowledge Based Neural Networks (KBNN), Correction-Based Artificial Neural Networks (CBNN) and the conventional interpolation techniques such as Kriging, Local Polynomial Interpolation (LPI), Global Polynomial Interpolation (GPI) and Radial Basis Function (RBF) using the K-fold cross validation method.

To my family for their love, endless support, and encouragement

Acknowledgements

I wish to express my deepest gratitude to my advisors Dr. Vijay Devabhaktuni and Dr. William Acosta for their continued support, guidance and encouragement, without which it would not have been possible to succeed in my research. I would also like to thank Dr. Ashok Kumar for proving the radon data from Ohio Radon Information System and for giving valuable inputs during my research. The ODH/United States Environmental Protection Agency and Ohio Air Quality Development Authority have been supporting indoor radon data collection for the past 27 years under the direction of Dr. Ashok Kumar. The financial support from the EECS Department in the form of a graduate/research assistantship is gratefully acknowledged. I would like to thank my lab mates and friends Deepak Bhatt, and Srujana Adusumilli for their constant support to complete my research.

I would like to thank my roommates Teja Bandaru, Prem kumar Bodiga, Vishwanath Ullagaddi and my friends Ahmad Yazdan Javaid, Sandeep Patil, Srimanth Gadde, Anuroopa Vanteru, Kavya Vittala, Snigdha Buddala, Shanmuka Harish Chalamuri and Pavan Mantripragada for their support and motivation throughout my masters, and also for making my stay in Toledo a memorable one. Above all, I would like to thank my parents and my brother, who with their love and encouragement have made all this possible.

Contents

List of Tables

List of Figures

List of Abbreviations

ANN..... Artificial Neural Networks

CBNN ..... Correction-Based Artificial Neural Networks
CV..... Cross-Validation

Fa_2..... Factor of Two
FB ..... Fractional Bias

KBNN ..... Knowledge Based Neural Networks

LPI ..... Local Polynomial Interpolation

MAE ..... Mean Average Error

NMSE ..... Normalized Mean Square Error

ODH..... Ohio Department of Health

pCi/L ..... Pico-Curie per liter of air
PKI..... Prior Knowledge Input

RBF..... Radial Basis Function
RFR..... Random Forest Regression
RMSE ..... Root Mean Square Error

SDM..... Source Difference Method
SM ..... Space Mapping
SMNN..... Space Mapped Neural Networks
SVR ..... Support Vector Regression

USEPA..... United States Environmental Protection Agency

Chapter 1

Introduction

1.1 Problem Statement

Radon is a colorless, odorless, radioactive gas formed by natural decay of uranium in soil, rock and water. Radon migrates through soil and enters home through basement walls, cracks in concrete floors or slabs, and openings around utility accesses. Radon causes an estimated 21,000 lung cancer deaths in the United States every year [1]. It is the second leading cause of lung cancer after active smoking and the leading cause among nonsmokers. Radon concentration is measured using radon-monitoring devices in the units of Pico-Curie per liter of air (pCi/L). The USEPA, based on indoor radon studies, suggests that homes or schools exceeding the radon concentration of 4pCi/L pose health risk, and recommend people to take preventive measures for reducing the radon level [2].

Research carried out by the USEPA and others has shown that it is possible to bring down the radon levels in buildings [3]. Since 1989, in order to help mitigators to identify the buildings with high radon concentration level and implement a cost effective radon mitigation plan, the Ohio Department of Health (ODH) with radon testing laboratories,

local health departments, and university researchers has collected radon information for more than 200,000 homes and schools across Ohio [4-7]. The data collected by these organizations are being managed by different database management systems [8-13]. Most of the Ohio indoor radon data are contained in the Ohio Radon Information System (ORIS) developed by The University of Toledo in association with Ohio Air Quality Development Authority and ODH [11].

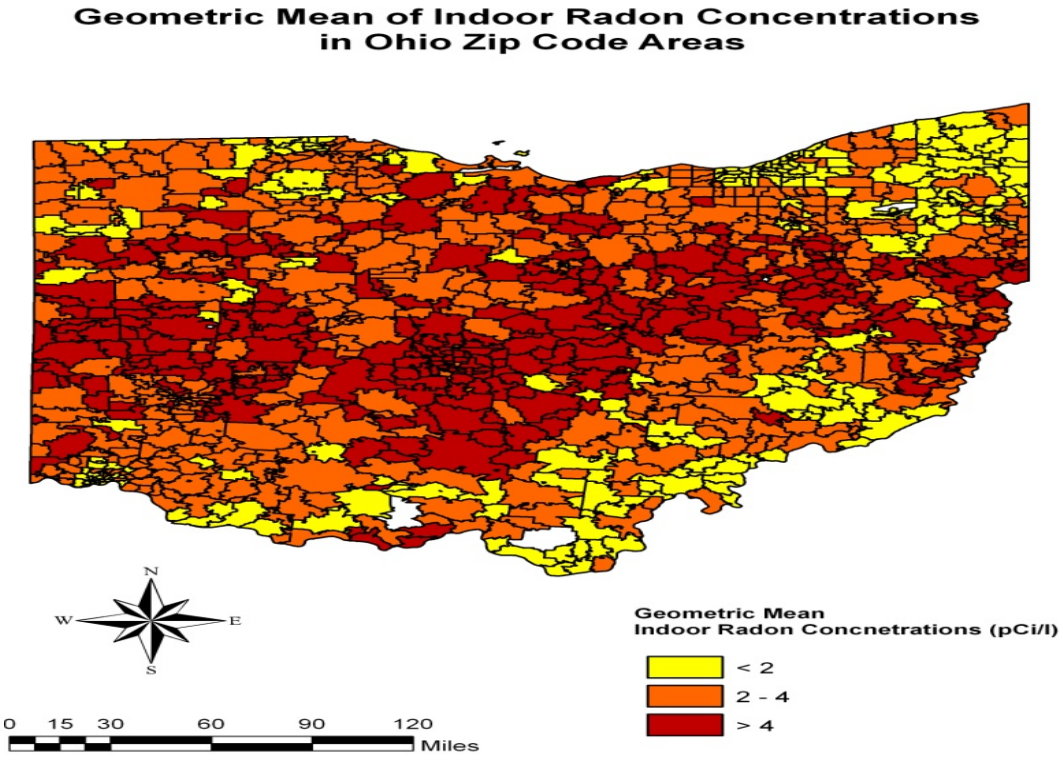

Figure 1-1: Map Depicting the Indoor Radon Concentrations in Ohio

Radon concentration data are available for 1261 zip codes out of the 1417 zip codes across Ohio, but it is not possible to collect the information for all the zip codes in the State of Ohio, owing to several reasons like inapproachability. Places where radon data cannot be collected, there is a growing need to estimate the radon concentration of these zip codes using various interpolation techniques. Figure 1-1 shows the indoor radon concentrations in Ohio [14].

1.2 Research Approach

In the process of developing a cost-effective and reliable radon concentration prediction model, several research papers have been published by researchers at The University of Toledo. Initial research was focused on the conventional interpolation techniques such as Kriging, Local Polynomial Interpolation (LPI), Global Polynomial Interpolation (GPI) and Radial Basis Function (RBF) to estimate the radon concentrations [15-21]. Since these conventional interpolation techniques are highly data-specific and are based on complex mathematical models; Artificial Neural Networks (ANN) such as 3-layer Multi-Mayer Perceptron (MLP3) was introduced for estimation of radon [16]. However, MLP3 networks have their own drawbacks such as dependency on the adequacy of data and are black-box models whose prediction accuracy is only dependent on training data. Later, in order to overcome the drawbacks of ANNs, more advanced ANNs such as Knowledge Based Neural Networks (KBNNs) and Correction-Based Neural Networks were introduced for estimation of radon. KBNN includes models such as Prior Knowledge Input (PKI), Source Difference Method (SDM) and Spatial Mapped Neural Networks (SMNN) which enhances the generalization ability and extrapolation capability over ANN using the existing knowledge [17]. In this thesis, the uranium concentration specific to zip code is the existing knowledge which is obtained from the maps published by the Ohio Department of Natural Resources. Correction-Based Artificial Neural Networks (CBNN) was introduced as an improvement to KBNN overcoming their structural complexity using a sensitivity-based approach [18]. However, the present research was limited to split-sample validation on neural networks and the

conventional interpolation techniques. Due to the need of improving the generalization ability, a Cross-Validation (CV) technique is implemented over split-sample validation. Also, this research approach introduces the applicability of two regression techniques namely Random Forest Regression (RFR) and Support Vector Regression (SVR) for modeling and estimating the radon concentrations. Comparisons between the existing conventional interpolation techniques, neural network approaches and the newly introduced SVR and RFR techniques are discussed in this thesis. The techniques are validated using 7-fold and 10-fold cross-validation data.

1.3 Organization of Thesis

The remaining document is organized as follows. Chapter 2 discusses the literature review. Chapter 3 gives an overview of conventional interpolation techniques, data preparation and the performance measures to compare the different interpolation techniques. Chapter 4 discusses different neural network approaches. Chapter 5 discusses the use of newly introduced regression techniques, SVR and RFR. Finally, chapter 6 concludes the thesis and discusses future work.

Chapter 2

Literature Review

2.1 Role of Interpolation

Interpolation plays a vital role in estimating values where no actual value can be measured. In order to estimate missing values or make interpretations, scientists, and environmental managers would need spatially continuous data. Thus, to generate spatially continuous data, the values of required attribute at unobserved locations need to be estimated. In such cases, where attribute values are not available spatial interpolation methods are used [22]. Assumptions followed while using spatial interpolation methods are attribute data is continuous over space where estimation at any unobserved location is within the data boundary and it is spatially dependent.

In this thesis, data provided by ODH is used for interpolating a method and creating radon concentration maps across Ohio. This work is further used for quantitative research like reducing the cancer, which is caused by inhaling the radon. This indicates the importance of identifying an accurate interpolation method for predicting the missing radon concentrations.

Spatial interpolation methods are widely used for employing geostatistics in various

fields such as environmental sciences, civil engineering, soil sciences and mathematics or statistical analysis. The working principle, advantages and disadvantages of various interpolation techniques are discussed in Table 2.1 [16].

Table 2.1: Conventional Interpolation Techniques commonly used in Environmental Science

Interpolation Techniques	**Working Principle**	**Advantages**	**Disadvantages**
Kriging	Linear combination of sample points including spatial autocorrelation	Fitting the data into a polynomial function, no edge-effects, best linear based spatial predictor	Unable to solve problems that are non- stationary, highly complex and requires sophisticated programming
Trend Surface analysis	Separates data based on regional and local variations	Broader trends are removed prior to further analysis	Spatial autocorrelation results in edge effects
Inverse Distance Weighting	Linear combination of data points, inversely weighted as per distance	Easy to use and works well with all kinds of data	Weights are not affected by spatial arrangement of samples
Polynomial Regression	Variables of interest are fitted into linear combination	Easy to model	Poor prediction of data points outside the range

GPI	Determines a polynomial function of whole surface and capture coarse-scale patterns	Computationally easy to use	If the data complexity increases the prediction accuracy decreases exponentially
LPI	Similar to GPI but determines the polynomial functions in individual neighborhoods	Very advantageous to predict data with local variations	Unlike GPI, the global trends are missing
RBF	Linear combination of different basis functions, fits the data reducing the total curvature of surface	Doesn't require huge amount of data	Requires good input space, doesn't work fine with data having local variations and not suitable for extrapolation

2.2 Prior Art

Conventional interpolation techniques such as Kriging, GPI, LPI and RBF have already been employed to estimate the missing radon concentrations in Ohio, limited to split sample validation [15]. These interpolation techniques are further discussed below.

Kriging is a stochastic interpolation method which is based on spatial autocorrelation of the data. Spatial autocorrelation determines the statistical relationship between values where sample observations are available. Thus obtaining the relationship, it is used to predict the attribute values at unsampled locations. Apart from predicting the attribute

values, this technique also estimates the accuracy in prediction including prediction standard error, probability, and standard error of indicators. Kriging has been previously used for mapping the soil parameters [23], temperature [24] and estimation of rainfall [25].

GPI is an interpolation method which is based on determining a polynomial function to input sample points. The prediction of a GPI method is based on polynomial function and if it is of higher order then it results in a bad prediction. The advantage of using this interpolation technique is determining a global model that evaluates a general trend of the surface. However, this global model cannot deal with the extreme high and low data values (outliers).

As GPI fits a polynomial function of the whole surface, LPI is based on polynomial functions for individual neighborhoods. LPI is advantageous to predict environmental data due to its accountability to local variations. Especially, in the case of environmental sciences, the attribute usually has local variations in addition to global variations. As LPI is based on neighborhood distance it can effectively predict the local variations in data. This technique works effectively with local variations in data but the global variations are missing which is a drawback. LPI and GPI techniques have been previously employed for estimating climate data [26] and agricultural water use [27].

Unlike Kriging, GPI and LPI, the RBF is not based on a single variant or multi-variant polynomial function; it is a linear combination of different basis functions. The five different basis functions mentioned are multi-quadric function, inverse multi-quadric function, spline with tension, thin-plate spline and completely regularized spline. Conceptually, RBF is described as fitting a rubber membrane with the given input values

in addition to reducing the total curvature of surface. RBF predicts the attribute value independent of the direction instead of using a polynomial function. The disadvantage of this technique is it predicts the attribute values which are minimum or maximum in range, which is not practical in all situations. However, having surface values within a short range, this technique is not suitable for interpolation. RBF has been previously employed in estimating water content of natural gases [28] and in analyzing air quality measurements [29].

2.3 Review of Neural Network Techniques

In computer aided modeling applications, ANNs play a vital role in estimating the missing values due to their efficient generalization capability. Unlike conventional interpolation techniques, which are based on complex mathematical functions, ANNs proved to be efficient. It is worth mentioning that the efficiency of a neural network model is directly proportional to number of training samples used for training the model. The disadvantage of ANN model is it lacks extrapolation capability. However, ANNs are being widely used for data classification and regression in many fields.

Recently, a review over multi-layer perceptron for estimating environmental variables in atmospheric sciences has been performed [30]. ANNs have been employed for data classification to recognize the hand written zip-code provided by the U.S. Postal Service [31]. ANNs have been extensively used for predicting air quality [32, 33] and water quality [34, 35] measurements using related data from past few years. ANNs have also been employed for predicting the ozone concentration levels [36, 37]. In UK, ANNs have

been used for prediction of surface ozone concentrations [38].

KBNNs are the more advanced ANNs that enhance the generalization and extrapolation capabilities of a neural network model [39]. The generalization capability allows the neural network model to predict the unknown data within the range, while the extrapolation capability allows predicting the data beyond the training range.

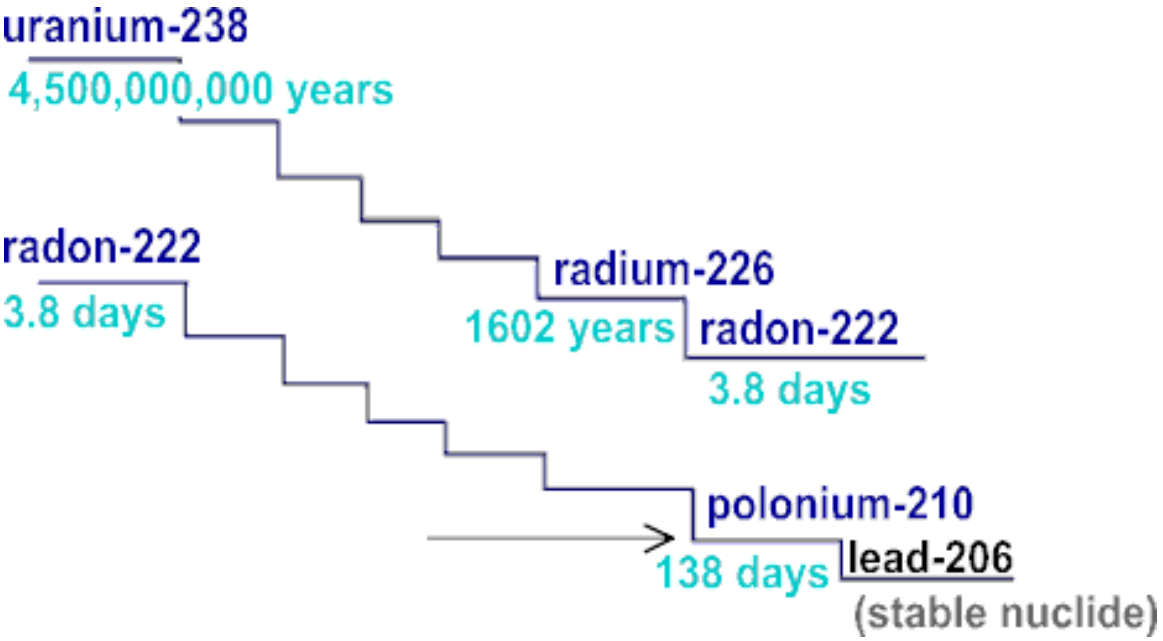

Figure 2-1: Uranium-238 Decay chain and half lives

The knowledge input considered in this application is uranium concentration as radon is a decay product of uranium. Figure 2-1 represents the uranium decay chain. In Harrell et al. [6], it was mentioned that constructions built above uranium-bearing rocks or sediments may have higher indoor radon levels. Figure 2-2 represents map of Ohio showing uranium concentration.

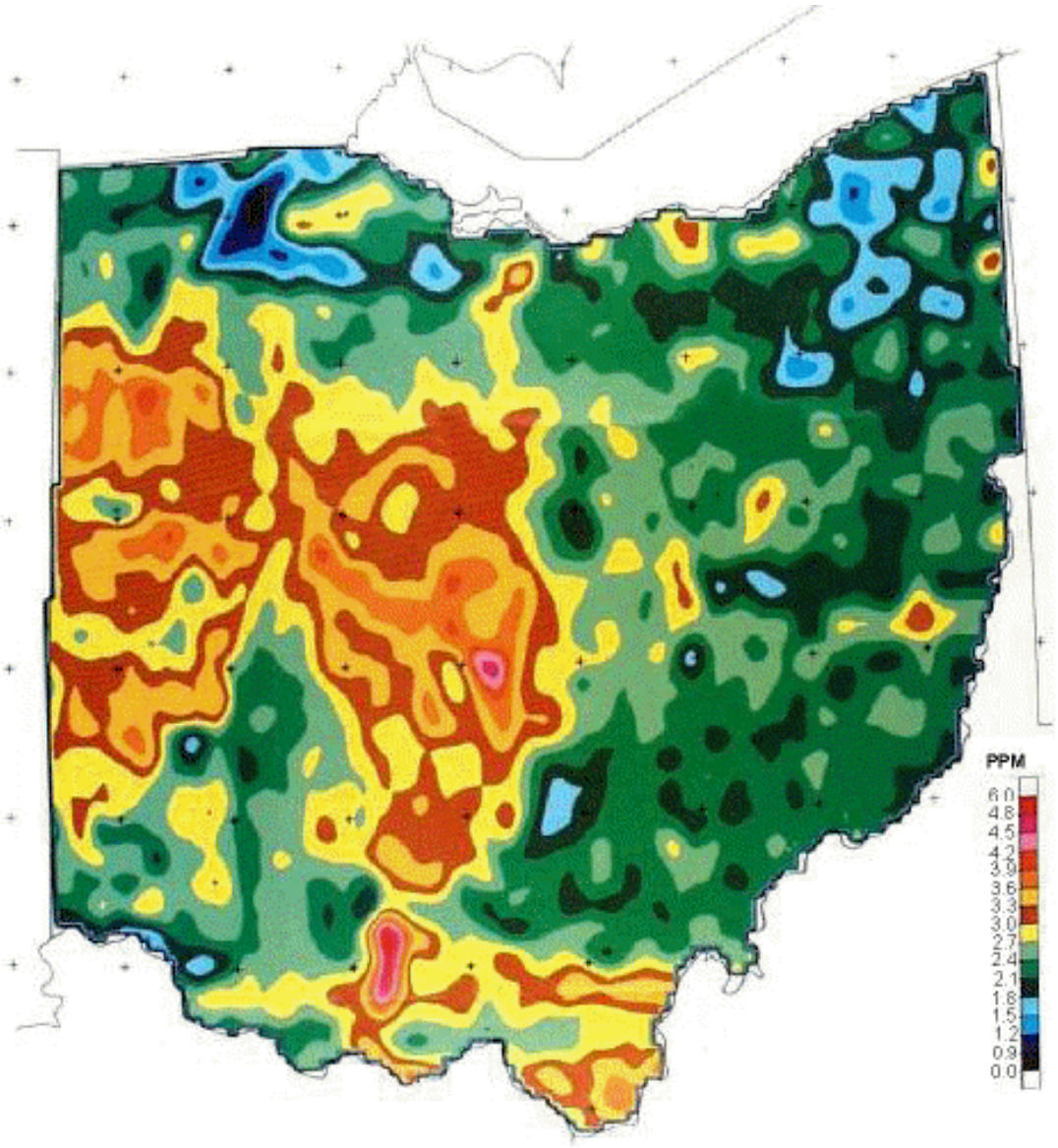

Figure 2-2: Radiometric map of Ohio showing uranium concentration in soil

The different type of KBNN approaches are Prior Knowledge Input (PKI), Source Difference Method (SDM), and Space Mapped Neural Networks (SMNN).

