A Thesis

entitled

Development of Artificial Neural Networks Based Interpolation

Techniques for the Modeling and Estimation of Radon

Concentrations in Ohio

by

Arjun Akkala

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

_______________________________________
Dr. Vijay Devabhaktuni, Advisor

_______________________________________
Dr. Ashok Kumar, Co-Advisor

_______________________________________
Dr. Mohammed Niamat, Committee Member

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

The University of Toledo

August 2010
An Abstract of

Development of Artificial Neural Networks based Interpolation Techniques for the Modeling and Estimation of Radon Concentrations in Ohio

by

Arjun Akkala

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

The University of Toledo
August 2010

Radon is a chemically inert, naturally occurring radioactive gas. It is one of the main causes of lung cancer second to smoking, and accounts for about 25,000 deaths every year in the US alone according to the National Cancer Institute. In order to initiate preventative measures to reduce the deaths caused by radon inhalation, it is helpful to have radon concentration data for each locality, e.g. zip code. However, such data are not available for every zip code in Ohio, owing to several reasons including inapproachability. In places where data is unavailable, radon concentrations must be estimated using interpolation techniques to take appropriate preventive measures against cancer.

This thesis proposes new interpolation techniques based on Artificial Neural Networks utilizing the available knowledge in terms of Radon concentration data and Uranium concentration data for modeling and predicting Radon concentrations in Ohio, US. Several models were first trained and then validated using available data to identify the best model for each technique. Model accuracies using the proposed approaches were proven to be significantly better in comparison to conventional interpolation techniques such as Kriging and Radial Basis Functions.
For my parents, fiancé, and friends
Acknowledgements

I wish to express my deepest gratitude to my advisor Dr. Vijay Devabhaktuni and co-advisor Dr. Ashok Kumar for their continued support, guidance and encouragement, without which it would not have been possible to succeed in my research. The financial support from the EECS Department and the Department of Civil Engineering in the form of a graduate/research assistantship is gratefully acknowledged. I would like to thank my colleague Kranthi Mogireddy for help with repetitive NeuroModeler execution for training and validating the ANN models. I am grateful to the assistance from Akhil Kadiyala and Dilip Manthena, in terms of providing the Radon data from the Ohio Radon Information System maintained for the ODH. The data collection has been supported by the ODH/USEPA and Ohio Air Quality Development Authority for the past 21 years.

Special thanks to all my friends for their enthusiastic and generous support through my research study.

Above all, I wish to express my deep appreciation to my family, fiancé, and friends who with their love and encouragement have made all this possible.
# Contents

Abstract ................................................................. iii

Acknowledgements ................................................. v

Contents ................................................................. vi

List of Tables ................................................................ ix

List of Figures ........................................................ xi

List of Abbreviations ................................................. xiii

1 Introduction .......................................................... 1

1.1 Problem Statement ............................................... 1

1.2 Research Approach ................................................ 3

1.3 Organization ......................................................... 4

2 Literature Review .................................................. 6

2.1 Principle of Interpolation ........................................ 6

2.2 Prior Art ............................................................... 8

2.3 Review of Artificial Neural Networks ..................... 14

3 Artificial Neural Networks Approach ..................... 17

3.1 Data Preparation ................................................... 21

3.2 Typical ANN Error Measures ............................... 23

3.3 Comparative Error Measures for Evaluating Spatial Interpolation Techniques 25
B.4 Testing Window of the *Neuromodeler* Software. ..................................................... 66
List of Tables

2-1 A list of available interpolation techniques commonly used in environmental studies...7

3-1 An illustrative portion of the University of Toledo radon database containing radon concentrations in addition to other relevant information. GM denotes geometric mean. NA indicates non-availability.................................................................22

3-2 Training and validation errors of ANN models with varying number of hidden neurons trained using the Backpropagation algorithm. Available data is divided into two sets, i.e. 90% training data and 10% validation data.................................................................28

3-3 Training and validation errors of ANN models with varying percentage of training data trained using the Backpropagation algorithm. Based on Table 3.2, the number of hidden neurons is fixed to be 90.................................................................28

3-4 Training and validation errors of ANN models with varying number of hidden neurons trained using the quasi-Newton algorithm. Available data is divided into two sets, i.e. 90% training data and 10% validation data.................................................................29

3-5 Training and validation errors of ANN models with varying percentage of training data trained using the quasi-Newton algorithm. Based on Table 3.4, the number of hidden neurons is fixed to be 110.................................................................29

4-1 Training and validation errors of the PKI model with varying number of hidden neurons. .......................................................................................................................43
4-2 Training and validation errors of the coarse model used in the SDM technique with varying number of hidden neurons. .................................................................45

4-3 Training and validation errors of the difference model used in the SDM technique and the overall SDM method. The number of hidden neurons for the coarse model is fixed as 110. .................................................................46

4-4 Training and validation errors of the SM model used in SMNN technique with varying number of hidden neurons. .................................................................47

4-5 Training and validation errors of the fine model used in SMNN technique with varying number of hidden neurons. The number of hidden neurons in the space mapping model is fixed to be 50. .................................................................47

4-6 Performance of the ANN and KBNN models developed in comparison with the conventional interpolation techniques. .................................................................48
List of Figures

1-1 Map of Ohio showing the geometric mean (GM) of radon concentrations (http://aprg.utoledo.edu/radon/). .................................................................3

2-1 Estimated radon concentrations obtained for Ohio by using four conventional interpolation techniques. ...........................................................................14

3-1 Proposed 3-layer MLP architecture for modeling radon concentrations. The network has two inputs (latitude and longitude), one output (radon concentration), and $b$ hidden neurons. .................................................................................................19

3-2 A screen-shot of the Neuromodeler software showing the validation data (magenta line) and the radon concentration values as predicted by the neural model (yellow line). ..................................................30

4-1 Decay series of uranium with daughter products and half-lives, showing the formation of radon from uranium. This chain helps us in understanding dependency of radon concentrations on those of uranium. .................................................................31

4-2 Radiometric map of Ohio showing the uranium concentration in soil (Source: http://radon.utoledo.edu). .............................................................................32

4-3 Structure of a Knowledge Based Neural Network, which embeds additional knowledge into the neural network, as a part of the training process. .................................................................33

4-4 The structure of a PKI method. The knowledge is used as an additional input to the model. ...........................................................................................................34
4-5 The principle of SDM method. The difference between the coarse model and the expected output is modeled for fine-tuning the estimations. .............................................35

4-6 The structure of a SMNN model has two neural models (space mapping model and fine model) where the output of the space mapping model is used as the input for the fine model..................................................................................................................................35

4-7 PKI method using 3 layer MLP network. .............................................................................37

4-8 The structure of the SDM technique used to train the neural model in two stages. ......40

4-9 Training approach in the SMNN method. Intermediate inputs are used between the input and the output, which map the actual inputs to the output using additional knowledge..................................................................................................................................41

A-1 Screenshot of the main window of the Neuromodeler software...........................................63

A-2 Screenshot of the new neural model creation window in the Neuromodeler software.....64

A-3 Screenshot of the training window of the Neuromodeler software, while training the fine model of the SMNN technique...............................................................................................65

A-4 Screenshot of the testing window of the Neuromodeler software, validating the fine model of the SMNN technique...............................................................................................66
List of Abbreviations

ANN...... Artificial Neural Network
Fa2....... Fraction of Two
FB......... Fractional Bias
GM......... Geometric Mean
GPI......... Global Polynomial Interpolation
IDW......... Inverse Distance Weighting
KBNN....... Knowledge Based Neural Network
LPI......... Local Polynomial Interpolation
MAE......... Mean Average Error
MLP......... Multi Layer Perceptron
NMSE....... Normalized Mean Square Error
PKI......... Prior Knowledge Input
RBF......... Radial Basis Function
RMSE....... Root Mean Square Error
SDM......... Source Difference Model
SM......... Space Mapping
SMNN........ Space Mapped Neural Network
ODH....... Ohio Department of Health
ORIS....... Ohio Radon Information System
US......... United States
USEPA...... United States Environment Protection Agency
Chapter 1