The PKI method proposed in [40] establishes a functional relationship between the inputs and outputs along with an additional knowledge input. A knowledge input could be a source variable to the attribute for prediction. A general representation of PKI

method is shown in Figure 2-3.

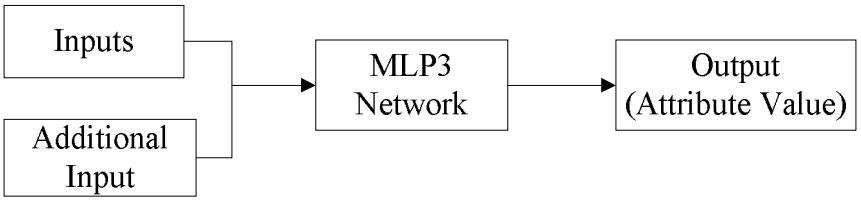

Figure 2-3: General Representation of PKI Method

The SDM method proposed in [41] is a combination of two different MLP3 models. The structure of SDM method is described to be a combination of coarse model and difference model, where the coarse model is trained and validated using the original inputs while the attribute value is the desired output. The prediction of this MLP3 model is used to calculate the estimation error between the predicted and original value. Using estimation error as the desired output, the difference model is trained and validated. Final output is obtained by adding the predictions from both the models. A general representation of SDM method is shown in Figure 2-4.

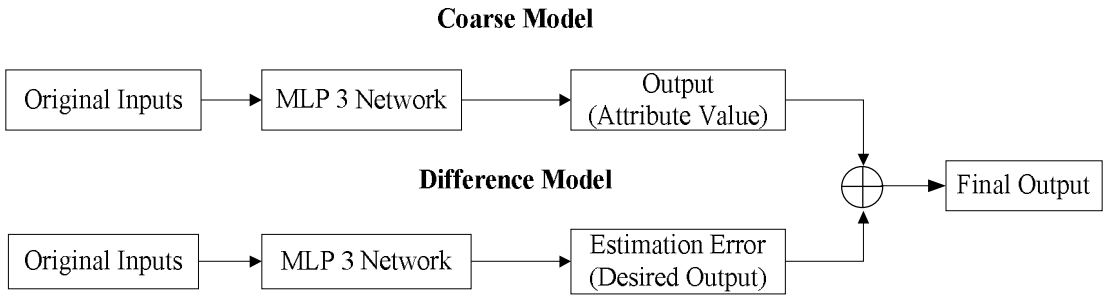

Figure 2-4: General Representation of SDM Method

The SMNN method proposed in Bandler et al. [42] is used to estimate the attribute value with intermediate inputs. The two models used in SMNN method are space mapping model and coarse model. The space mapping model uses a space-mapping (SM)

technique [40] that maps the fine model input space into a coarse model input space. Finally, the coarse model input spaces are used to estimate the attribute value. A general representation of SMNN method is shown in Figure 2-5.

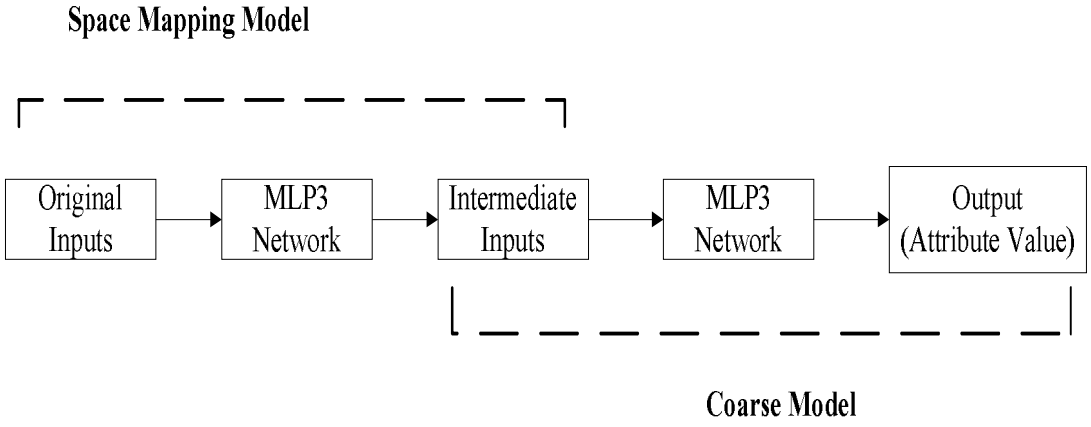

Figure 2-5: General Representation of SMNN Method

Recently, Akkala et al. (2010) has employed ANNs [19] and KBNNs [20] to estimate the missing radon concentrations in Ohio and concluded that these models have better prediction accuracy over conventional interpolation techniques.

Correction based ANN's use different kind of approaches such as Newton's method and steepest-descent method for estimating the attribute values. A sensitivity-based approach that has the advantages of Newton's method and steepest-descent method has been employed to model RF/microwave components [44]. Recently, this sensitivity based correction ANN approach has also been employed to estimate the missing radon concentrations in Ohio [18].

Chapter 3

Conventional Interpolation Techniques

3.1 Data Preparation

Initially, the radon concentration data available for 1261 zip codes are partitioned into subsets for cross-validation. In this thesis, the K-fold cross-validation technique has been employed where the available data is partitioned into k subsamples. The technique used in K-fold cross-validation is to use every k^{th} dataset to validate the model, while the remaining $k-1$ datasets are used to train the model. The advantage of using this technique is that all the samples are used for training and validation, and each sample is used exactly once in the validation. The process of partitioning the data is done by using a MATLAB function [45] given as

$$f = cvpartition(n,'kfold',k) \tag{3.1}$$

In eq. (3.1), n is the total number of samples, $kfold$ is the technique used for partitioning the data and k is the size of subsets.

Further, the training and validation data sets are separated using two methods of $cvpartition$ class given as

$$t = training(f, i)$$
$$v = test(f, i) \tag{3.2}$$

In eq. (3.2), t and v are vectors of training and validation indices respectively, $training$ and $test$ are methods in $cvpartition$ class, f is an object of $cvpartition$ class and i is the index of respective sample.

In this thesis, 7-fold cross-validation ($k = 7$ i.e., 7 training and validation data sets respectively) and 10-fold cross-validation ($k = 10$ i.e., 10 training and validation data sets respectively) have been used.

To compare all the methods conventional interpolation techniques, ANN, KBNN, correction-based ANN, SVR and RFR on a fair basis, we used the same data sets for training and validation of these models. The final performance measure in K-fold cross-validation is taken as the average of performance measures of individual k datasets.

3.2 Comparative Performance Measures for Evaluating Interpolation Techniques

It is necessary to assess the performance of an interpolation technique to determine if one technique is better than the other. The performance measures are used to estimate the difference between the predicted values and the actual values of an attribute. Recently, Kumar et al. [15. 46-49] have given important parameters to assess the performance of air-quality models. The different type of performance measures mentioned are Mean Absolute Error (MAE), Factor of Two (Fa_2), Root Mean Square Error (RMSE), Fractional Bias (FB), and Normalized Mean Square Error which are described further in

this section. Apart from these commonly used performance measures, E_{avg} is the commonly used validation error to measure the performance of an interpolation technique that is discussed in Chapter 4.

3.2.1 Mean Absolute Error

The MAE is used to measure the average magnitude of errors in a set of predictions and is given as

$$MAE = \frac{\sum_{i=1}^{N_v} |obs(i) - pred(i)|}{N_v} \tag{3.2.1}$$

In eq. (3.2.1), $pred(i)$ is the predicted value from the model for every i^{th} sample in validation set, $obs(i)$ is the actual or observed value of an attribute and N_v is the total number of samples. The optimal value of MAE is zero.

3.2.2 Factor of Two

The Fa2 is used to define the percentage of predicted value over observed value and is given as

$$Fa_2 = \frac{obs(i)}{pred(i)} \tag{3.2.2}$$

The ratio of predicted value to actual value lies between 0.5 and 2.0. The optimal value of Fa_2 is one.

3.2.3 Root Mean Square Error

The RMSE is a strictly proper scoring rule that is used to calculate the average magnitude of error and is given as

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N_v} (obs(i) - pred(i))^2}{N_v}} \tag{3.2.3}$$

The performance measure is ideally used to measure the magnitude of extreme errors

in prediction. The optimal value of $RMSE$ is zero.

3.2.4 Fractional Bias

The fractional bias or normalized bias is used to calculate the fraction of mean concentrations and is given as

$$FB = \frac{\sum_{i=1}^{N_v}(obs(i) - pred(i))}{\frac{1}{2}\sum_{i=1}^{N_v}(obs(i) + pred(i)} \tag{3.2.4}$$

The fractional bias lies between -2 and +2. The optimal value of FB is zero.

3.2.5 Normalized Mean Square Error

The NMSE is used to assess the scatter in the complete data set and is given as

$$NMSE = \frac{\sum_{i=1}^{N_v}(obs(i) - pred(i))^2}{\frac{1}{N_v}(\sum_{i=1}^{N_v} pred(i) * \sum_{i=1}^{N_v} obs(i))} \tag{3.2.5}$$

Normalization is used to assure that this performance measure is not biased in over prediction or under prediction of a model. $NMSE$ values are inversely proportional to the performance of a model (i.e., the smaller $NMSE$ value indicates better performance of model). The optimal value of $NMSE$ is zero.

3.3 Advantages and Disadvantages of Conventional Interpolation Techniques

The conventional interpolation techniques implemented in this thesis are Kriging, LPI, GPI, and RBF. However, there are certain advantages and disadvantages of these conventional interpolation techniques [49]. Kriging is the best linear unbiased spatial predictor but doesn't deal with the data points in real world data sets. LPI is having the

advantage of interpolating local or short-range variations but cannot interpolate the long-range variations and misses the global trends. GPI is computationally easy but with the increasing complexity of the model, the estimation error increases exponentially. RBF can be used for modeling with few samples but it is not suitable for extrapolation.

3.4 Results and Discussions

In order to utilize the conventional interpolation techniques such as Kriging, LPI, GPI and RBF, ArcGIS analyst software [15, 49] is used for estimating the radon concentrations. The performance measures of these individual conventional interpolation techniques are compared in Table 3.1.

Table 3.1: Performance Measures of Conventional Interpolation Techniques usin different Cross-Validation Methods

Conventional Interpolation Technique	**CV Type**	**E_{avg}**	**MAE**	**Fa_2**	**RMSE**	**FB**	**NMSE**
Kriging	7-fold CV	4.16	1.60	0.7803	2.97	0.0293	0.765
	10-fold CV	**4.12**	**1.59**	**0.7922**	2.94	0.0280	0.760
Local Polynomial Interpolation	7-fold CV	4.25	1.63	0.7708	2.97	0.0240	0.760
	10-fold CV	4.22	1.62	0.7645	**2.93**	0.0205	**0.745**
Global Polynomial Interpolation	7-fold CV	5.13	1.97	0.6558	3.18	0.0107	0.854
	10-fold CV	5.11	1.97	0.6550	3.16	**0.0083**	0.843
Radial Basis Function	7-fold CV	4.40	1.69	0.7779	3.09	-0.0027	0.804
	10-fold CV	4.65	1.67	0.7756	3.05	-0.0009	0.781

From Table 3.1, it can be clearly stated that using any type of cross-validation

technique, these conventional interpolation techniques did not show much difference in the errors. It can be observed that Kriging using 10-fold cross-validation (CV) type has shown better performance over other interpolation techniques in case of evaluating E_{avg}, MAE and Fa_2. The optimal value for $E_{avg,}$ MAE, RMSE, FB and NMSE is zero whereas for Fa_2 it is one. LPI technique using 10-fold CV type is having the least magnitude error (RMSE and NMSE) as shown in Table 3.1. The optimal value of RMSE, NMSE, and FB is zero. Though GPI and RBF have least values in FB, when these techniques are compared with remaining performance measures, it can be concluded that GPI and RBF are not suitable to interpolate data having short-range variations. Conventional interpolation techniques have the disadvantage of using complex mathematical functions. However, neural network techniques can be used to define the functional relationship between the inputs and outputs. The different types of neural network approaches are discussed in Chapter 4.

Chapter 4

Neural Network Approaches

4.1 Artificial Neural Network Approach

ANNs is a multilayer feed forward network. The most widely used ANN is 3-layer perceptron network often called MLP3. The MLP3 network architecture is shown in Figure 4.1. In this thesis, ANN is implemented using k-fold cross-validation technique, as the existing implementation of ANNs to estimate the missing radon concentrations is limited to split-sample validation technique [16].

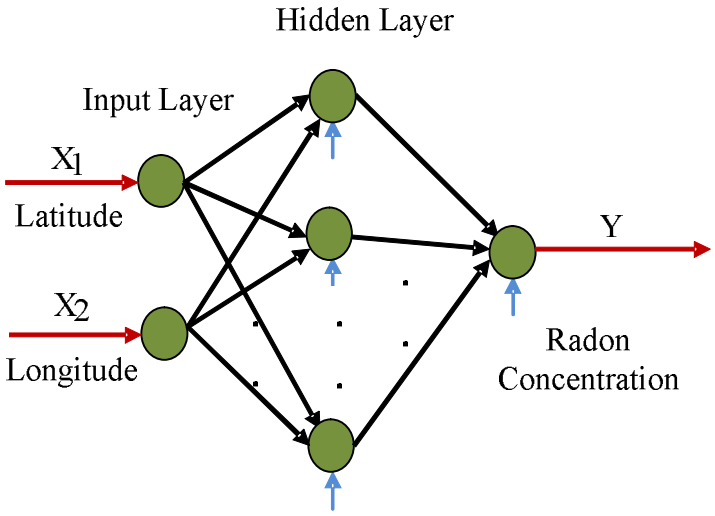

Figure 4-1: MLP3 Network Architecture to Estimate Radon Concentrations (ANN)

The main advantage of ANNs over conventional interpolation techniques is defining

the functional relationship between the inputs and outputs of a model rather than estimating the output using complex mathematical models [50]. The neural network approaches are trained and validated using a CAD tool named *Neuromodeler* [51].

4.1.1 Methodology

In Figure 4-1, let x and y be the vectors that contain model inputs and outputs respectively, then the relationship between these vectors can be defined as

$$y = f(x) \tag{4.1}$$

In eq. (4.1), f represents the functional relationship between x and y, which is derived from the training process using a set of sample pairs given by

$$\{(x_p, d_p), p = 1, \dots, N\} \tag{4.2}$$

In eq. (4.2), d_p represents the desired output corresponding to p training input vector x_p, N is the number of data samples available for training, and p is simply a sample index.

During the training process, a model parameter w, called weights, in neural network is iteratively adjusted to minimize the error between neural model output y and desired output d, given by

$$E(w) = \frac{1}{2}\sum_{p=1}^{N}\sum_{q=1}^{m}(y_{pq}(x_p, w) - d_{pq})^2 \tag{4.3}$$

In eq. (4.3), $y_{pq}(x_p, w)$ is the q^{th} output of the neural network with input x_p and the model parameter w is constantly updated, given by

$$w_{next} = w_{now} + \eta g \tag{4.4}$$

In eq. (4.4), η is a positive step-size and g is the update direction. From the above

equation, it states that w_{next} is the adjusted weight from w_{now} along the update direction g.

The output at every j^{th} hidden neuron is calculated using the sigmoid activation function and is given by

$$z_j = \frac{1}{1 + \exp\left(-\left(\sum_{i=1}^{n} u_{ij} * x_i + u_{0j}\right)\right)} \tag{4.5}$$

In eq. (4.5), u_{0j} is the bias parameter for j^{th} hidden neuron and u_{ij} is the weight between the i^{th} input neuron and the j^{th} hidden neuron of x_i input that is latitude or longitude in present case.

Finally, the radon concentration, y, is calculated as

$$y = \sum_{j=1}^{h} \left(z_j * v_j\right) + v_0 \tag{4.6}$$

In eq. (4.6), v_j is the weight between the j^{th} hidden neuron and the output neuron, h is the total number of hidden neurons and v_0 is the bias parameter.

The validation error of Artificial Neural Network (ANN) i.e., E_{avg} is given as

$$E_{avg} = \frac{1}{N_v} \sum_{i=1}^{N_v} \left| \frac{f_{ann}(x_i, w) - y_i}{y_{max} - y_{min}} \right| \tag{4.7}$$

In eq. (4.7), N_v is the number of training samples, y_i is the actual output and $f_{ann}(x_i, w)$ is the predicted output to the input vector x_i.

4.1.2 Results and Discussions

In modeling artificial neural network's, MLP3 network is considered and trained by varying number of hidden neurons. The neural network is trained using two supervised

learning algorithms namely Backpropagation and Quasi-Newton. The best model in neural network is selected based on the validation error E_{avg} and the optimal value of E_{avg} is zero. Tables 4.1 and 4.2 represent the training and validation errors of ANN approach using Backpropagation and Quasi-Newton algorithms, respectively.

Table 4.1: Training and validation errors of ANN models using Backpropagation algorithm.

No. of Hidden Neurons	**CV Type**	**Training Error**	**Validation Error**	
			E_{avg} (%)	**E_{worst} (%)**
10	7-fold CV	4.67	4.77	61.98
	10-fold CV	4.40	**4.52**	53.95
20	7-fold CV	4.57	4.64	61.46
	10-fold CV	4.47	4.53	53.74
30	7-fold CV	4.77	4.77	62.05
	10-fold CV	4.50	4.58	53.80
40	7-fold CV	4.46	**4.58**	61.54
	10-fold CV	4.48	4.63	53.86
50	7-fold CV	4.92	5.00	60.51
	10-fold CV	4.53	4.55	53.49
60	7-fold CV	4.59	4.70	60.54
	10-fold CV	4.60	4.67	52.92
70	7-fold CV	4.79	4.84	59.91
	10-fold CV	4.67	4.72	53.79
80	7-fold CV	6.41	6.41	64.12
	10-fold CV	5.42	5.52	55.22
90	7-fold CV	5.30	5.33	61.12
	10-fold CV	5.08	5.08	53.04

Table 4.2: Training and validation Errors of ANN models using the Quasi-Newton algorithm.

No. of Hidden Neurons	CV Type	Training Error	Validation Error	
			E_{avg} (%)	E_{worst} (%)
10	7-fold CV	4.25	4.42	59.56
	10-fold CV	4.25	4.43	52.89
20	7-fold CV	4.25	**4.34**	60.02
	10-fold CV	4.21	**4.36**	52.35
30	7-fold CV	4.24	4.42	59.13
	10-fold CV	4.21	4.40	52.60
40	7-fold CV	4.22	4.41	59.83
	10-fold CV	4.29	4.49	52.09
50	7-fold CV	4.23	4.37	59.47
	10-fold CV	4.22	4.37	52.41
60	7-fold CV	4.23	4.44	59.59
	10-fold CV	4.21	4.40	51.86
70	7-fold CV	4.23	4.42	59.58
	10-fold CV	4.20	4.40	52.09
80	7-fold CV	4.24	4.45	59.50
	10-fold CV	4.22	4.38	52.46
90	7-fold CV	4.23	4.40	59.54
	10-fold CV	4.22	4.37	52.21

From Tables 4.1 and 4.2, it can be observed that training by Quasi-Newton algorithm has relatively more accurate results compared to Backpropagation. From Table 4.1, it can be observed that ANN approach using the Backpropagation algorithm and the 7-fold cross-validation (CV) method has least validation error (E_{avg}) of 4.58 for 40 hidden neurons, while using 10-fold cross-validation (CV) method it has least E_{avg} of

4.52 for 10 hidden neurons. From Table 4.2, it can be observed that ANN approach using Quasi-Newton algorithm and the 7-fold CV method has the least E_{avg} of 4.34 for 20 hidden neurons, while using the 10-fold CV method it has least E_{avg} of 4.36 for 20 hidden neurons. Further, the performance of best models in Artificial Neural Network (ANN) is compared with conventional interpolation techniques, KBNN, CBNN, SVR and RFR in Table 5.4.

4.2 Knowledge Based Neural Network Approaches

KBNN are more advanced ANNs that enhances both the generalization and the extrapolation capabilities of a neural model [52]. The generalization capability allows a neural model to predict unknown data, while the extrapolation capability predicts data that is beyond the training range. The existing implementation of KBNNs is limited to a split-sample validation technique. A knowledge-based neural network, in simple terms, is described as 'adding a prior knowledge to the existing neural networks' i.e., adding an additional input such as uranium concentration [20] to the conventional ANN. The different Knowledge Based Neural Network approaches such as Prior Knowledge Input (PKI), Source Difference Method (SDM) and Space-Mapped Neural Networks (SMNN) are further discussed.

4.2.1 Methodology

The PKI method, proposed in Watson et al. [40], was used to estimate the radon concentration by Akkala et al. [17] but was limited to split-sample validation. This method uses the knowledge input as an additional input to the existing neural network,

which is MLP3 as shown in Figure 4-2. In this thesis, uranium concentration is used as an additional input. The PKI method is trained using latitude, longitude, and uranium concentration as inputs and the radon concentration as the desired output. Once the model is trained it is used to predict the radon concentration. The limitation in using a PKI method is the uranium concentration availability for each individual sample where radon concentration needs to be predicted.

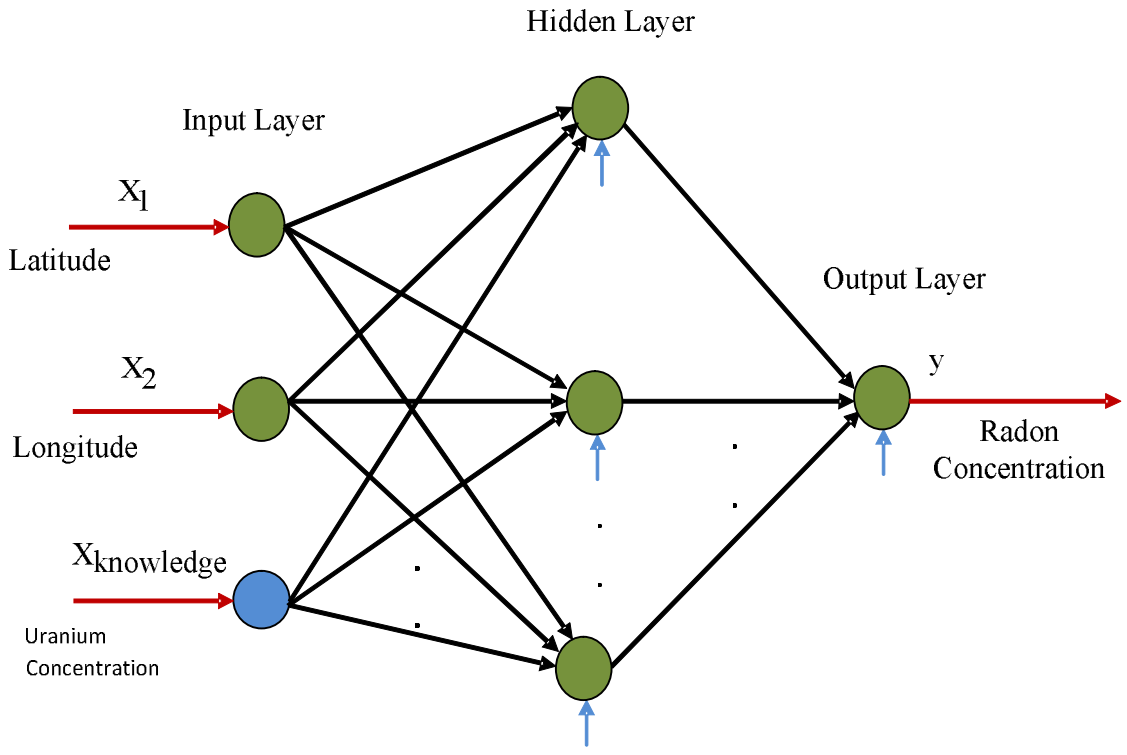

Figure 4-2: MLP3 Network Architecture for PKI Method

The Source Difference Method (SDM) was proposed in Gupta et al. [41] and was used to estimate the radon concentration in Akkala et al. [17]. This method consists of two MLP3 neural networks that are named as the coarse model and the difference model. The coarse model is used to model the functional relationship between the latitude and longitude as inputs and radon concentration as desired output. Once the model is trained,

it is used to predict the radon concentration. The aim of the method is to add an additional knowledge i.e., an estimation error to the predicted output from coarse model. The best coarse model is evaluated by changing the number of hidden neurons.

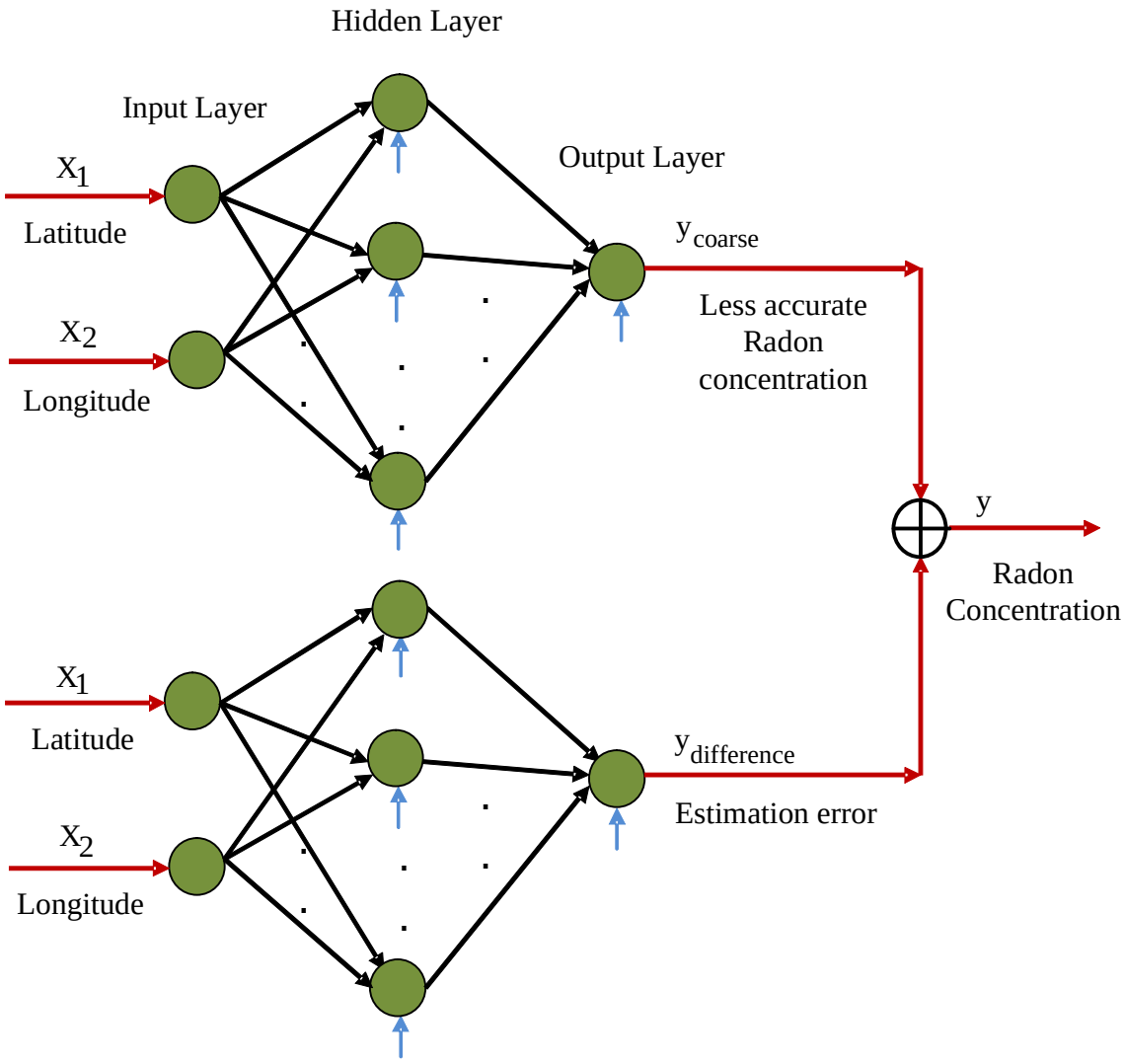

Figure 4-3: MLP3 Network Architecture for SDM Method

Once the best coarse model is evaluated, an estimation error, i.e., the difference between the predicted radon concentration from coarse model and the actual radon concentration available is calculated. The difference model is used to model the functional relationship between latitude and longitude as inputs and the estimation error

as desired output. The best difference model is evaluated by changing the number of hidden neurons. Once the best difference model is evaluated, it is combined with the coarse model to obtain the final radon concentrations. The MLP3 network architecture for SDM is shown in Figure 4-3.

The Space-Mapped Neural Networks (SMNN) was proposed in Devabhaktuni et al. [53] and was used to estimate the radon concentration in Akkala et al. [17]. The SMNN is used to map the fine model input-space into a coarse model input-space using a Space-Mapping (SM) technique. A fine model in this work is assumed to be the radon concentration obtained from different commercial testing devices in Ohio. At first, a MLP3 network is used to train the model using radon concentration as input where the desired outputs are latitude and longitude. The predicted output from this model is used to predict the output *latitude'* and *longitude'*. The SM technique is used to map the fine model input-space x to a coarse model input-space x_{coarse}. The SM model is trained using latitude, longitude and uranium concentration as inputs while *latitude'* and *longitude'* as desired outputs. Thus a space-mapping technique identifies a mathematical relationship between the SM network and coarse network. Once, the SM network is trained and used for predicting the *latitude'* and *longitude'*, the coarse model is used to train the MLP3 network using the predicted *latitude'* and *longitude'* as inputs where radon concentration is the desired output. The MLP3 network architecture for SMNN is shown in Figure 4-4. Finally, the coarse model is used for predicting the expected radon concentration.

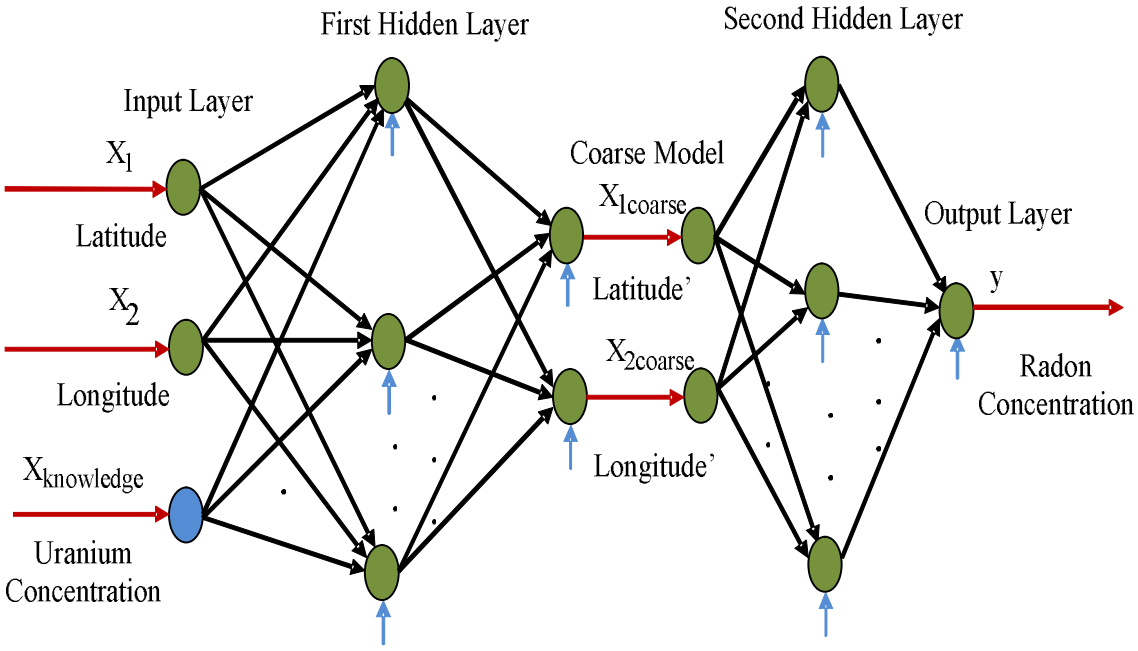

Figure 4-4: MLP3 Network Architecture for SMNN Method

4.2.2 Results and Discussions

In modeling PKI approach, MLP3 network is trained with a varying number of hidden neurons in the hidden layer. The neural network is trained using two supervised learning algorithms namely Back propagation and quasi-Newton. The best model in neural network is selected based on the validation error E_{avg}. The optimal value of E_{avg} is zero. Table 4.3 represents the training and validation errors of knowledge based neural networks PKI approach using Back propagation algorithm and Table 4.4 represents the training and validation errors using quasi-Newton algorithm.

Table 4.3: Training and validation errors of PKI models using Backpropagation Algorithm.

No. of Hidden Neurons	CV Type	Training Error	Validation Errors	
			E_{avg} (%)	E_{worst} (%)
10	7-fold CV	4.49	4.62	61.48
	10-fold CV	4.53	4.63	53.50
20	7-fold CV	4.78	4.85	62.34
	10-fold CV	4.64	4.71	53.13
30	7-fold CV	4.43	**4.50**	61.73
	10-fold CV	4.52	4.64	54.59
40	7-fold CV	4.57	4.65	60.89
	10-fold CV	4.37	**4.48**	54.26
50	7-fold CV	4.68	4.76	60.08
	10-fold CV	4.44	4.49	54.13
60	7-fold CV	4.59	4.62	59.75
	10-fold CV	4.55	4.70	53.62
70	7-fold CV	4.96	5.05	59.30
	10-fold CV	4.73	4.85	52.93
80	7-fold CV	5.49	5.54	67.24
	10-fold CV	4.75	4.86	53.36
90	7-fold CV	5.37	5.48	60.19
	10-fold CV	4.66	6.67	182.14

From Table 4.3, it can be observed that PKI approach using the Backpropagation algorithm and the 7-fold CV method has least E_{avg} of 4.50 for 30 hidden neurons, while using the 10-fold CV method it has least E_{avg} of 4.48 for 40 hidden neurons.

Table 4.4: Training and validation errors of PKI models using Quasi-Newton Algorithm.

No. of Hidden Neurons	CV Type	Training Error	Validation Errors	
			E_{avg} (%)	E_{worst} (%)
10	7-fold CV	4.33	**4.56**	58.89
	10-fold CV	4.31	4.58	52.15
20	7-fold CV	4.27	4.57	59.73
	10-fold CV	4.26	**4.54**	52.58
30	7-fold CV	4.28	4.56	59.47
	10-fold CV	4.27	4.63	53.01
40	7-fold CV	4.27	4.60	60.40
	10-fold CV	4.25	4.62	52.19
50	7-fold CV	4.26	4.61	61.33
	10-fold CV	4.25	4.62	52.83
60	7-fold CV	4.27	4.60	59.26
	10-fold CV	4.25	4.60	52.98
70	7-fold CV	4.28	4.71	59.43
	10-fold CV	4.25	4.58	52.50
80	7-fold CV	4.29	4.65	61.65
	10-fold CV	4.25	4.61	52.59
90	7-fold CV	4.27	4.62	60.04
	10-fold CV	4.24	4.60	52.27

From Table 4.4, it can be observed that PKI approach using the Quasi-Newton algorithm and the 7-fold CV method has least E_{avg} of 4.56 for 10 hidden neurons, while using 10-fold CV method it has least E_{avg} of 4.54 for 20 hidden neurons.

In modeling SDM approach, two models, namely the coarse and difference models are trained and validated with varying number of hidden neurons. In this thesis, the coarse model is considered as the ANN modeling approach, for which the training and validation errors are shown in Tables 4.1 and 4.2 using Backpropagation and Quasi-Newton algorithms, respectively. From Table 4.1, it is evident that coarse model using the Backpropagation algorithm and the 7-fold CV method has least E_{avg} of 4.58 for 40 hidden neurons, while using 10-fold CV method it has least E_{avg} of 4.52 for 10 hidden neurons. From Table 4.2, it is evident that coarse model using Quasi-Newton algorithm and the 7-fold CV method has the least E_{avg} of 4.34 for 20 hidden neurons, while using the 10-fold CV method it has least E_{avg} of 4.36 for 20 hidden neurons. Once the best coarse model has been identified, the difference model is trained and validated using an estimation error (i.e., the difference between the predictions of best coarse model and the actual radon values) as the desired output. The training and validation data for the difference model are obtained by replacing the radon values with estimation error for their respective samples. Further, the difference model is trained and validated with a varying number of hidden neurons in the hidden layer. The training and validation errors of the difference model are tabulated in Table 4.5 for the one using Backpropagation algorithm and in Table 4.6 for the one using quasi-Newton algorithm.

Table 4.5: Training and validation errors of difference models in SDM approach using Backpropagation algorithm.