Introduction

1.1 Problem Statement

Radon, which is an invisible, colorless, odorless gas, is a daughter element in the radioactive decay series of uranium. Uranium is widespread in small quantities in rocks and sediments. Both radon and its decay products are radioactive. Radon can cause lung cancer in people exposed to high levels over a long period of time (Price and Gelman 2006), a health issue that many homeowners unknowingly face. Radon is responsible for about 25,000 lung cancer deaths every year in the US. About 2,900 of these deaths occur among people who never smoked (US Geological Survey 2009). Radon is classified as a Class A carcinogen by the US Environmental Protection Agency (USEPA). The USEPA and other organizations have launched research efforts to help assess risks and remedial options. In this context, some of the questions worth investigating include: (i) What is the statistical and spatial distribution of indoor radon; (ii) What methods can be used to reduce radon concentrations in homes; (iii) What is the risk as a function of exposure; etc. There have been ongoing efforts, including those at The University of Toledo, in terms of maintaining radon concentration databases for states with high radon levels, e.g. Ohio.
Although Ohio’s radon concentrations are not as high as those in some other states, they are well above the national average. With an objective of providing a healthy living environment, the USEPA continues to support preventive actions for all homes with higher radon activity. For instance, Ohio Department of Health (ODH) runs a campaign aimed at measuring radon concentrations across Ohio. Health authorities, in conjunction with county health departments, commercial testing services, and university researchers have so far gathered information for more than 130,000 homes/schools across Ohio (Harrell et al. 1991; Harrell et al. 1993; Heydinger et al. 1991; Kumar et al. 2001; Kumar and Varadarajan 2005). Data management has been carried out using different database management systems (Kumar et al. 1990; Kumar et al. 1998; Ojha et al. 2001; Joshi et al. 2002; Kumar et al. 2003).

Such data is available for many zip codes, but however, not for all of them owing to certain reasons including inapproachability. For such places, we need to estimate the concentration values using interpolation techniques based on the available data. Figure 1.1 shows the Radon concentrations in Ohio. The regions shown in white color are the ones that have no available Radon concentration data, for which interpolation techniques need to be used. The current database has Radon concentrations available for 1262 zip codes out of 1492 zip codes in Ohio.
1.2 Proposed Research Approach

In the first phase of this thesis, a new Artificial Neural Network (ANN) based scheme for modeling and predicting radon concentrations is proposed. The neural networks employed in this work are 3-layer multi-layer perceptrons often referred to as 3-layer MLP or simply MLP3.

There is a growing need for reduction in the cost of model development and improvement in model reliability. In the first phase, we used the ANN approach to model and predict the Radon concentrations. This approach is shown to be better than the classical geometric mean (GM) of radon concentrations.
interpolation techniques, such as Kriging, Radial Basis Functions (RBF). This was the first time, the ANN approach was used to model Radon concentrations. This work used only the available Radon concentration data.

The accuracy of the commonly used MLP model largely depends on adequacy of the training data, apart from the number of hidden neurons. Since MLP belongs to the type of black-box models structurally embedding no problem-dependent information/knowledge, it derives the entire information about the relationship between the parameters from the training data. Consequently, a large amount of training data is needed to ensure model accuracy. In environmental monitoring, training data is obtained through measurements at points in the site of interest. Obtaining a huge training dataset of environmental data could be very expensive, because measurements have to be performed at many geographic locations (e.g., pollutant concentrations for various zip codes in a state/county). Even with sufficient amount of training data, the reliability of MLP, when used for extrapolation, is not guaranteed and in some cases is very poor. Hence, a new technique named Knowledge Based Neural Networks (KBNNs) is used in the second phase of the thesis. Along with the Radon concentrations, the ODH has taken additional efforts to measure the Uranium concentrations as well; and this data is available for 1013 out of the 1492 zip codes. Additional knowledge was identified in the form of Uranium concentration data apart from the available Radon concentration data, which is used in the KBNN models to improve the reliability of the estimations.

1.3 Organization

The remainder of the document is organized as follows. Chapter 2 discusses the literature review. Chapter 3 gives a detail introduction to ANNs, data collection and
preparation, various comparative measures for evaluating spatial interpolation techniques, and the results of such comparisons. Chapter 4 gives an overview of the various KBNN models, the details of implementation, and the results comparison of the KBNN techniques with the conventional techniques and the previous MLP approach. Finally, Chapter 5 gives the conclusions of the work done and future work.
Chapter 2

Literature Review

2.1 Principle of Interpolation

Interpolation is a method or mathematical function that estimates the values at locations where no measured values are available. Interpolation can be as simple as a number line, however, most environmental and geographic information science research involves spatial data. Spatial interpolation assumes that the attribute data are continuous over space, allowing for the estimation of the attribute at any location within the data boundary. Another assumption is that the attribute is spatially dependent, indicating the values closer together are more likely to be similar than the values farther apart. These assumptions allow for the spatial interpolation methods to be formulated. Spatial interpolation is widely used for creating continuous data when data are collected at discrete locations (i.e. grids/points). For example, Radon concentration data provided by the ODH are collected from testing laboratories for different homes that are located in the state. An interpolation method is used to create concentration maps covering the entire state of Ohio. These point data are displayed as interpolated surfaces for qualitative interpretation. In addition to qualitative research, these interpolated surfaces can also be used in quantitative research towards reduction of cancer caused by Radon inhalation.
When an interpolated surface is used as part of a larger research project, both the method and accuracy of the interpolation technique become critical and need to match the requirements (Akkala et al. 2010). The goal of spatial interpolation is to create a surface that is intended to best represent empirical reality. Therefore, the method selected must be assessed for accuracy for larger studies.