No. of Hidden Neurons	CV Type	Training Error	Validation Errors	
			E_{avg} (%)	E_{worst} (%)
10	7-fold CV	4.48	4.60	59.20
	10-fold CV	4.34	4.40	52.57
20	7-fold CV	4.47	4.55	59.05
	10-fold CV	4.31	**4.32**	52.26
30	7-fold CV	4.78	4.95	59.42
	10-fold CV	4.50	4.32	45.47
40	7-fold CV	4.72	4.83	58.69
	10-fold CV	4.40	4.43	52.24
50	7-fold CV	4.34	4.43	59.70
	10-fold CV	4.49	4.56	52.06
60	7-fold CV	4.32	**4.38**	60.00
	10-fold CV	4.43	4.42	51.77
70	7-fold CV	4.40	4.49	58.83
	10-fold CV	4.33	4.39	52.33
80	7-fold CV	4.39	4.48	55.75
	10-fold CV	5.28	5.33	54.24
90	7-fold CV	5.34	5.38	60.58
	10-fold CV	5.07	5.03	52.95

From Table 4.5, it can be observed that difference model using Backpropagation algorithm and the 7-fold CV method has least E_{avg} of 4.38 for 60 hidden neurons, while using the 10-fold CV method it has least E_{avg} of 4.32 for 20 hidden neurons.

The best SDM model is obtained by adding the predicted values of the best coarse model using the Backpropagation algorithm with 40 hidden neurons and the difference model with 60 hidden neurons using the 7-fold CV method, while using the 10-fold CV method SDM model is obtained by adding the coarse model using the Backpropagation algorithm with 10 hidden neurons and difference model with 20 hidden neurons. Hence, by combining the two models where Backpropagation algorithm is used for training, the resulting validation error E_{avg} of SDM is 4.48 while using the 7-fold CV method and 4.66 while using the 10-fold CV method.

Table 4.6: Training and validation errors of difference models in SDM Approach using Quasi-Newton algorithm.

No. of Hidden Neurons	**CV Type**	**Training Error**	**Validation Errors**	
			E_{avg} (%)	**E_{worst} (%)**
10	7-fold CV	4.31	**4.45**	60.89
	10-fold CV	4.30	**4.45**	53.56
20	7-fold CV	4.31	4.46	60.89
	10-fold CV	4.30	4.45	53.57
30	7-fold CV	4.31	4.46	60.89
	10-fold CV	4.30	4.45	53.57
40	7-fold CV	4.31	4.48	60.56
	10-fold CV	4.30	4.46	53.60
50	7-fold CV	4.31	4.46	60.90
	10-fold CV	4.30	4.46	53.54
60	7-fold CV	4.31	4.46	60.89
	10-fold CV	4.30	4.46	53.62
70	7-fold CV	4.31	4.48	60.44
	10-fold CV	4.30	4.46	53.73

80	7-fold CV	4.31	4.46	60.90
	10-fold CV	4.30	4.46	53.63
90	7-fold CV	4.31	4.47	60.90
	10-fold CV	4.29	4.47	53.72

From Table 4.6, it can be observed that difference model using Quasi-Newton algorithm and the 7-fold CV method has least E_{avg} of 4.45 for 10 hidden neurons, while using the 10-fold CV method it has least E_{avg} of 4.45 for 10 hidden neurons.

The best SDM model is obtained by adding the predicted values of the best coarse model using the Quasi-Newton algorithm with 20 hidden neurons and the difference model with 10 hidden neurons, in both the cases of using the 7-fold CV method or the 10-fold CV method. Hence, by combining the two models using the Quasi-Newton algorithm for training, the resulting validation error E_{avg} in SDM is 4.34 while using the 7-fold CV method and 4.35 while using the 10-fold CV method.

In modeling the SMNN approach, two models namely the SM model and coarse model are trained and validated by varying number of hidden neurons. Tables 4.7 and 4.8 represent the training and validation errors of SM model using Backpropagation and Quasi-Newton algorithms, respectively. Once, the SM model is trained and validated, a best model is selected based on the validation error criterion $E_{avg}(\%)$. The space mapped *latitude'* and *longitude'* are then used for training the coarse model by varying number of hidden neurons. while the radon concentration is the desired output. Tables 4.9 and 4.10 represent the training and validation errors of coarse model using Backpropagation and Quasi-Newton algorithms, respectively.

Table 4.7: Training and validation errors of SM models in SMNN Approach using Backpropagation algorithm.

No. of Hidden Neurons	CV Type	Training Error	Validation Errors	
			E_{avg} (%)	E_{worst} (%)
10	7-fold CV	9.57	9.85	57.85
	10-fold CV	10.94	11.33	56.80
20	7-fold CV	9.68	9.89	55.33
	10-fold CV	11.17	11.51	56.80
30	7-fold CV	9.59	9.76	56.93
	10-fold CV	11.01	11.23	57.26
40	7-fold CV	9.55	9.78	57.26
	10-fold CV	10.99	11.20	58.01
50	7-fold CV	9.72	9.94	59.95
	10-fold CV	10.89	**11.13**	57.44
60	7-fold CV	9.61	9.81	59.50
	10-fold CV	11.12	11.47	57.51
70	7-fold CV	9.82	9.97	57.21
	10-fold CV	11.19	11.56	56.43
80	7-fold CV	9.71	**9.75**	56.40
	10-fold CV	10.99	11.50	75.02
90	7-fold CV	13.68	17.44	67.72
	10-fold CV	11.15	11.55	57.64

From Table 4.7, it is evident that SM model using Backpropagation algorithm and the 7-fold CV method has least E_{avg} of 9.75 for 80 hidden neurons, while using the 10-fold CV method it has the least E_{avg} of 11.13 for 50 hidden neurons.

Table 4.8: Training and validation errors of SM Model in SMNN approach using Quasi-Newton algorithm.

No. of Hidden Neurons	CV Type	Training Error	Validation Errors	
			E_{avg} (%)	E_{worst} (%)
10	7-fold CV	8.32	8.48	52.64
	10-fold CV	7.37	7.75	54.79
20	7-fold CV	8.21	**8.47**	52.97
	10-fold CV	7.27	7.76	55.12
30	7-fold CV	8.14	8.53	52.47
	10-fold CV	7.24	7.71	55.20
40	7-fold CV	8.15	8.56	53.13
	10-fold CV	7.24	7.75	54.55
50	7-fold CV	8.14	8.54	53.88
	10-fold CV	7.20	7.73	53.75
60	7-fold CV	8.12	8.56	55.04
	10-fold CV	7.22	7.72	54.10
70	7-fold CV	8.08	8.56	52.52
	10-fold CV	7.21	7.77	53.95
80	7-fold CV	8.12	8.67	53.22
	10-fold CV	7.20	**7.67**	53.98
90	7-fold CV	8.07	8.60	57.20
	10-fold CV	7.20	7.72	53.49

From Table 4.8, it is evident that SM model using Quasi-Newton algorithm and the 7-fold CV method has least E_{avg} of 8.47 for 20 hidden neurons, while using the 10-fold CV method it has the least E_{avg} of 7.67 for 80 hidden neurons.

Table 4.9: Training and validation errors of coarse model in SMNN approach using Backpropagation algorithm.

No. of Hidden Neurons	CV Type	Training Error	Validation Errors	
			E_{avg} (%)	E_{worst} (%)
10	7-fold CV	4.40	4.90	61.75
	10-fold CV	4.81	5.02	52.05
20	7-fold CV	5.15	5.27	59.90
	10-fold CV	4.27	**4.41**	53.02
30	7-fold CV	4.45	4.53	60.61
	10-fold CV	4.35	4.53	53.25
40	7-fold CV	6.90	6.95	59.87
	10-fold CV	4.29	4.42	53.12
50	7-fold CV	4.34	**4.48**	61.28
	10-fold CV	4.27	4.44	52.78
60	7-fold CV	4.31	5.36	59.33
	10-fold CV	4.36	4.55	52.38
70	7-fold CV	4.42	4.87	55.55
	10-fold CV	4.29	4.44	52.54
80	7-fold CV	3.72	5.16	62.24
	10-fold CV	4.32	4.49	52.54
90	7-fold CV	3.75	11.08	78.99
	10-fold CV	4.58	5.63	55.30

From Table 4.9, it is observed that coarse model using Backpropagation algorithm and the 7-fold CV method has least validation E_{avg} of 4.48 for 50 hidden neurons, while using the 10-fold CV method it has the least E_{avg} of 4.41 for 20 hidden neurons.

Table 4.10: Training and validation errors of coarse models in SMNN approach using Quasi-Newton algorithm.

No. of Hidden Neurons	CV Type	Training Error	Validation Errors	
			E_{avg} (%)	E_{worst} (%)
10	7-fold CV	4.11	**4.49**	60.37
	10-fold CV	4.12	4.39	52.55
20	7-fold CV	4.12	4.56	60.65
	10-fold CV	4.12	4.41	52.63
30	7-fold CV	4.30	4.67	60.48
	10-fold CV	4.11	4.39	52.44
40	7-fold CV	4.11	4.61	63.93
	10-fold CV	4.11	4.46	52.77
50	7-fold CV	4.12	4.51	60.22
	10-fold CV	4.19	4.39	52.60
60	7-fold CV	4.11	4.60	63.34
	10-fold CV	4.10	4.38	52.70
70	7-fold CV	4.11	4.62	63.46
	10-fold CV	4.13	4.40	52.61
80	7-fold CV	4.13	4.56	61.62
	10-fold CV	4.12	4.41	52.84
90	7-fold CV	4.12	4.53	60.31
	10-fold CV	4.12	**4.37**	52.53

From Table 4.10, it is observed that coarse model using Quasi-Newton algorithm and the 7-fold CV method has least E_{avg} of 4.49 for 10 hidden neurons, while using the 10-fold CV method it has the least E_{avg} of 4.37 for 90 hidden neurons. Further, the performance of best models in KBNNs is compared with conventional interpolation techniques, ANN, CBNN, SVR and RFR in Table 5.4.

4.3 Correction-Based Neural Network Approach

The correction model based ANN approach is a more advanced ANN that enhances both the generalization and extrapolation capabilities, as well as reduces the structural complexity of a neural network model [18].

4.3.1 Methodology

This approach includes the conventional neural model f_{ann} and a set of candidate correction models $f_{ann,j}$ in order to achieve the condition $E_{obj} < E_{user}$ and improve the prediction accuracy of f_{ann}[54]. Here, E_{user} is a user defined error and E_{obj} is the validation error of each sample in $f_{ann,j}$ given as

$$E_{obj} = \frac{original - predicted}{original} * 100 \tag{4.8}$$

The desired output of candidate correction model is taken as j^{th} element of input vector in f_{ann} and the input is replaced with the j^{th} element of f_{ann} desired output. The structure of a correction models $f_{ann,j}$ is given as

$$x_{1,c} = f_{ann,1}(y, x_2, w_1), \tag{4.9}$$

$$x_{2,c} = f_{ann,2}(x_1, y, w_2), \tag{4.10}$$

Eqs. (4.9) and (4.10) represent the general structure of correction models in this thesis, where $x_{1,c}$ and $x_{2,c}$ represent the desired outputs to latitude and longitude respectively, y is the radon concentration, x_1 and x_2 are latitude and longitude respectively, w_1 and w_2 are the ANN weight vectors of the first candidate correction model and second candidate correction model respectively. The MLP3 network

architecture for the two candidate correction models is shown in Figure 4-5 and 4-6. After training and validating the two candidate correction models, the best candidate correction model has to be identified based on the error criterion E_{avg}, given in eq. (4.7).

Once the best candidate correction model out of the two candidate correction models is identified, it is combined with the less accurate f_{ann} and a root finding algorithm is iteratively implemented, such that the condition $E_{obj} < E_{user}$ is satisfied.

In this thesis, a sensitivity-based root finding algorithm [54] has been implemented, which uses the partial derivatives to find the step-size and update direction. The partial derivatives of the candidate correction model $f_{ann,j}$ with respect to output y is obtained by applying the chain rule of calculus and is given as

$$\frac{\partial x_{j,c}}{\partial y} = \sum_{k=1}^{r} \frac{\partial x_{j,c}}{\partial z_k} \frac{\partial z_k}{\partial y_k} \frac{\partial y_k}{\partial y} \tag{4.11}$$

In eq. (4.11), r is the number of hidden neurons, y_k is weighted sum of all inputs to $f_{ann,j}$ and z_k is the sigmoid activation function. Since, the sigmoid activation function is computationally better over other functions such as arc tangent or hyperbolic tangent; it is employed in ANN training. The sigmoid activation function z_k is given as

$$z_k = \frac{1}{1 + e^{-y_k}} \tag{4.12}$$

In the initial stages of implementing the root finding algorithm, output y from f_{ann} is taken as the initial input and applied on "f_{ann} and $f_{ann,j}$ pair". The output y is then constantly updated using a root finding algorithm until the condition $E_{obj} < E_{user}$ is satisfied for each subsample in the dataset. Figure 4-7 represents the flowchart of Correction-Based Neural Network using sensitivity based root finding algorithm.

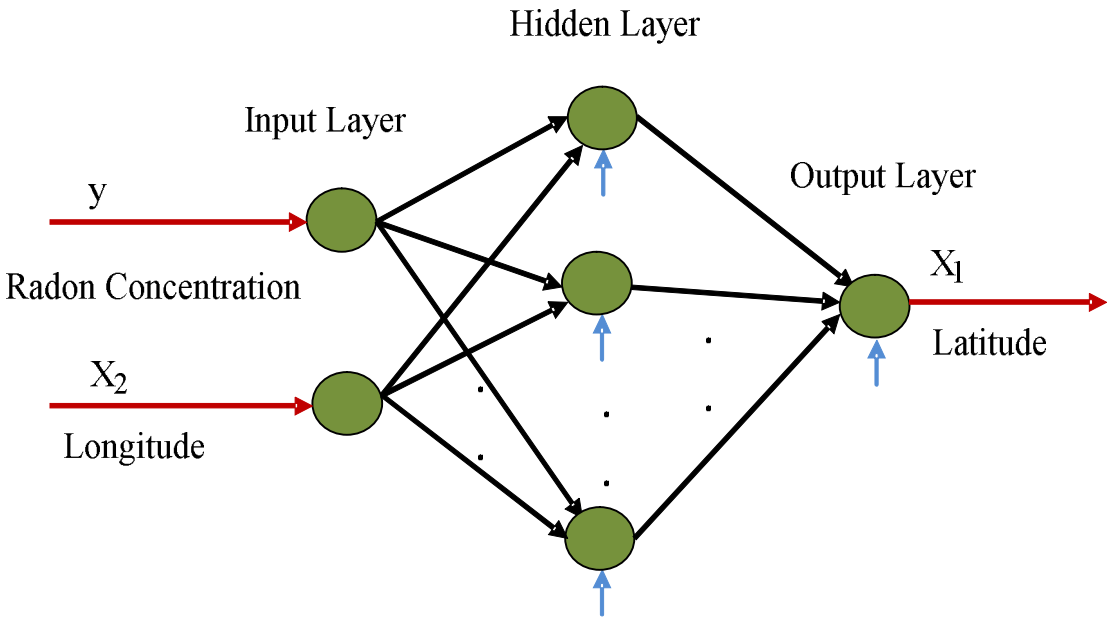

Figure 4-5: MLP3 Network Architecture for Candidate Correction Model 1 (Latitude as desired output)

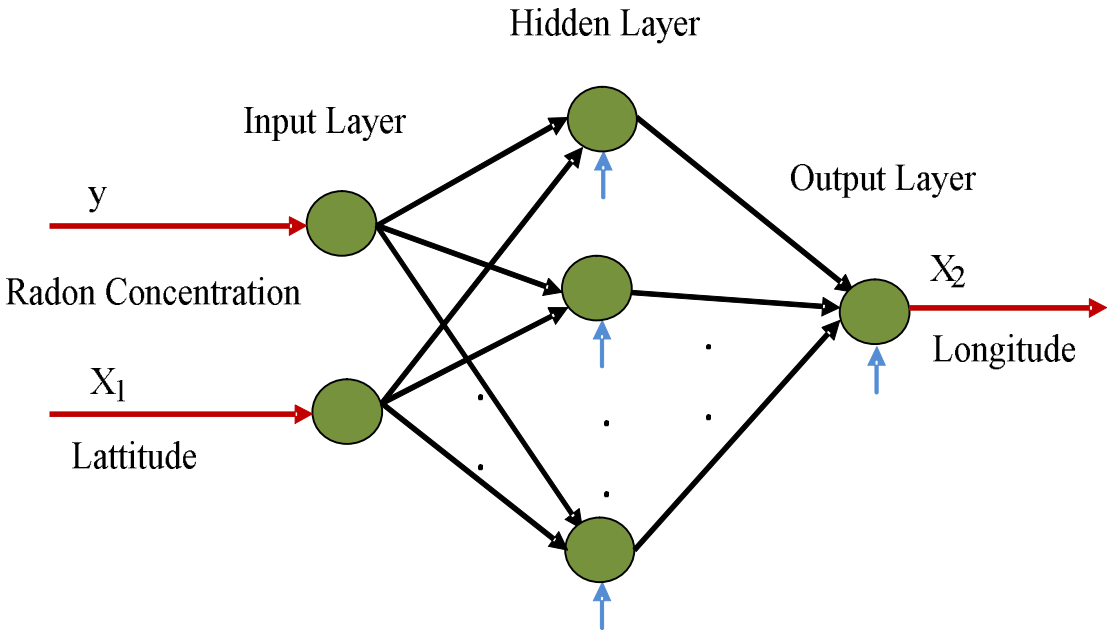

Figure 4-6: MLP3 Network Architecture for Candidate Correction Model 2 (Longitude as desired output)

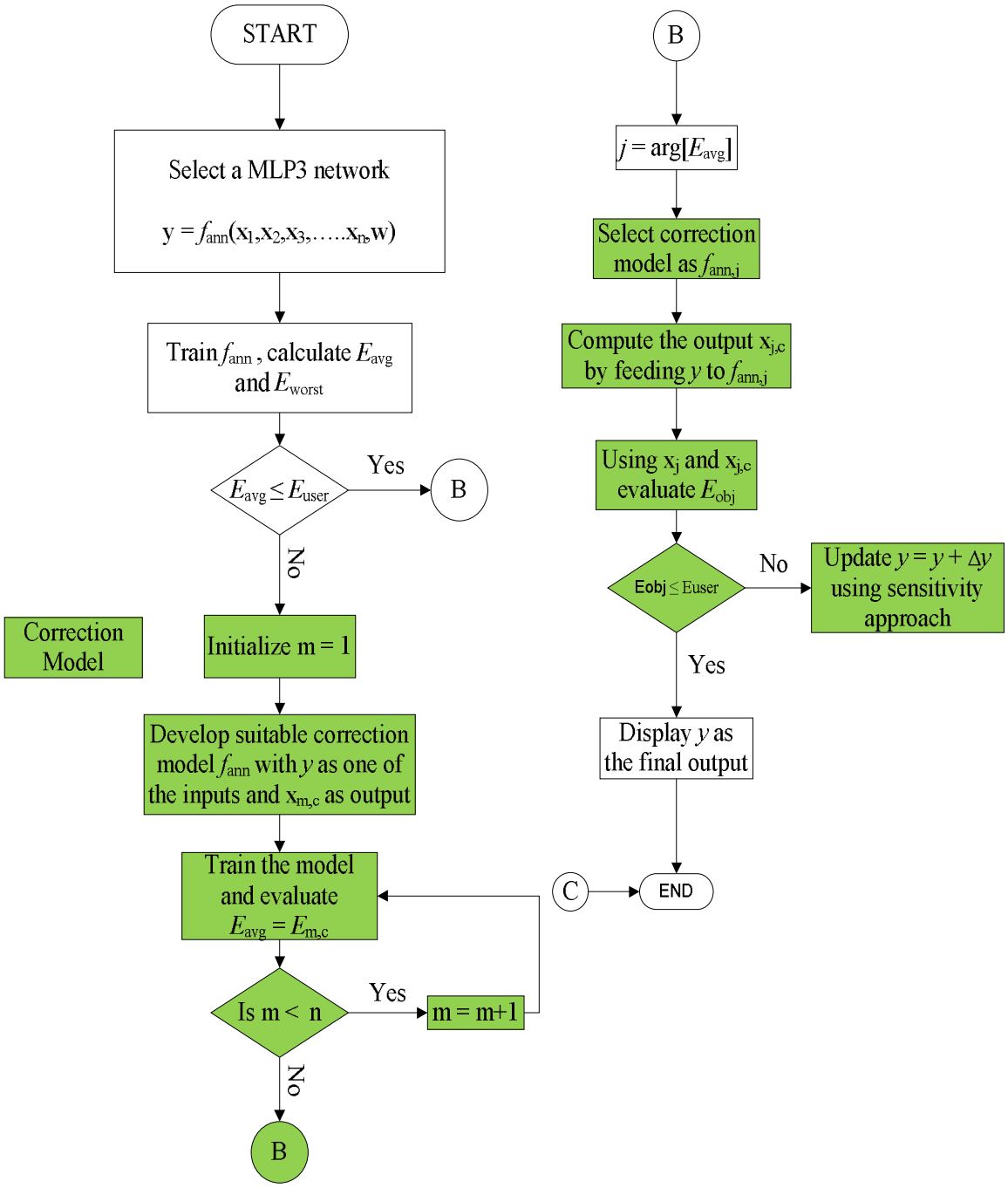

Figure 4-7: Flow Chart for Correction-Based Neural Networks using sensitivity based root finding algorithm

4.3.2 Results and Discussions

In modeling CBNN approach, the two candidate correction models are trained and validated, one with the latitude as desired output and the other with longitude as the desired output. Tables 4.11 and 4.12 represent the training and validation errors of candidate correction model 1 (latitude as the desired output) using Backpropagation and Quasi-Newton algorithms, respectively. Tables 4.13 and 4.14 represent the training and validation errors of candidate correction model 2 (longitude as the desired output) using Backpropagation and Quasi-Newton algorithms, respectively. The candidate correction model and the ANN model form a mutual supportive pair [18]. Hence, in order to keep the structure of both the models the same, a fixed number of hidden neurons are taken for both the models. Once the candidate correction model with least validation error is identified, it is paired with the less accurate ANN model, and a root finding algorithm is iteratively implemented such that the condition $E_{obj} < E_{user}$ is satisfied. In the root finding algorithm, for each sample the E_{obj} is calculated as given in eq. (4.8).

Table 4.11: Training and validation errors of candidate correction model 1(latitude as desired output) using Backpropagation algorithm.

No. of Hidden Neurons	CV Type	Training Error	Validation Errors	
			E_{avg} (%)	E_{worst} (%)
10	7-fold CV	18.42	18.63	54.46
	10-fold CV	18.45	**18.46**	50.11
20	7-fold CV	18.57	18.78	54.97
	10-fold CV	18.54	18.56	51.53
30	7-fold CV	18.51	**18.57**	52.98
	10-fold CV	18.51	18.68	50.57
40	7-fold CV	18.48	18.64	54.01
	10-fold CV	18.42	18.52	50.63
50	7-fold CV	18.51	18.84	61.99
	10-fold CV	18.42	18.65	58.68
60	7-fold CV	18.47	19.18	148.02
	10-fold CV	18.75	18.93	58.24
70	7-fold CV	18.41	19.14	137.69
	10-fold CV	18.48	18.56	54.24
80	7-fold CV	18.40	18.98	104.50
	10-fold CV	18.56	18.78	56.99
90	7-fold CV	19.05	19.19	75.05
	10-fold CV	19.03	19.93	144.60

From Table 4.11, it is observed that candidate correction model 1 using the Backpropagation algorithm and the 7-fold CV method has least E_{avg} of 18.57 for 30 hidden neurons, while using the 10-fold CV method it has the least E_{avg} of 18.46 for 10 hidden neurons.

Table 4.12: Training and validation errors of candidate correction model 1(latitude as desired output) using Quasi-Newton algorithm.

No. of Hidden Neurons	CV Type	Training Error	Validation Errors	
			Eavg (%)	Eworst (%)
10	7-fold CV	16.50	17.38	87.72
	10-fold CV	16.39	17.10	59.54
20	7-fold CV	16.15	**16.98**	73.46
	10-fold CV	16.22	17.07	77.87
30	7-fold CV	16.12	17.03	77.87
	10-fold CV	16.19	17.10	72.49
40	7-fold CV	16.18	17.25	107.92
	10-fold CV	16.15	16.87	67.88
50	7-fold CV	16.14	17.29	99.70
	10-fold CV	16.17	17.15	90.42
60	7-fold CV	16.12	17.18	79.08
	10-fold CV	16.14	16.94	68.07
70	7-fold CV	16.14	17.36	114.44
	10-fold CV	16.18	16.93	59.89
80	7-fold CV	16.21	17.17	92.44
	10-fold CV	16.22	17.04	74.06
90	7-fold CV	16.18	17.25	73.51
	10-fold CV	16.21	**16.84**	70.45

From Table 4.12, it is observed that candidate correction model 1 using Quasi-Newton algorithm and the 7-fold CV method has least E_{avg} of 16.98 for 20 hidden neurons, while using the 10-fold CV method it has the least validation E_{avg} of 16.84 for 90 hidden neurons.

Table 4.13: Training and validation errors of candidate correction model 2(longitude as desired output) using Backpropagation algorithm.

No. of Hidden Neurons	CV Type	Training Error	Validation Errors	
			E_{avg} (%)	E_{worst} (%)
10	7-fold CV	24.32	24.69	63.76
	10-fold CV	24.04	24.31	58.89
20	7-fold CV	24.18	24.52	58.74
	10-fold CV	24.13	24.30	54.45
30	7-fold CV	24.24	24.50	67.99
	10-fold CV	24.14	24.35	58.94
40	7-fold CV	24.28	24.71	66.22
	10-fold CV	24.32	24.68	61.40
50	7-fold CV	24.04	24.39	84.18
	10-fold CV	24.12	24.26	58.55
60	7-fold CV	24.07	**24.24**	59.61
	10-fold CV	23.76	**23.96**	60.28
70	7-fold CV	24.27	24.35	60.22
	10-fold CV	23.84	24.55	102.46
80	7-fold CV	24.10	24.73	73.45
	10-fold CV	23.95	24.31	60.69
90	7-fold CV	24.20	24.44	88.80
	10-fold CV	24.11	24.13	61.08

From Table 4.13, it is observed that candidate correction model 2 using Backpropagation algorithm and the 7-fold CV method has least E_{avg} of 24.24 for 60 hidden neurons, while using the 10-fold CV method it has the least E_{avg} of 23.96 for 60 hidden neurons.

Table 4.14: Training and validation errors of candidate correction model 2 (longitude as desired output) using Quasi-Newton algorithm.

No. of Hidden Neurons	CV Type	Training Error	Validation Errors	
			E_{avg} (%)	E_{worst} (%)
10	7-fold CV	21.80	22.66	106.80
	10-fold CV	21.72	22.79	71.71
20	7-fold CV	21.20	22.88	113.14
	10-fold CV	21.29	22.61	98.87
30	7-fold CV	21.17	22.61	129.42
	10-fold CV	21.37	22.80	101.21
40	7-fold CV	20.97	**22.38**	81.58
	10-fold CV	21.20	22.97	127.81
50	7-fold CV	21.08	23.19	199.34
	10-fold CV	21.28	**22.26**	76.77
60	7-fold CV	21.31	22.65	114.77
	10-fold CV	21.13	22.78	124.51
70	7-fold CV	21.25	22.40	100.63
	10-fold CV	21.19	22.51	88.24
80	7-fold CV	21.28	22.98	163.80
	10-fold CV	21.19	22.83	119.81
90	7-fold CV	21.28	22.86	168.63
	10-fold CV	21.09	22.88	152.45

From Table 4.14, it is observed that candidate correction model 2 using Quasi-Newton algorithm and the 7-fold CV method has least validation E_{avg} of 22.38 for 40 hidden neurons, while using the 10-fold CV method it has the least E_{avg} of 22.26 for 50 hidden neurons.

To implement root finding algorithm the best candidate correction model of the two models need to be selected. From Tables 4.11 and 4.13, it is evident that candidate correction model 1 (i.e., latitude as the desired output), using a Backpropagation algorithm and the 7-fold CV method has least E_{avg} of 18.57 for 30 hidden neurons, while using the 10-fold CV method it has the least E_{avg} of 18.46 for 10 hidden neurons. From Table 4.12 and 4.14, it is evident that candidate correction model 1 (i.e., latitude as the desired output) using the Quasi-Newton algorithm and the 7-fold CV method has least E_{avg} of 16.98 for 20 hidden neurons, while using the 10-fold CV method it has the least E_{avg} of 16.84 for 90 hidden neurons. Once the best model of the two candidate correction models is selected, to keep the structure of both candidate correction model 1 and ANN the same, the model is fixed at 30 hidden neurons using the 7-fold CV method and the Backpropagation algorithm, while for the 10-fold CV method it is fixed at 10 hidden neurons. Similarly, the structure of models using the Quasi-Newton algorithm is kept same; the model is fixed at 20 hidden neurons using the 7-fold CV method, while for the 10-fold CV method it is fixed at 90 hidden neurons. After implementing the sensitivity based root finding algorithm where Backpropagation algorithm is used for training, the resulting validation error E_{avg} of CBNN is 2.71 while using the 7-fold CV method and 2.56 while using the 10-fold CV method. Similarly, root finding algorithm implemented where the Quasi-Newton algorithm is used for training, the resulting validation error E_{avg} of CBNN is 3.58 while using the 7-fold CV method and 3.62 while using the 10-fold CV method. Further, the performance of best models in CBNN is compared with conventional interpolation techniques, ANN, KBNN, SVR and RFR in Tables 5.3 and 5.4.

Chapter 5

Support Vector Regression and Random Forest Regression

5.1 Support Vector Regression

5.1.1 Review of Support Vector Machines

Support Vector Machines (SVM's) are widely used for data classification and regression in various researches. Unlike neural network techniques which are based on Empirical Risk Minimization Principle, SVM's prediction are based on Structural Minimization principle that overcomes the problem of local minimization and over-fitting. To overcome the drawbacks of the existing neural network techniques an enhanced SVM based error model namely nu-SVR technique [55] has been developed. The most commonly used versions of SVM regression are 'epsilon-SVR' and 'nu-SVR'. Previously, Support Vector Regression (SVR) applicability was based on the penalty parameters C [0, inf) and epsilon [0, inf) to optimize the individual samples which were not correctly predicted. Hence, to improvise the applicability of SVR with a slightly different penalty, epsilon parameter was replaced by nu (0, 1]. The motivation of nu

version in SVM is it represents an upper bound on the badly predicted errors and lower bound on the support vectors. Though the optimization problem solved in both the versions of SVR is the same; the penalty parameters epsilon and nu are slightly different. SVM's are used for classification in research areas of bioinformatics [56], intrusion detection [57], credit rating analysis [58], and for drug/non drug classification in [59]. SVMs are used for regression in evaporation estimation [60], and estimation of soil moisture [61]. Recently, a work done by Bhatt et al. [62] has shown that nu-SVR performed better than ANN for modeling and estimation of the MEMS sensor errors.

5.1.2 Methodology

The SVM type employed in this thesis is a nu-SVR modeling technique [62] which aims to approximate the nonlinear regression function given in eq. (5.1).

$$f(x) = w^T . \emptyset(x) + b \tag{5.1}$$

In eq. (5.1), w^T is the weight vector to the corresponding $\emptyset(x)$, $\emptyset(x)$ is a nonlinear mapping function which maps the input space to a higher dimensional space, and b is the bias. In order to approximate the nonlinear regression function, the parameters w and b need to be estimated, such that the function $f(x)$ should be as close as possible to desired output y and should be as flat as possible in order to hold back the problem of overfitting. To achieve the above two objectives of closeness and flatness, the primal problem of Nu-SVR given in eq. (5.1) has to be minimized

$$\frac{1}{2}||w||^2 + C\left\{\gamma . \varepsilon + \frac{1}{n}\sum_{i=1}^{n}(\xi + \xi^*)\right\} \tag{5.2}$$

having constraints to:

$$y_i - \langle w^T . \emptyset(x)\rangle - b \leq \varepsilon + \xi_i^*, \tag{5.3}$$

$$\langle w^T . \emptyset(x) \rangle + b - y_i \leq \varepsilon + \xi_i,$$

$$\xi_i^*, \xi_i \geq 0.$$

In eq. (5.2), ε is an insensitive loss function that is used to find absolute error between actual and predicted value, $\xi_i^* and\ \xi_i$ are slack variables introduced by Vapnik and Cortes [63]. $||w||^2$ is a parameter norm of $f(x)$ that is used to measure the flatness, C is a regularized parameter that determines the tradeoff between tolerance of error above ε and parameter norm, $\gamma(0 \leq \gamma \leq 1)$ is an upper bound on the fraction of margin errors in the training set and a lower bound on the fraction of support vectors. In cases where γ equals to both fractions, the dual problem of Nu-SVR is solved by constructing a Lagrange function (L), given as,

$$\begin{aligned} L(\alpha, \alpha*, \beta, \eta, \eta*) &= \frac{1}{2}||w||^2 + C\left\{\gamma.\varepsilon + \frac{1}{n}\sum_{i=1}^{n}(\xi + \xi^*)\right\} - \frac{1}{n}\sum_{i=1}^{n}(\eta.\xi + \eta^*.\xi^*) \\ &- \frac{1}{n}\sum_{i=1}^{n}(\varepsilon + \xi_i + y_i - w^t.\emptyset(x) - b) \\ &- \frac{1}{n}\sum_{i=1}^{n}(\epsilon + \xi_i - y_i + w^t.\emptyset(x) + b) - \beta.\epsilon \end{aligned} \tag{5.4}$$

In eq. (5.4), the Lagrange multipliers are $\alpha, \alpha*, \eta, \eta*, \beta$. Thus, optimizing the dual problem of Nu-SVR using partial derivatives of L with respect to variables such as $w, \varepsilon, b, \xi i$ yields to

$$maximum -\frac{1}{2}\sum_{i,j=1}^{n}(\alpha_i - \alpha_i^*).(\alpha_j - \alpha_j^*).K(x_i, x_j) + \sum_{i=1}^{n} y_i.(\alpha_i - \alpha_i^*) \tag{5.5}$$

having constraints to:

$$\sum_{i=1}^{n}(\alpha_i - \alpha_i^*) = 0,$$
$$\sum_{i=1}^{n}(\alpha_i + \alpha_i^*) \leq C\gamma, \tag{5.6}$$
$$\alpha_i, \alpha_i^* \in \left[0, \frac{C}{n}\right].$$

In eq. (5.5), $k(x_i, x_j)$ is a kernel function given as $k(x_i, x_j) = \emptyset(x_i)t.\emptyset(x_j)$. Precisely, optimization of the dual problem of Nu-SVR yields to Lagrange multipliers and weight parameter w as

$$w = \sum_{i=1}^{n}(\alpha_i - \alpha_i^*).\emptyset(x_i) \tag{5.6}$$

Hence substituting w in eq. (5.1) gives the approximated prediction function as

$$f(x) = \sum_{i=1}^{n}(\alpha_i - \alpha_i^*).K(x_i, x_j) + b \tag{5.7}$$

In eq. (5.7), b is a bias parameter that is identified using Kuhn, Tucker and Karush conditions given in [64, 65]. $k(x_i, x_j)$ is a kernel function, whose selection plays an important role in SVM, as it is a key point in increasing the prediction accuracy of the model. The different types of kernel functions that are frequently used in SVM are Polynomial Function (PF), Radial Basis Function (RBF), Sigmoid Function (SF) and Linear Function (LF). In this article, RBF kernel function is used, as it has better prediction accuracy and a relatively less complex model for implementation [66]. Identifying the Lagrange multipliers α, α^*, b using Nu-SVR approach and selecting an appropriate kernel function, the approximated function in eq. (5.7) can be used to predict the desired output.

5.1.3 Results and Discussions

In modeling support vector regression, the Nu-SVR technique has been implemented

using *LibSVM* software [37]. As the radial basis kernel function is easy to implement and produces accurate models it has been chosen as the kernel function in this work. Kernel function controls the shape of the separating hyper plane, and the associated parameter that does this is the gamma value. It is noted that as the gamma value increases, the number of support vectors increases hence improvising the prediction accuracy. Hence, the support vector regression is implemented by varying the gamma value in the kernel function. Table 5.1 represents the error criterion E_{avg} value with varying gamma value in the kernel. The equation for error criterion E_{avg} is the same as given in eq. (4.7).

Table 5.1: Validation Error of Support Vector Regression with varying Gamma Value

Gamma Value	**CV Type**	**E_{avg}**
0.01	7-fold	4.892964
	10-fold	4.798435
0.1	7-fold	4.25935
	10-fold	4.242329
1	7-fold	4.057695
	10-fold	4.045169
10	7-fold	**4.002825**
	10-fold	**4.000835**

From Table 5.1, it can be observed that as the gamma value increases the validation error decreases in SVR. It is evident that there is no significant change in the validation error when the gamma value is changed from 1 to 10. Further, the best model i.e., the SVR model using the RBF kernel function and gamma value as 10 is compared with other interpolation techniques in Table 5.4.

5.2 Random Forest Regression

5.2.1 Review of Random Forest Regression

Random Forest (RF) [68] is an ensemble learning technique used for data classification and regression applications. In random forests a forest of trees are grown from a bootstrap sample taken from the training data and the final output is taken as the average of individual tree outputs. RF gained popularity due to robustness and flexibility in handling complex input-output relationships. Random Forests are capable of generalizing high dimensional data effectively, and the inbuilt cross-validation capability improves the prediction accuracy, thus making it suitable for real-time implementation. RFR has been used in language modeling application to predict the next word based on words already seen before [69]. RF has also been used for classifying microarray data in bioinformatics [70, 71]. The application of RF as a classifier has also been used in ecology, for classifying data on invasive plant species presence in Lava Beds National Monument, California, USA, rare lichen species presence in the Pacific Northwest, USA, and nest sites for cavity nesting birds in the Uinta Mountains, Utah, USA [72]. Recently, application of RF is used for integrating Global Positioning System (GPS) and Inertial Navigation System (INS) for reliable, accurate and continuous navigation solution [73]. The current application of predicting Radon concentration with latitude and longitude as inputs, require spatial interpolation and thus the applicability of RFR on environmental data. The application of RFs to environmental data is for predicting seabed mud content based on samples extracted from Geoscience Australia's Marine Samples Database [74].

5.2.2 Methodology

Random Forest Regression (RFR) is a non-parametric regression approach that gained popularity for its robustness and flexibility in modeling the input-output functional relationship appropriately [72]. In RFR, a collection of regression trees are trained using different bootstrap samples from training data. While choosing the boot strap samples, some of the training data may be left out of the sample and some may be repeated in the sample. These left out data sample constitute the out-of-bag samples, which are used to calculate the learning error given in eq. (5.8), thus tests the training accuracy. This built-in cross-validation capability works with the help of out-of-bag samples and provides a realistic prediction error estimates during the training process thus, improving the generalization capability of the random forests. The final output is taken as the average of the individual tree outputs.