There are a wide variety of interpolation techniques that are used in real time applications such as environmental studies. A list of available interpolation techniques, their principle of working, some advantages, and disadvantages are given in Table 2.1.

Table 2.1: A list of available interpolation techniques commonly used in environmental studies.

<table>
<thead>
<tr>
<th>Interpolation Technique</th>
<th>Principle</th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nearest Neighbor and Thiessen Polygons</td>
<td>Selection of value at closest data point</td>
<td>Ease of use</td>
<td>Inaccurate in less densely sampled scenarios</td>
</tr>
<tr>
<td>Triangulated Irregular Network</td>
<td>Set of conterminous triangles associated with a mass factor is used to define the space</td>
<td>Ability to describe the surface at different levels of resolution</td>
<td>In most cases, requires visual inspection and manual control of the network</td>
</tr>
<tr>
<td>Polynomial Regression</td>
<td>Fits the variable of interest to a linear combination of regressor variables</td>
<td>Simple model</td>
<td>Model has poor ability to predict outside the range of data points</td>
</tr>
<tr>
<td>Global Polynomial Interpolation (GPI)</td>
<td>Works by capturing coarse-scale patterns in the data, and fitting a polynomial</td>
<td>Computationally less intensive</td>
<td>Estimation errors increase exponentially with increasing complexity</td>
</tr>
<tr>
<td>Local Polynomial Interpolation (LPI)</td>
<td>Similar to GPI, but the curve is fitted to a local subset defined by a window</td>
<td>Can interpolate short-range variations</td>
<td>Misses the global trends in data</td>
</tr>
<tr>
<td>Trend Surface Analysis</td>
<td>Separates the data into regional trends and local variations</td>
<td>Assists in removal of broader trends prior to further</td>
<td>Edge effects and multi-collinearity caused by spatial</td>
</tr>
<tr>
<td>Method</td>
<td>Linear combination of known points, weighted inversely by distance</td>
<td>Ease to use, and works well with noisy data</td>
<td>Spatial arrangement of samples does not affect weights</td>
</tr>
<tr>
<td>---------------------------</td>
<td>---------------------------------------------------------------------</td>
<td>---------------------------------------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>Inverse Distance Weighing (IDW)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Splines</td>
<td>Fits a smooth curve to a series of data points</td>
<td>Visually appealing curves or contour lines</td>
<td>May mask uncertainty present in the data</td>
</tr>
<tr>
<td>Kriging</td>
<td>Similar to the principle of IDW; however, additionally accounts for the spatial arrangement</td>
<td>Best linear unbiased spatial predictor; and no edge-effects resulting from trying to force a polynomial to fit the data</td>
<td>Sophisticated programming required; and problems of nonstationarity in real-world data sets</td>
</tr>
<tr>
<td>Radial Basis Functions</td>
<td>Similar to the principle of splines, except the curve is not smooth</td>
<td>Require fewer samples</td>
<td>Requires good coverage of the input space; and not suited for extrapolation</td>
</tr>
</tbody>
</table>

**2.2 Prior Art**

Conventional interpolation techniques such as Kriging, IDW, RBF, GPI, and LPI have been employed (Manthena et al. 2009; Kumar et al. 2007). These techniques are discussed below.

GPI fits a smooth surface that is defined by a mathematical function (a polynomial) to the input sample points. The global polynomial surface changes gradually and captures coarse-scale patterns in the data. Conceptually, GPI is like taking a piece of paper and fitting it between the raised points (raised to the height of value). Global interpolators determine a single function which is mapped across the whole region. Change in one input value affects the entire map. Previously, GPI was used in the field of agriculture (Komuscu et al. 1998), climatology (Nalder and Wein 1998; Ninyerola et al. 2000), etc.
An advantage of GPI is that the surface reveals the general/overall trend, enabling us to determine the global model. It is also easy to understand. A disadvantage of GPI is that the calculated surfaces are highly susceptible to outliers (extremely high and low values), especially at the edges. The resulting surface is smooth and lacks minor details. If the polynomial is made of higher order, it results in potentially unreasonable results. GPI is mainly used for fitting a surface to the sample points when the surface has a gently varying trend over the area of interest (e.g., pollution over an industrial area) and for examining or removing the effects of long-range or global trends. In such circumstances the technique is often referred to as trend surface analysis.