Given a input-output dataset i.e., $\{(x_1, y_1), .., (x_n, y_n)\}$, where x_i, $i = 1, .., n$ is an input vector containing latitude and longitude and y_i is the radon concentration, the training procedure is followed as:

1. A randomly selected sample with replacement or a bootstrap sample is collected from the available dataset.

2. Using the bootstrap sample, grow a tree to maximum size without pruning.

3. Repeat Step 2 until user defined number of trees are grown.

The above procedure results in a set of M trees $\{T_1(X), T_2(X), \dots, T_M(X)\}$, where $X = \{x_1, x_2, \dots, x_p\}$, is a p-dimension input vector that forms a forest. The ensemble produces M outputs corresponding to each tree $\hat{y}_1 = T_1(X), \dots, \hat{y}_m = T_M(X)$, where $\hat{y}_m, m = 1, \dots, M$, is the m^{th} tree output.

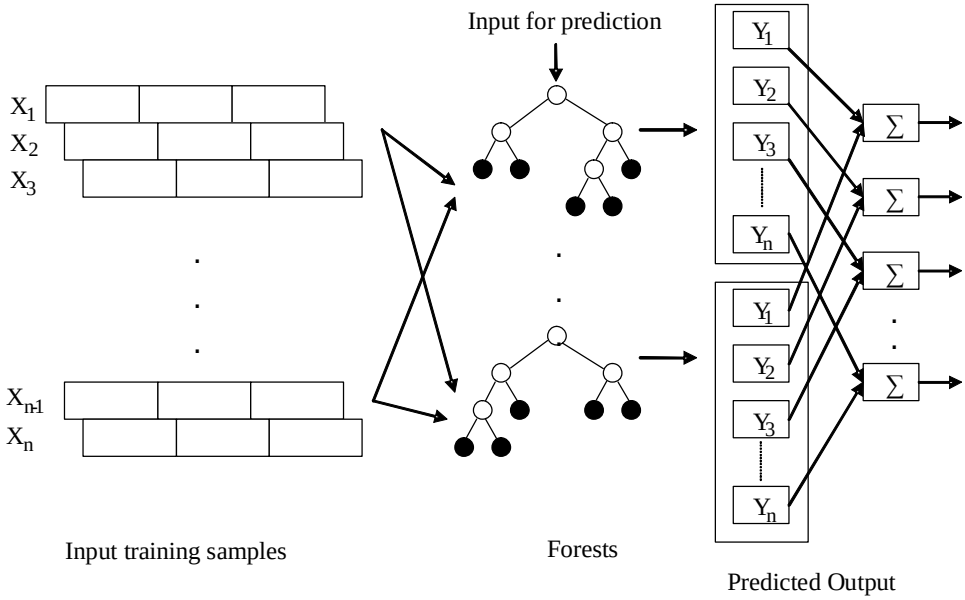

Figure 5-1: Random Forests Work Flow

The RFR workflow is shown in Figure 5-1, where input training samples x_1, x_2, . . , x_n shown on left is used to grow the user defined number of trees. The testing samples i.e., the inputs for prediction are passed along the trees and the final output is the average of individual tree estimates. To determine how efficient the random forest prediction would be when it is exposed to testing samples, the out-of-bag error estimate is calculated using eq. (5.8).

$$MSE \approx MSE^{OOB} = n^{-1} \sum_{i=1}^{n} [\hat{y}(x_i) - y_i]^2 \tag{5.8}$$

In eq. (5.8), $\hat{y}(x_i)$ is the predicted output corresponding to a given input sample, y_i is the observed output and n represents the total number of out of bag samples.

5.2.3 Results and Discussions

In random forest regression, the model is trained and validated using an integrated development environment (IDE) software named R [40]. The R computing structure is

organized in packages which are a combination of respective codes, data and documentation. In this work, the RFR is implemented using the 'randomForest' package. The applicability of RFR in this application is trained and validated by varying the number of trees. Table 5.2 represents the validation error criterion E_{avg} by varying number of trees. The equation for error criterion E_{avg} is the same as given in eq. (4.7).

Table 5.2: Performance of RFR Based on Validation Error (E_{avg}) by Varying the Number of Trees

No. of Trees	**CV Type**	**E_{avg}**
500	7-fold CV	4.50
	10-fold CV	4.45
1000	7-fold CV	4.51
	10-fold CV	4.45
2000	7-fold CV	4.52
	10-fold CV	4.45

From Table 5.2, it can be observed that the performance of the RFR did not improvise with varying number of trees.

Finally, the best models in all the interpolation techniques are compared with various performance measures as given in chapter 3.2. Table 5.3 represents the performance measure of various interpolation techniques discussed where neural network techniques are trained using Backpropagation algorithm. Table 5.4 represents the performance measure of various interpolation techniques discussed where neural network techniques are trained using Quasi-Newton algorithm.

Table 5.3: Comparison of Conventional Interpolation Techniques, Neural Network Techniques using Backpropagation Algorithm in Training, Support Vector Regression and Random Forest Regression

Techniques for interpolating radon concentration	**CV Type**	**E_{avg}**	**MAE**	**Fa_2**	**RMSE**	**FB**	**NMSE**
Kriging	7-fold CV	4.16	1.60	0.780	2.97	0.029	0.765
	10-fold CV	4.12	1.59	0.792	2.94	0.028	0.760
Local Polynomial Interpolation	7-fold CV	4.25	1.63	0.771	2.97	0.024	0.760
	10-fold CV	4.22	1.62	0.764	2.93	0.021	0.745
Global Polynomial Interpolation	7-fold CV	5.13	1.97	0.656	3.18	**0.011**	0.854
	10-fold CV	5.11	1.97	0.655	3.16	**0.008**	0.843
Radial Basis Function	7-fold CV	4.40	1.69	0.778	3.09	-0.003	0.804
	10-fold CV	4.65	1.67	0.776	3.05	-0.001	0.781
Artificial Neural Networks	7-fold CV	4.58	1.76	0.722	3.08	0.059	0.838
	10-fold CV	4.52	1.74	0.720	3.07	0.089	0.874
Prior Knowledge Input	7-fold CV	4.50	1.73	0.737	3.12	0.117	0.931
	10-fold CV	4.48	1.72	0.730	3.07	0.128	0.962
Source Difference Model	7-fold CV	4.48	1.72	0.741	3.06	0.075	0.864
	10-fold CV	4.66	1.79	0.699	3.10	0.134	0.967
Space-Mapped Neural Networks	7-fold CV	4.48	1.72	0.738	3.09	0.088	0.885
	10-fold CV	4.41	1.68	0.742	2.99	0.062	0.787
Correction Model	7-fold CV	**2.71**	**1.03**	**0.790**	**1.83**	-0.165	**0.245**
	10-fold CV	**2.57**	**0.99**	**0.818**	**2.20**	-0.141	**0.488**
Support vector Regression	7-fold CV	4.00	1.53	0.785	2.98	0.139	0.861
	10-fold CV	4.00	1.54	0.787	2.95	0.137	0.850
Random Forest regression	7-fold CV	4.50	1.73	0.767	3.09	-0.007	0.802
	10-fold CV	4.45	1.71	0.772	3.08	-0.008	0.790

Table 5.4: Comparison of Conventional Interpolation Techniques, Neural Network Techniques using Quasi-Newton Algorithm in Training, Support Vector Regression and Random Forest Regression

Techniques for interpolating radon concentration	**CVType**	**E_{avg}**	**MAE**	**Fa_2**	**RMSE**	**FB**	**NMSE**
Kriging	7-fold CV	4.16	1.60	0.780	2.97	0.029	0.765
	10-fold CV	4.12	1.59	0.792	2.94	0.028	0.760
Local Polynomial Interpolation	7-fold CV	4.25	1.63	0.771	2.97	0.024	0.760
	10-fold CV	4.22	1.62	0.764	2.93	0.021	0.745
Global Polynomial Interpolation	7-fold CV	5.13	1.97	0.656	3.18	0.011	0.854
	10-fold CV	5.11	1.97	0.655	3.16	0.008	0.843
Radial Basis Function	7-fold CV	4.40	1.69	0.778	3.09	-0.003	0.804
	10-fold CV	4.65	1.67	0.776	3.05	-0.001	0.781
Artificial Neural Networks	7-fold CV	4.34	1.66	0.761	3.00	0.003	0.767
	10-fold CV	4.36	1.67	0.763	2.96	-0.003	0.745
Prior Knowledge Input	7-fold CV	4.56	1.75	0.748	3.06	0.004	0.792
	10-fold CV	4.54	1.74	0.736	3.03	-0.005	0.780
Source Difference Model	7-fold CV	4.34	1.67	0.764	3.00	0.003	0.768
	10-fold CV	4.35	1.67	0.763	2.96	-0.003	0.745
Space-Mapped Neural Networks	7-fold CV	4.49	1.72	0.753	3.07	**0.001**	0.795
	10-fold CV	4.37	1.68	0.759	2.96	**0.004**	0.753
Correction Model	7-fold CV	**3.58**	**1.37**	**0.795**	**2.71**	-0.076	**0.579**
	10-fold CV	**3.63**	**1.40**	**0.784**	**2.63**	-0.087	**0.547**
Support vector Regression	7-fold CV	4.00	1.53	0.785	2.98	0.139	0.861
	10-fold CV	4.00	1.54	0.787	2.95	0.137	0.850
Random Forest regression	7-fold CV	4.50	1.73	0.767	3.09	-0.007	0.802
	10-fold CV	4.45	1.71	0.772	3.08	-0.008	0.790

From Tables 5.3 and 5.4, it can be observed that correction-based ANN model has better performance compared to other interpolation techniques. The Ideal Values (IV) of performance measures E_{avg}, MAE, RMSE and NMSE are zero, whereas for Fa_2, it is one. The FB lies between -2 and +2 and the ideal value of this evaluation parameter is zero.

In neural network approaches, it is observed that training by Quasi-Newton algorithm have accurate models than neural networks trained by Backpropagation algorithm.

In PKI approach, neural network trained using Backpropagation algorithm has slightly better validation error E_{avg} compared to neural network trained using Quasi-Newton algorithm. Such a behavior in neural networks can be attributed to over learning for the neural network trained using Quasi-Newton algorithm.

In correction-Based neural network approach, initially, it was observed that the Quasi-Newton algorithm performed better than the Backpropagation algorithm while training and validating the candidate correction models as evident from Tables 4.11 and 4.12. Hence, it can be concluded that Quasi-Newton predictions are more accurate than Backpropagation. It is also noted that, in CBNN the required attribute value (i.e., radon concentration in this case) prediction is based on a sensitivity based root finding algorithm, where the required attribute value is repetitively updated to predict the desired output in the candidate correction model such that the E_{obj} (as given in eq. (4.8)) is within the user defined threshold. In CBNN, implementation of the sensitivity-based root finding algorithm has lead to deviation of optimized Quasi-Newton predictions. While using the Back propagation, the predictions which are struck at the local minima are being optimized using the sensitivity approach and thus outperforming Quasi-Newton.

Further, considering neural networks using the Quasi-Newton algorithm have better results over Backpropagation, the best models in neural networks using the Quasi-Newton algorithm are compared with other interpolation techniques in Table 5.4. From Table 5.4, it can be concluded that Kriging and LPI have better results over GPI and RBF. From Table 5.4, it can be concluded that GPI and RBF did not perform better for data having short-range variations. In neural network techniques, ANN performance can be explained from the fact that CV requires averaging of performance measures from k distinct models, resulting in the deteriorated performance of ANN compared to conventional interpolation techniques, like Kriging and LPI. The performance of ANN modeling depends on training data and the number of hidden neurons, as cross-validation method requires averaging the performance of k distinct models for a fixed number of hidden neurons, it can be concluded that ANN did not perform better over conventional interpolation techniques using the CV methods. Though KBNNs did not perform better over conventional interpolation techniques like ANN using the cross-validation methods, the performance of KBNN's especially in the case of SDM and SMNN using the 10-fold CV method is slightly better over ANN's. The Correction-based ANN model has shown better performance in evaluating five of the six performance measures of all the interpolation techniques discussed. SVMs have proved to be better than ANNs [62], which is true in this application as evident from Table 5.4. Unlike ANNs and KBNNs, in the CBNN where the required attribute value is optimized using a sensitivity-based root finding algorithm, rather estimating the output based on MLP3 network thus, shows that the CBNN could perform better than SVR. Among the newly introduced regression techniques for this application, SVR has shown better performance over RFR. RF may be

better than SVM for interpolating environmental variables as in Li et al. [74], but in our application, unlike other SVM methods, nu-SVR methodology models the error bounds into the intervals of [0,1] (i.e., similar to scaling), thus leading to better performance of nu-SVR modeling over RFR. In general, RF predictions are more accurate for extrapolation than interpolation [73, 74], but the performance of RFR for interpolation in geo-statistical applications can be improved by using secondary variables as in [74]. However, the correction based ANN approach having better performance of all the interpolation techniques, SVR can be equally considered for various reasons such as requirement of less number of training samples, overcoming the problem of local minimization and over-fitting, as well as employing Structural Risk Minimization (SRM) unlike ANN's Empirical Risk Minimization (ERM) principle.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

This thesis presents the comparison between various conventional interpolation techniques, neural network approaches, SVR and RFR for modeling and predicting the radon concentrations in Ohio. Observing the results, it can be concluded that the Correction-Based ANN approach and SVR have delivered better performance compared to other techniques in predicting the radon concentrations, as shown in Table 5.4. Furthermore, the Correction-Based ANN model outperforms in five of the six evaluation parameters among all the interpolation techniques, while the SVR has given better performance of all the interpolation techniques except the Correction-Based ANN modeling, using both the K-fold cross-validation methods. Though Correction-Based ANN model outperforms SVR, owing to the advantage of SVR in solving the problem of over-fitting, this technique can be equally considered with Correction-Based ANN modeling.

6.2 Future Work

In this work, it is found that Correction-Based ANN modeling approach using local optimization algorithms such as Backpropagation and Quasi-Newton for training in the MLP3 network leads to the problem of deviations in predictions due to local minima. Hence, to overcome the problem of local minimization in CBNN, sensitivity-based root finding algorithm can be used to optimize the radon concentration predictions using global optimization algorithms such as Particle Swarm Optimization (PSO) or Genetic Algorithms (GA) in MLP3 network. In RFR, it was found that lack of secondary variables might decrease the performance of the model. Hence, identifying the secondary variables and modeling RFR could lead to better predictions.

References

[1] Hansen, M. (2008). GeoHazards on Radon. Available at: http:// www. dnr.state.oh.Us/ tabid/7906/default.aspx. Accessed 07 July 2013.

[2] Page, S. (1993). EPA'S strategy to reduce risk of radon. Journal of Environment and Health, 56(5), 27-36.

[3] U.S. Environmental Protection Agency. Information on Radon. Available at: http://www.epa. gov/radon/index.html. Accessed 19 June 2013.

[4] Harrell, J. A., Belsito, M. E., & Kumar, A. (1991). Radon hazards associated with outcrops of Ohio Shale in Ohio. Environmental Geology and Water Sciences, 18(1), 17-26.

[5] Kumar, A., Harrell, J. A., & Heydinger, A. G. (2001). Ohio goes online to combat indoor Radon. em-pittsburgh-air and waste management association-, 30-33.

[6] Harrell, J. A., McKenna, J. P., & Kumar, A. (1993). Geological controls on indoor radon in Ohio. Ohio Department of Natural Resources, Division of Geological Survey, 36.

[7] Heydinger, A. G., Kumar, A., & Harrell, J. A. (1991). Development of an indoor radon information system. Environmental Software, 6(4), 194-201.

[8] Kumar, A., Tandala, A., Kalapati, R. S., & Ghosh, S. (2003). Management of radon mitigation data in the state of Ohio. Environmental Progress, 22(3), O19-O24.

[9] Ojha, S., Thomas, S. J., & Kumar, A. (2001). Experience in integrating geographical information systems (GIS) to an indoor radon database. Environmental Progress, 20(3), O7-O11.

[10] Kumar, A., Sud, A., & Heydinger, A. (1998). Application of oracle 7.3 system for the development of an environmental database. Environmental progress, 17(3), F11-F14.

[11] Kumar, A., & Varadarajan, C. (2005). Development of Ohio radon information system. In Proceeding of Environmental Data Analysis–Assessing Health and Environmental Impacts, Developing Policy, and Achieving Regulatory Compliance Conference, Oak Brook, IL, Air & Waste Management Association.

[12] Kumar, A., Heydinger, A. G., & Harrell, J. A. (1990). Development of an indoor radon information system for Ohio and its application in the study of the geology of radon in Ohio. Ohio Air Quality Development Authority, Columbus, OH, 85-111.

[13] Joshi, A., Manne, G. K., & Kumar, A. (2002). Managing Ohio's radon data with MS access/SQL server 7.0. Environmental Progress, 21(4), D8-D12.

[14] Zone maps for Radon concentration in Ohio. Available at: http://www.eng.utoledo.edu/aprg/ radon/index.html. Accessed 25 June 2012.

[15] Kumar, A., Maroju, S., & Bhat, A. (2007). Application of ArcGIS geostatistical analyst for interpolating environmental data from observations. Environmental Progress, 26(3), 220-225.

[16] Akkala, A., Devabhaktuni, V., Kumar, A., & Bhatt, D. (2011). Development of

an ANN interpolation scheme for estimating missing radon concentrations in Ohio. Open Environmental & Biological Monitoring Journal, 4, 21-30.

[17] Akkala, A., Bhatt, D., Devabhaktuni, V., & Kumar, A. (2012). Knowledge-based neural network approaches for modeling and estimating radon concentrations. Environmental Progress & Sustainable Energy, 32, 355-364.

[18] Yerrabolu, P., Mareddy, L., Bhatt, D., Aggarwal, P., Kumar, A., & Devabhaktuni, V. (2012). Correction Model-Based ANN Modeling Approach for the Estimation of Radon Concentrations in Ohio. Environmental Progress & Sustainable Energy.

[19] Kumar, A., Kadiyala, A., Devabhaktuni, V., Akkala, A., Manthena, D.V. (2010). Examination of Ohio indoor radon data. American Association of Radon Scientists and Technologists (AARST) Proceedings of 22nd International Radon Symposium. Available at (www.aarst.org/proceedings/2010/06_examination_of_ohio_indoor_radon_data.pdf. (Accessed on August 15, 2012).

[20] Manthena, D.V., Kadiyala, A., Kumar, A. (2009). Interpolation of radon concentrations using GIS based kriging and cokriging techniques. Environ. Prog. Sustain. Energy, 28, 487-492.

[21] Manthena, D.V., Kumar, A., Kadiyala, A. 2012. Development of an approach to predict radon gas concentrations for unmeasured zip codes using GIS based interpolation techniques. In A.R. Baswell (Ed.), Advances in Mathematics Research. Volume 15 (pp. 47-76). New York: Nova Science Publishers Inc.

[22] Li, J. (2008). A review of spatial interpolation methods for environmental scientists (Vol. 137). Canberra: Geoscience Australia.

[23] Gotway, C. A., Ferguson, R. B., Hergert, G. W., & Peterson, T. A. (1996). Comparison of kriging and inverse-distance methods for mapping soil parameters. Soil Science Society of America Journal, 60(4), 1237-1247.

[24] Hudson, G., & Wackernagel, H. (1994). Mapping temperature using kriging with external drift: theory and an example from Scotland. International journal of Climatology, 14(1), 77-91.

[25] Goovaerts, P. (2000). Geostatistical approaches for incorporating elevation into the spatial interpolation of rainfall. Journal of hydrology, 228(1), 113-129.

[26] Apaydin, H., Sonmez, F. K., & Yildirim, Y. E. (2004). Spatial interpolation techniques for climate data in the GAP region in Turkey. Clim Res, 28(1), 31-40.