While GPI fits a polynomial to an entire surface, LPI fits many polynomials, each within specified overlapping neighborhoods. By varying the shape, maximum and minimum number of points to use, and sector configuration, different interpolated surfaces can be achieved. Thus, LPI produces surfaces that account for more local variation. Local interpolators apply algorithms repeatedly to a small portion of a total set of points. Change in an input value affects the results within the window. LPI fits the specified order (zero, first, second, third, and so on) polynomial using all points only within the defined neighborhood. The neighborhoods overlap and the value used for each prediction is the value of the fitted polynomial at the center of the neighborhood.

GPI is useful for creating smooth surfaces and identifying long-range trends in a dataset. However, in earth sciences, the variable of interest usually has short-range variation in addition to long-range trend. When the dataset exhibits short-range variation, LPI maps can capture the short-range variation. LPI is sensitive to the neighborhood distance. Previously, LPI has been used in estimation of wind speeds (Luo et al. 2008), etc.
LPI is advantageous because it accounts for the local variations, which is the case with most real environmental data. However, the disadvantage of the technique is also the same because, because it is only concerned with local trends in the data, the global trends in data are missed. The LPI method is best suited to situations which have clusters of data with no major global trend.

IDW interpolation explicitly implements the assumption that things close to one another are more alike than those farther apart. To predict a value for any unmeasured location, IDW will use the measured values surrounding the prediction location. Those measured values closest to the prediction location will have more influence on the predicted value than those farther away. Thus, IDW assumes that each measured point has a local influence that diminishes with distance. It weighs the points closer to the prediction location greater than those farther away, hence the name Inverse Distance Weighting. IDW is an exact interpolator, where the maximum and minimum values in the interpolated surface can only occur at sample points. The output surface is sensitive to clustering and the presence of outliers. IDW assumes that the surface is being driven by the local variation, which can be captured through the neighborhood. The difference between LPI and IDW is that IDW defines a model that accounts for anisotropy. IDW has been used in the interpolation of environmental data, e.g. estimating snow distributions (Erxleben et al. 2002), or air quality (Wong 2004), etc.

An advantage of IDW is that it is intuitive and efficient. IDW is also the quickest interpolator. The resulting estimates are continuous over the area of interest. IDW is easy to use and is a common choice in Geographic Information System packages. A main disadvantage of IDW is that this method assumes isotropy; i.e., only the distance between the two locations that is important in determining the weights and not the direction of the line
segment connecting them. Another disadvantage of the IDW functions is that the function is forced to have a maximum or minimum at the data points (or on a boundary of the study region). Similar to Nearest Neighbor, with IDW, unevenly distributed data clusters result in introduced errors. Never is the interpolated value greater than the value at data points on which the interpolation is based. Similar to Spline, IDW is sensitive to outliers. This interpolation works best with evenly distributed points and is moderate in density. Higher density situations result in a rougher surface, while in cases which have lower data points, the error produced is higher.

RBF methods are a series of exact interpolation techniques, \( i.e. \), the surface must go through each measured sample value. RBFs approximate multivariable functions by linear combinations of terms based on a single univariate function (the basis function). There are five important different basis functions:

- Thin-plate spline
- Spline with tension
- Completely regularized spline
- Multiquadric function
- Inverse multiquadric function

Each basis function has a different shape and results in a slightly different interpolation surface. RBF methods are a form of ANNs. They are conceptually similar to fitting a rubber membrane through the measured sample values while minimizing the total curvature of the surface (Burrough and McDonnell 1998). The selection of a basis function determines how the rubber membrane will fit between the values.

When comparing the RBF to the IDW method (another exact interpolator), IDW will never predict values above the maximum measured value or below the minimum measured
value, however, the RBF can predict values above the maximum and below the minimum measured values. The optimal parameters are determined using cross validation in a similar manner as shown for IDW and LPI. RBFs have been previously employed in analyzing the spatial distributions of air pollutants (Duc et al. 2000), estimation of soil zinc (White et al. 1997), etc.