[27] Boken, V. K., Hoogenboom, G., Hook, J. E., Thomas, D. L., Guerra, L. C., & Harrison, K. A. (2004). Agricultural water use estimation using geospatial modeling and a geographic information system. Agricultural Water Management, 67(3), 185-199.

[28] Mohammadi, A. H., & Richon, D. (2007). Use of artificial neural networks for estimating water content of natural gases. Industrial & engineering chemistry research, 46(4), 1431-1438.

[29] Sun, G., Hoff, S. J., Zelle, B. C., & Nelson, M. A. (2008). Forecasting daily source air quality using multivariate statistical analysis and radial basis function networks. Journal of the Air & Waste Management Association, 58(12), 1571-

1578.

[30] Gardner, M. W., & Dorling, S. R. (1998). Artificial neural networks (the multilayer perceptron)--a review of applications in the atmospheric sciences.Atmospheric environment, 32(14-15), 2627-2636.

[31] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. Neural computation, 1(4), 541-551.

[32] Kolehmainen, M., Martikainen, H., & Ruuskanen, J. (2001). Neural networks and periodic components used in air quality forecasting. Atmospheric Environment, 35(5), 815-825.

[33] Reich, S. L., Gomez, D. R., & Dawidowski, L. E. (1999). Artificial neural network for the identification of unknown air pollution sources. Atmospheric Environment, 33(18), 3045-3052.

[34] Maier, H. R., & Dandy, G. C. (1996). The use of artificial neural networks for the prediction of water quality parameters. Water resources research, 32(4), 1013-1022.

[35] Singh, K. P., Basant, A., Malik, A., & Jain, G. (2009). Artificial neural network modeling of the river water quality—a case study. Ecological Modelling, 220(6), 888-895.

[36] Sousa, S. I. V., Martins, F. G., Alvim-Ferraz, M. C. M., & Pereira, M. C. (2007). Multiple linear regression and artificial neural networks based on principal components to predict ozone concentrations. Environmental Modelling & Software, 22(1), 97-103.

[37] Comrie, A. C. (1997). Comparing neural networks and regression models for ozone forecasting. Journal of the Air & Waste Management Association, 47(6), 653-663.

[38] Spellman, G. (1999). An application of artificial neural networks to the prediction of surface ozone concentrations in the United Kingdom. Applied Geography,19(2), 123-136.

[39] Wang, F., & Zhang, Q. J. (1997). Knowledge-based neural models for microwave design. Microwave Theory and Techniques, IEEE Transactions on, 45(12), 2333-2343.

[40] Watson, P. M., Gupta, K. C., & Mahajan, R. L. (1998, June). Development of knowledge based artificial neural network models for microwave components. In Microwave Symposium Digest, 1998 IEEE MTT-S International (Vol. 1, pp. 9-12). IEEE.

[41] Watson, P. M., & Gupta, K. C. (1996). EM-ANN models for microstrip vias and interconnects in dataset circuits. Microwave Theory and Techniques, IEEE Transactions on, 44(12), 2495-2503.

[42] Bandler, J. W., Ismail, M. A., Rayas-Sánchez, J. E., & Zhang, Q. J. (1999). Neuromodeling of microwave circuits exploiting space-mapping technology.Microwave Theory and Techniques, IEEE Transactions on, 47(12), 2417-2427.

[43] Bandler, J. W., Biernacki, R. M., Chen, S. H., Grobelny, P. A., & Hemmers, R. H. (1994). Space mapping technique for electromagnetic optimization.Microwave Theory and Techniques, IEEE Transactions on, 42(12),

2536-2544.

[44] Devabhaktuni, V., Mareddy, L., Vemuru, S., Cheruvu, V., Goykhman, Y., & Ozdemir, T. (2012). Sensitivity driven artificial neural network correction models for RF/microwave devices. International Journal of RF and Microwave Computer-Aided Engineering, 22(1), 30-40.

[45] MATLAB version 8.1 (2013) Natick, Massachusetts: The MathWorks Inc.

[46] Patel, V. C., & Kumar, A. (1998). Evaluation of three air dispersion models: ISCST2, ISCLT2, and SCREEN2 for mercury emissions in an urban area.Environmental Monitoring and Assessment, 53(2), 259-277.

[47] Gudivaka, V., & Kumar, A. (1990). An evaluation of four box models for instantaneous dense-gas releases. Journal of Hazardous Materials, 25(1), 237-255.

[48] Kumar, A., Luo, J., & Bennett, G. F. (1993). Statistical evaluation of lower flammability distance (LFD) using four hazardous release models. Process Safety Progress, 12(1), 1-11.

[49] Akkala, A., Devabhaktuni, V., & Kumar, A. (2010). Interpolation techniques and associated software for environmental data. Environmental progress & sustainable energy, 29(2), 134-141.

[50] Kangas, L. J., Keller, P. E., Hashem, S., Kouzes, R. T., & Allen, P. A. (1995, June). Adaptive life simulator: a novel approach to modeling the cardiovascular system. In American Control Conference, 1995. Proceedings of the (Vol. 1, pp. 796-800). IEEE.

[51] Zhang, Q. J. (1999). NeuroModeler. Department of Electronics, Carleton

University, 1125.

[52] Wang, F., & Zhang, Q. J. (1997). Knowledge-based neural models for microwave design. Microwave Theory and Techniques, IEEE Transactions on, 45(12), 2333-2343.

[53] Devabhaktuni, V. K., Chattaraj, B., Yagoub, M. C., & Zhang, Q. J. (2003). Advanced microwave modeling framework exploiting automatic model generation, knowledge neural networks, and space mapping. Microwave Theory and Techniques, IEEE Transactions on, 51(7), 1822-1833.

[54] Mendhurwar, K. A., Sundani, H., Khan, Z., Bhattacharya, P., & Devabhaktuni, V. (2009, December). Device modeling using correction model based ANN and support vector regression. In 12th international symposium on microwave and optical technology. New Delhi, India.

[55] Chang, C. C., & Lin, C. J. (2002). Training v-support vector regression: theory and algorithms. Neural Computation, 14(8), 1959-1977.

[56] Pirooznia, M., & Deng, Y. (2006). SVM Classifier–a comprehensive java interface for support vector machine classification of microarray data. BMC bioinformatics, 7(Suppl 4), S25.

[57] Mukkamala, S., Janoski, G., & Sung, A. (2002). Intrusion detection using neural networks and support vector machines. In Neural Networks, 2002. IJCNN'02. Proceedings of the 2002 International Joint Conference on (Vol. 2, pp. 1702-1707). IEEE.

[58] Huang, Z., Chen, H., Hsu, C. J., Chen, W. H., & Wu, S. (2004). Credit rating analysis with support vector machines and neural networks: a market

comparative study. Decision support systems, 37(4), 543-558.

[59] Byvatov, E., Fechner, U., Sadowski, J., & Schneider, G. (2003). Comparison of support vector machine and artificial neural network systems for drug/nondrug classification. Journal of Chemical Information and Computer Sciences, 43(6), 1882-1889.

[60] Moghaddamnia, A., Ghafari, M., Piri, J., & Han, D. (2008). Evaporation estimation using support vector machines technique. World Academy of Science, Engineering and Technology, 40, 14-22.

[61] Stamenkovic, J., Tuia, D., De Morsier, F., Borgeaud, M., & Thiran, J. P. (2013). Estimation of Soil Moisture from Airborne Hyperspectral Imagery with Support Vector Regression.

[62] Bhatt, D., Aggarwal, P., Bhattacharya, P., & Devabhaktuni, V. (2012). An enhanced mems error modeling approach based on nu-support vector regression. Sensors, 12(7), 9448-9466.

[63] Cortes, C., & Vapnik, V. (1995). Support-vector networks. Machine learning, 20(3), 273-297.

[64] Kuhn, H. W., & Tucker, A. W. (1951, July). Nonlinear programming. In Proceedings of the second Berkeley symposium on mathematical statistics and probability (Vol. 5), 481-492.

[65] Karush, W. (1939). Minima of functions of several variables with inequalities as side constraints (Doctoral dissertation, Master's thesis, Dept. of Mathematics, Univ. of Chicago).

[66] Keerthi, S. S., & Lin, C. J. (2003). Asymptotic behaviors of support vector

machines with Gaussian kernel. Neural computation, 15(7), 1667-1689.

[67] Chang, C. C., & Lin, C. J. (2011). LIBSVM: a library for support vector machines. ACM Transactions on Intelligent Systems and Technology (TIST), 2(3), 27.

[68] L. Breiman, Random forests, Machine learning 45(1) (2001) 5-32.

[69] P. Xu, F. Jelinek, Random forests in language modeling, In Proc. EMNLP (2004).

[70] H. Jiang, Y. Deng, H. S. Chen, L. Tao, Q. Sha, J. Chen, C. J. Tsai, S. Zhang. (2004) Joint analysis of two microarray gene-expression data sets to select lung adenocarcinoma marker genes, BMC bioinformatics 5(1) 81.

[71] Statnikov, A., Wang, L., & Aliferis, C. F. (2008). A comprehensive comparison of random forests and support vector machines for microarray-based cancer classification. BMC bioinformatics, 9(1), 319.

[72] D. R. Cutler, T. C. Edwards Jr, K. H. Beard, A. Cutler, K. T. Hess, J. Gibson, J. J. Lawler. (2007). Random forests for classification in ecology, Ecology 88(11) 2783-2792.

[73] S. Adusumilli, D. Bhatt, H. Wang, P. Bhattacharya, V. Devabhaktuni. (2013). A Low-Cost INS/GPS Integration Methodology based on Random Forest Regression, Expert Systems with Applications.

[74] Li, J., Heap, A. D., Potter, A., & Daniell, J. J. (2011). Application of machine learning methods to spatial interpolation of environmental variables. Environmental Modelling & Software, 26(12), 1647-1659.

Appendix-A

Source Code

A.1 Source code for K-fold Cross-Validation Data Preparation

```
clc;
clear;
filename = uigetfile('*.csv', 'select file for input');
x = load(filename);
folds = input('Enter the number of folds:\n');
fname = input('Enter the file name to save the sets:\n','s');
[row,columns]=size(x);
rows=row;
cols=columns;
sh=0;
c = cvpartition(rows,'kfold',folds);
for i=1:folds
   tr = training(c,i);
   x=cat(2,x,tr);
end
t=cols+1;
for se=1:folds
   p=0;q=0;
for i=1:rows
```

```
    if x(i,t)==0
        q=q+1;
    else
        p=p+1;
    end
end
y=zeros(p,cols);
z=zeros(q,cols);
j=1;l=1;m=1;r=1;s=1;
for i=1:rows
    if x(i,t)==1
        for j=1:cols;
       y(l,m)=x(i,j);
       m=m+1;
        end
              m=1; l=l+1;
    else
        for k=1:cols
           z(r,s)=x(i,k);
           s=s+1;
        end
               s=1; r=r+1;
           end
  end
    sh=sh+1;
  xlswrite(fname,y,sh)
  sh=sh+1;
  xlswrite(fname,z,sh)
    t=t+1;
end
```

A.2 Source code for Correction-Based ANN modeling approach

```
clc;
clear all;
close all;
n = 1;
p = 0;
q=3;
train=input('enter the number of training samples');
test=input('enter the number of testing samples');
total=test+train;
h=input('enter the number of hidden neurons');
for k=1:test
file1 = fopen('data.dat');
input = fscanf(file1, '%g',train);
input1 = input(n); % Latitude
input2 = input(n+1); %Longitude
outputp = input(n+2); % Radon
inputv = [input1 input2];
fclose(file1);
%Initial radon predictions
x=fopen('plot.dat');
y=fscanf(x, '%g',total);
yout=y(q);
file2 = fopen('trad.txt','a');
fprintf(file2, '%f\n',yout);
fclose(file2);
input1c = yout; % Radon
input2c = input2; % Longitude
inputc = [input1c input2c];
```

```
outputpc = model(inputc);
file3 = fopen('mod.txt','a');
fprintf(file3, '%f\n',outputpc);
fclose(file3);
% Difference between original and predicted latitude values
Diff = outputpc-input1;
file4 = fopen('error.txt','a');
fprintf(file4, '%f\n',Diff);
fclose(file4);
%input scaling
x(1)=-1.0+(2.0)*(input1c-(0.1)) / ((39.0) - (0.1));
x(2)=-1.0+(2.0)*(input2c-(-84.7905)) / ((-80.53069) - (-84.7905));
%calculating hidden neurons
z(1) = 1.0 / ( 1.0 + exp(-1.0 * (-0.444642+x(1)*(2.31087)+x(2)*(-0.345894))));
z(2) = 1.0 / ( 1.0 + exp(-1.0 * (-0.556844+x(1)*(-1.39897)+x(2)*(-1.63432))));
z(3) = 1.0 / ( 1.0 + exp(-1.0 * (5.903+x(1)*(8.59608)+x(2)*(6.28252))));
z(4) = 1.0 / ( 1.0 + exp(-1.0 * (-2.78883+x(1)*(-1.57637)+x(2)*(-1.48526))));
z(5) = 1.0 / ( 1.0 + exp(-1.0 * (-3.72274+x(1)*(-8.03895)+x(2)*(6.36265))));
z(6) = 1.0 / ( 1.0 + exp(-1.0 * (-1.74254+x(1)*(-3.49334)+x(2)*(-10.7203))));
z(7) = 1.0 / ( 1.0 + exp(-1.0 * (1.76763+x(1)*(2.7054)+x(2)*(4.72469))));
z(8) = 1.0 / ( 1.0 + exp(-1.0 * (0.433244+x(1)*(-2.72137)+x(2)*(-4.34913))));
z(9) = 1.0 / ( 1.0 + exp(-1.0 * (-0.267094+x(1)*(2.03351)+x(2)*(-4.69041))));
z(10) = 1.0 / ( 1.0 + exp(-1.0 * (0.818484+x(1)*(1.06151)+x(2)*(4.12581))));
z(11) = 1.0 / ( 1.0 + exp(-1.0 * (0.64654+x(1)*(3.00522)+x(2)*(-0.971178))));
z(12) = 1.0 / ( 1.0 + exp(-1.0 * (6.5286+x(1)*(0.85773)+x(2)*(-4.79507))));
z(13) = 1.0 / ( 1.0 + exp(-1.0 * (3.48275+x(1)*(7.69355)+x(2)*(-2.0921))));
z(14) = 1.0 / ( 1.0 + exp(-1.0 * (3.43261+x(1)*(1.9282)+x(2)*(8.24004))));
z(15) = 1.0 / ( 1.0 + exp(-1.0 * (0.245269+x(1)*(2.45342)+x(2)*(-3.30095))));
z(16) = 1.0 / ( 1.0 + exp(-1.0 * (-5.60611+x(1)*(-7.74941)+x(2)*(-6.42479))));
z(17) = 1.0 / ( 1.0 + exp(-1.0 * (3.83942+x(1)*(-0.6754)+x(2)*(1.81926))));
z(18) = 1.0 / ( 1.0 + exp(-1.0 * (-1.6599+x(1)*(-2.58227)+x(2)*(2.06372))));
```

```
z(19) = 1.0 / ( 1.0 + exp(-1.0 * (-3.40633+x(1)*(-0.376907)+x(2)*(-4.09953))));
z(20) = 1.0 / ( 1.0 + exp(-1.0 * (3.53675+x(1)*(8.10244)+x(2)*(-5.87667))));

%calculating output neurons
y(1) = 0.882239+z(1)*(0.261229)+z(2)*(0.688574)+z(3)*(-3.8392)+z(4)*(-
0.0172129)+z(5)*(3.49482)+z(6)*(1.83517)+z(7)*(5.56405)+z(8)*(1.42946)+z(9)*(-
0.153877)+z(10)*(-0.0956302)+z(11)*(0.853805)+z(12)*(-2.67437)+z(13)*(-
0.42149)+z(14)*(-2.82694)+z(15)*(-3.05137)+z(16)*(-
3.63576)+z(17)*(0.893393)+z(18)*(-1.86511)+z(19)*(0.800741)+z(20)*(3.48717);
%output scaling
output(1) = 38.438617+(y(1)-(0.0))*((41.934916) - (38.438617))/((1.0)-(0.0));

bw = [2.31087   -1.39897    8.59608 -1.57637    -8.03895    -3.49334    2.7054  -2.72137
2.03351 1.06151 3.00522 0.85773 7.69355 1.9282  2.45342 -7.74941    -0.6754 -2.58227
-0.376907   8.10244];
dw = [0.261229  0.688574    -3.8392 -0.0172129  3.49482 1.83517 5.56405 1.42946 -
0.153877   -0.0956302  0.853805    -2.67437    -0.42149    -2.82694    -3.05137    -
3.63576    0.893393    -1.86511    0.800741    3.48717];
ew= 0.882239;
sensitivity_in = 0.0;

% Calculating sensitivity based on chain rule
for i = 1:h
        sensitivity_in = sensitivity_in + dw(i)*z(i)*(1-z(i))*bw(i);
end
sensitivity_in2 = sensitivity_in * (((39.0) - (0.1))/2.0);
sensitivity_final = sensitivity_in2 * ((41.934916) - (38.438617));
S = sensitivity_final;
alpha = 0.1;
Euser = 0.1;iteration = 0; yout1 = yout;
```

```
% Defining error
Eobj = ((input1-outputpc)/input1)*100;
while(abs(Eobj)>Euser)
delta_out = abs(-Diff/S);
% Updating the Radon value using sensitivity value
if(yout<outputp)
yout = yout1 + (alpha * delta_out);
elseif(yout>outputp)
yout = yout1 - (alpha * delta_out);
end
yout1 = yout;
if(yout>outputp)
break;
end
input = [yout input2];
outputpc2 = model(input);
Eobj = ((input1-outputpc2)/input1)*100;
R = outputpc2 -input1;
iteration = iteration + 1;
if (iteration > 1000)
p= p+1;
break;
end
end
yout;
file5 = fopen('pred.txt','a');
fprintf(file5, '%f\n',yout);
fclose(file5);
n = n+3;
q=q+4;
end
```

A.3 Source code for nu-SVR using *LIBSVM* package

```
clear all;
close all;
clc
SPECTF = csvread('training.csv');
labels1 = SPECTF(:, 5);
features = SPECTF(:, 2:3);
features_sparse = sparse(features);
libsvmwrite('rad_train', labels1, features_sparse);
[label_vector4, instance_matrix4] = libsvmread('rad_train');
model_rad = svmtrain(label_vector4, instance_matrix4, '-h 0 -s 4 -t 2 -g 10');
SPECTF = csvread('testing.csv');
labels = SPECTF(:, 5);
features = SPECTF(:, 2:3);
features_sparse = sparse(features);
libsvmwrite('rad_test', labels, features_sparse);
[label_vector, instance_matrix] = libsvmread('rad_test');
output = svmpredict(label_vector, instance_matrix, model_rad);
x=[output labels];
csvwrite('pred.csv', x);
```

A.4 *R* Software commands for executing 'randomForest' package

```
training_data<-read.table("training.csv", header=TRUE,sep=",")
testing_data<-read.table("testing.csv", header=TRUE,sep=",")
train_inp<-subset(training_data,select=c(y2,y3))
train_out<-subset(training_data,select=c(y5))
test_inp<-subset(testing_data,select=c(y2,y3))
test_out<-subset(testing_data,select=c(y5))
rf<-
randomForest(train_inp,train_out[,1],test_inp,test_out[,1],keep.forest=TRUE,ntree=500)
pred<-predict(rf,test_inp)
y<-data.frame(pred,test_out)
write.table(y,file="pred.csv",sep=",",row.names=pred)
```

Appendix-B

Screenshots of the *Neuromodeler* Software

B.1 *Neuromodeler* software Main Window

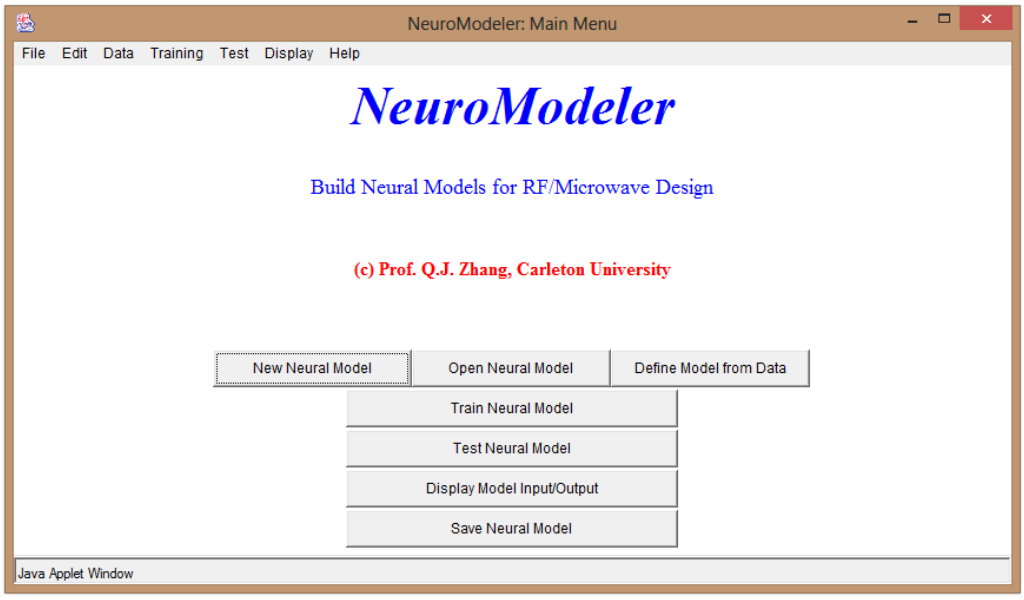

Figure B-1: Screenshot of the main window of the *Neuromodeler* software

B.2 New Model Window in the *Neuromodeler* Software

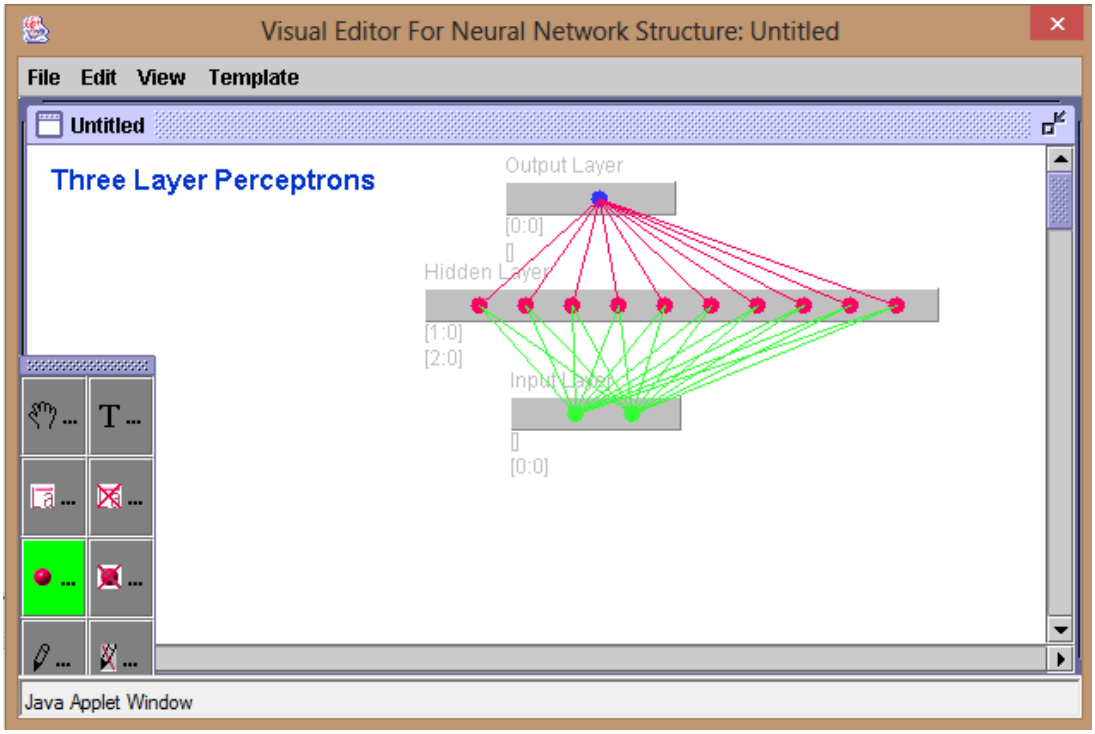

Figure B-2: Screenshot of the new neural model creation window in the *Neuromodeler* software

B.3 Training Window of the *Neuromodeler* software

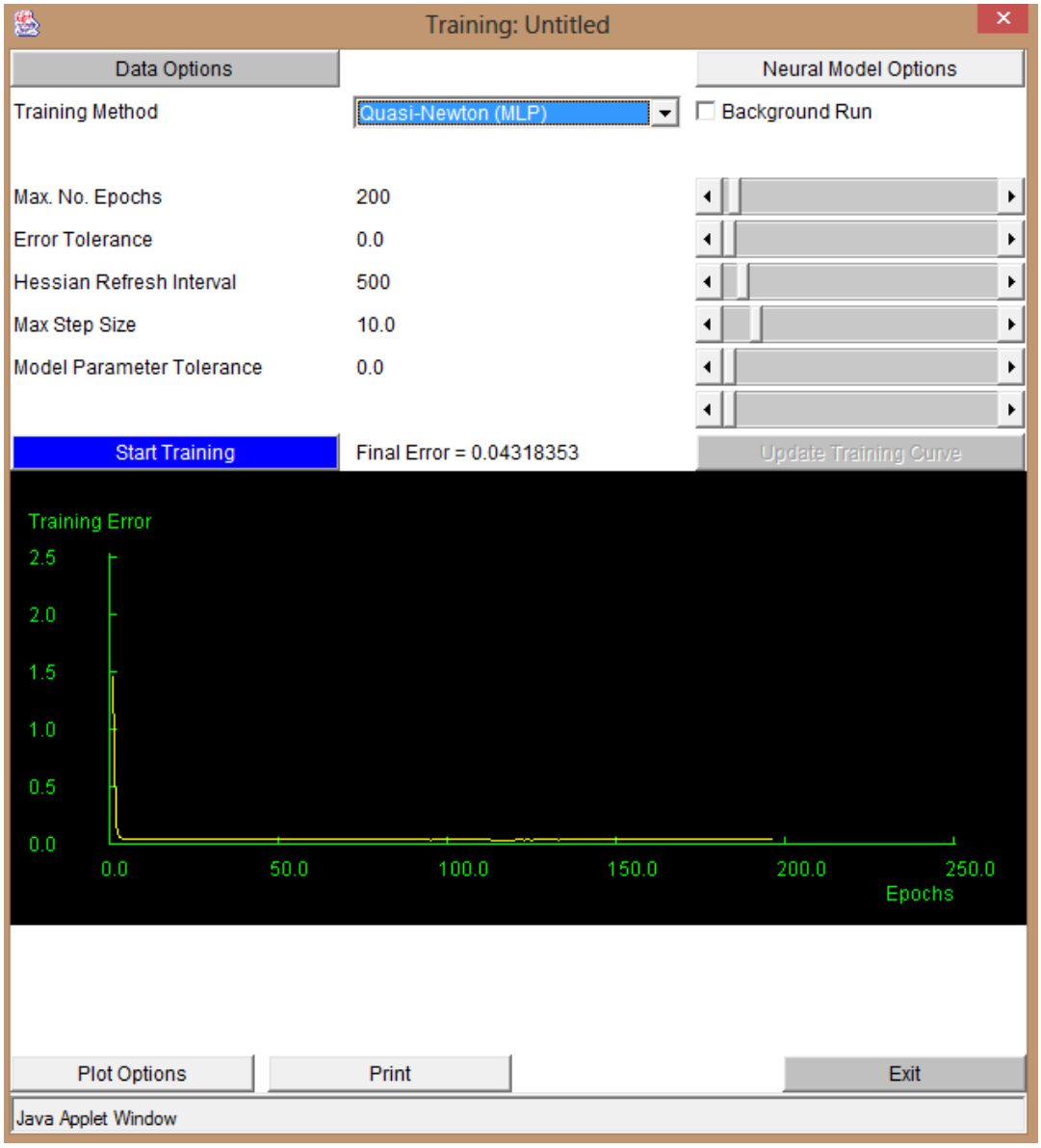

Figure B-3: Screenshot of the training window of the *Neuromodeler* software

B.4 Window for modifying hidden neurons in the *Neuromodeler* software

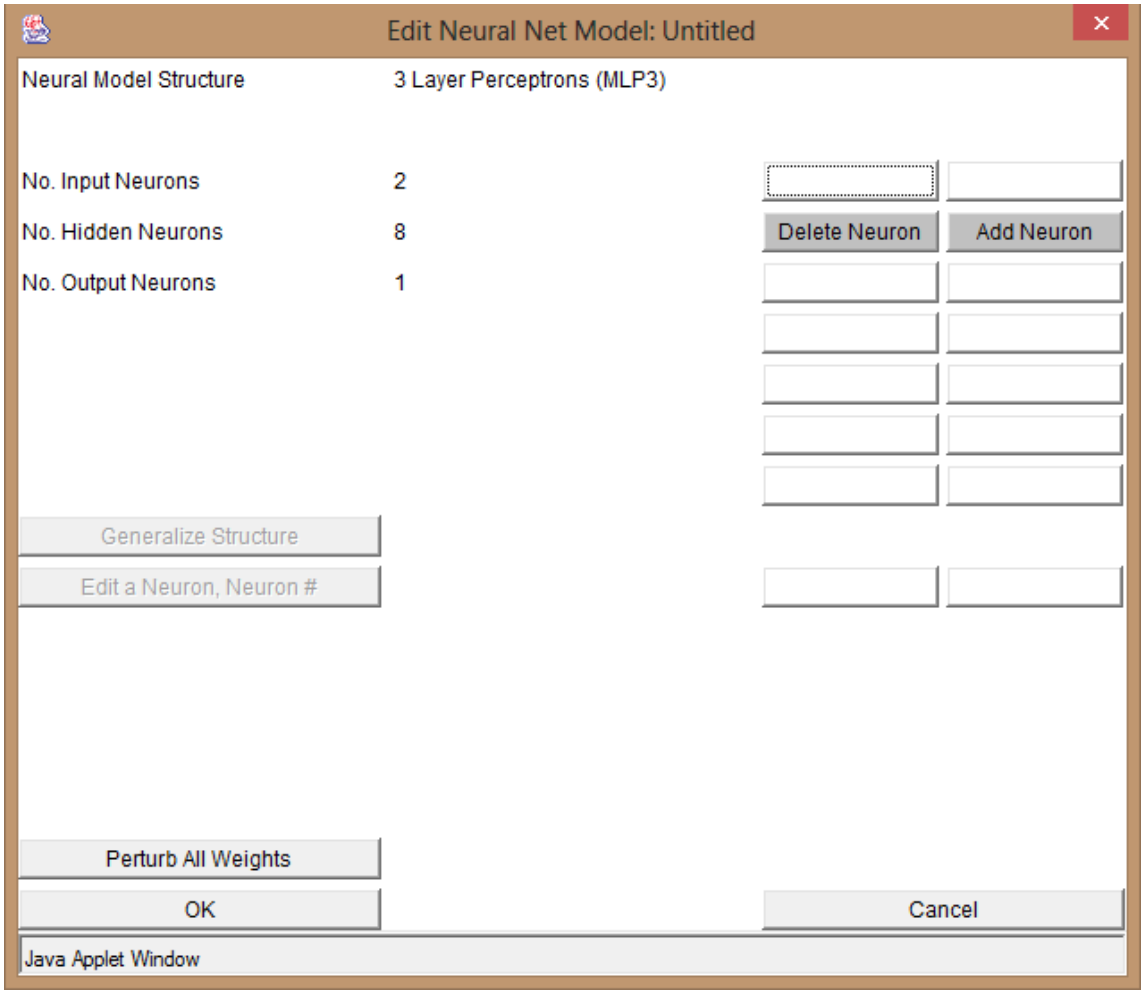

Figure B-4: Screenshot for modifying hidden neurons in the N*euromodeler* software

B.5 Testing Window of the *Neuromodeler* Software

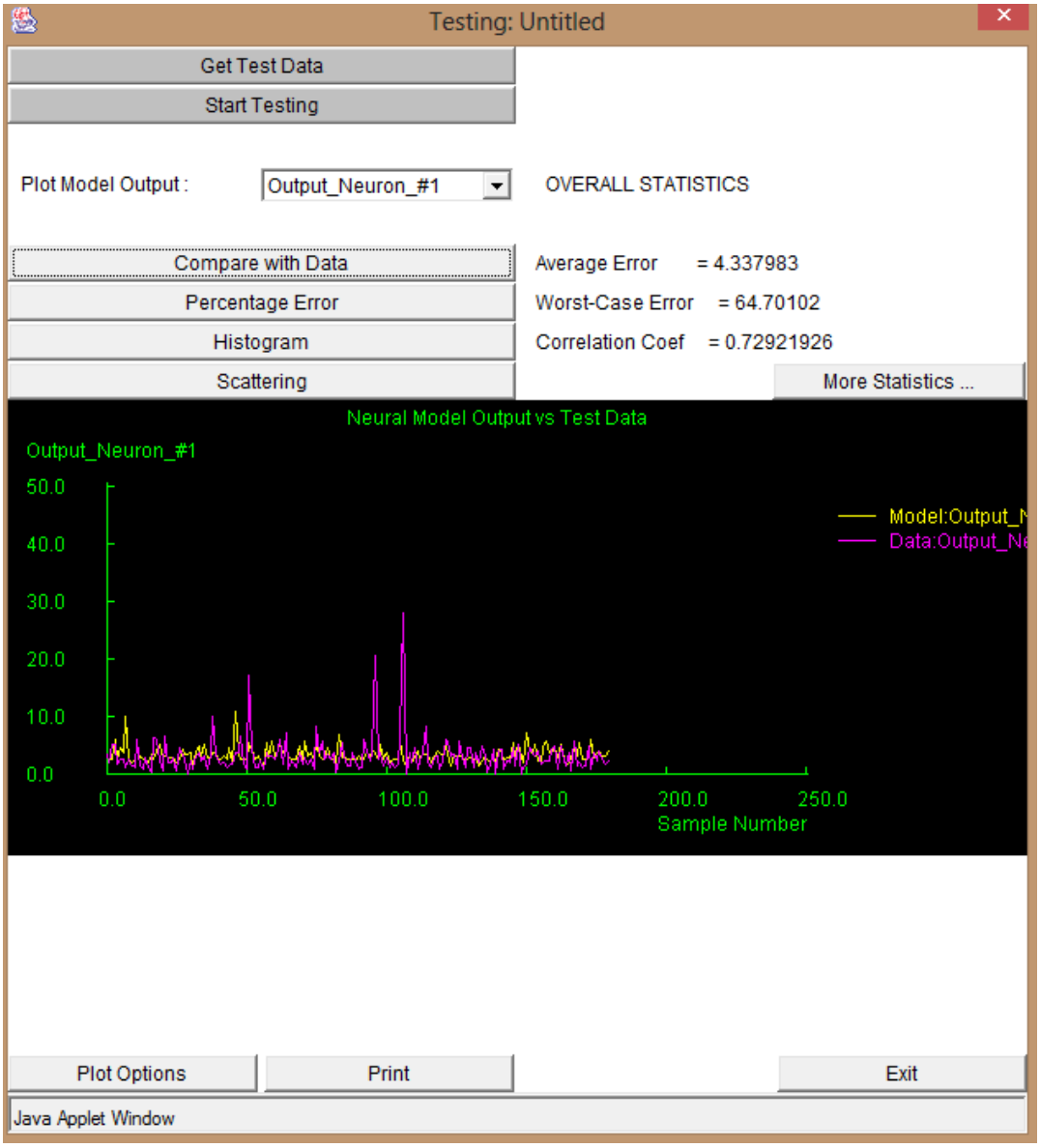

Figure B-5: Screenshot of the testing window of the *Neuromodeler* software

B.6 Export model to different platforms in the *Neuromodeler* Software

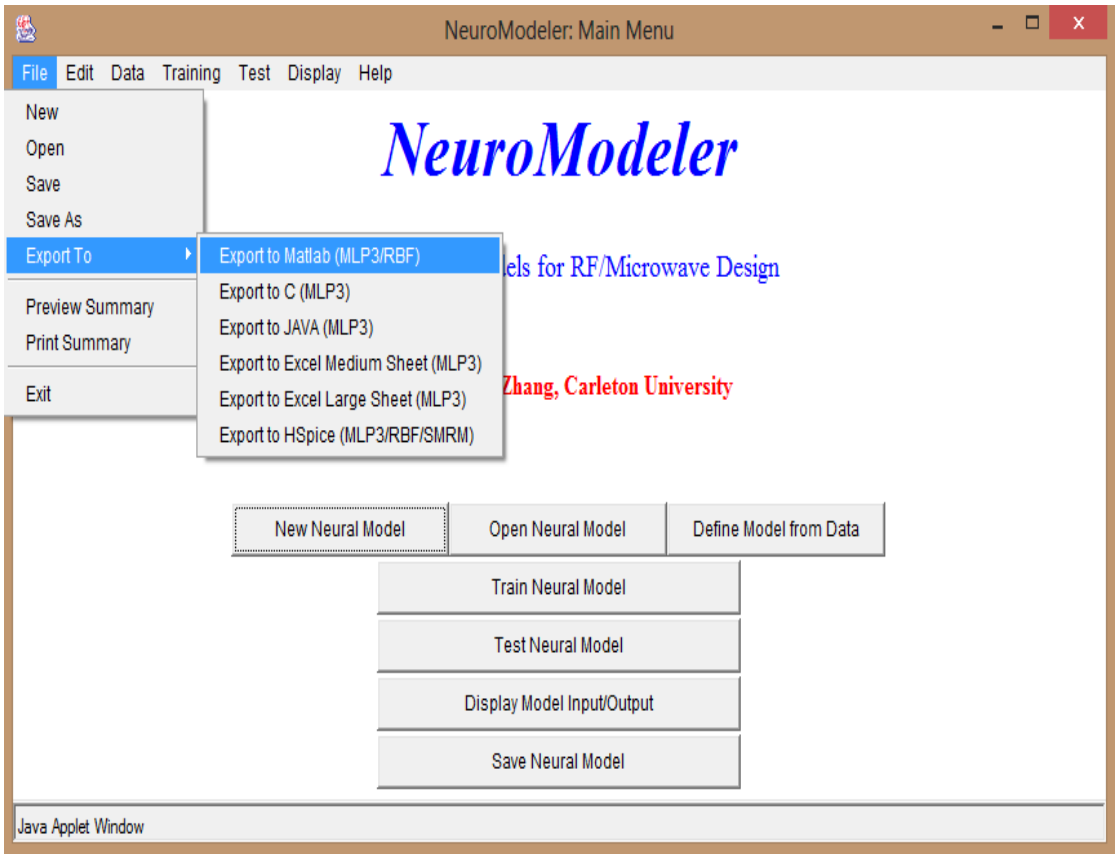

Figure B-6: Screen shot to export model from *Neuromodeler* software