The advantage of RBFs is that they are independent of direction, unlike polynomial schemes, hence complex data patterns can be modeled. A disadvantage is that an RBF predicts values which are above the maximum or below the minimum which might be practically absurd in many situations. RBFs are used for calculating smooth surfaces from a large number of data points. The functions produce good results for gently varying surfaces such as elevation. These techniques are inappropriate when there are large changes in the surface values within a short horizontal distance and/or when we suspect the sample data is prone to error or uncertainty.

Kriging has become a fundamental tool in the field of geostatistics over the past several decades. It is based on the assumption that the parameter being interpolated can be treated as a regionalized variable. A regionalized variable is intermediate between a truly random variable and a completely deterministic variable. This means that it varies in a continuous manner from one location to the next, therefore points that are near each other have a certain degree of spatial correlation, but points that are widely separated are statistically independent (Davis 1986).

Kriging is a stochastic interpolation method. It is similar to IDW in that surrounding measured values are weighted to predict values at unmeasured locations. Unlike IDW, however, Kriging weights are estimated based on spatial autocorrelation between sample points. That is, a statistical relationship between values at sampled points is determined. This
relationship is then applied to make predictions about unmeasured points (Englund 1990). The autocorrelation is a function of distance. In geostatistics, the information on spatial locations allows us to compute distances between observations and to model autocorrelation as a function of distance.

Kriging is a moderately quick interpolator that can be exact or smooth depending on the measurement error model. It is very flexible and allows the user to investigate graphs of spatial autocorrelation. It uses statistical models that allow a variety of map outputs including predictions, prediction standard error, standard error of indicators, and probability. Kriging can be used with larger data sets than many other methods. The ability to compute and assess error, unique to stochastic methods, is another advantage. It requires substantially more computing and modeling time and it also requires more input from the user. Moreover, the flexibility of Kriging can require a lot of decision making. Kriging assumes that the data comes from a stationary stochastic process. This technique is most appropriate when we know there is a spatially correlated distance or directional bias in the data. Kriging has previously been used for location of ores (Journel and Huijbregts 1978; Richmond 2003), estimation of rainfall (Pardo-Iguzquiza 1998), etc.

For the purpose of illustration, estimation maps of radon concentrations using Kriging, IDW, GPI, and LPI are shown in Figure 2.1.
2.3 Review of Artificial Neural Networks

ANNs are being regarded as accurate and fast vehicles to computer aided modeling in comparison to empirical and polynomial models (Meijer 1990). With their remarkable ability to abstract or derive inherent functional relationships from complicated or imprecise data samples, ANNs are gaining attention for modeling complicated patterns/trends that fail other conventional interpolation techniques.

Since the early 1990’s, ANNs have become popular and useful for modeling environmental systems. Any interpolation is based on the idea that a definite relationship exists between the inputs and the outputs. Due to the complexity of real systems, traditional statistical methods have limitations in estimating these underlying functions. Irie and Miyake
(1998) and Hornik et al. (1989) observed ANNs as universal functional approximators since a network can approximate any continuous function to any desired accuracy. Hill et al. (1994) suggested potential advantages of ANN over statistical models. One such advantage is the better performance of the ANN when many extreme values exist. Papanastasiou et al. (2007) also concluded ANN to better perform in forecasting over statistical regression methods. Researchers have used various training algorithms for developing neural network models for various environmental studies.

Akkala et al. (2010) has shown that recently, the numbers of studies that have used neural network techniques in predicting atmospheric pollutant concentrations are increasing. ANNs are a potential alternative to estimate such weather data. ANNs are computer models that mimic the structure and functioning of the human brain (Ward Systems Group Inc., Frederick, MD, 1993). ANNs can determine the relationships among the independent variables to predict or estimate dependent variables. ANNs are known for their ability to generalize well on a wide variety of problems and are well suited for prediction applications. Unlike statistical methods, ANN models do not make dependency assumptions among input variables and solves multivariate problem with nonlinear relationships among input variables. This technique has been used in a wide range of applications such as classification, pattern recognition, automatic control and function approximation (McAvoy et al. 1989; Leonard et al. 1992; Rao and Gupta 1993). Han and Felker (1997) implemented an ANN to estimate daily soil water evaporation from average relative air humidity, air temperature, wind speed, and soil water content in a cactus field study. They found that the ANN achieved a good agreement between predicted and measured values. They concluded that the ANN technique appeared to be an improvement over the multi-linear regression technique for estimating soil evaporation. Elizondo et al. (1994) used an ANN to estimate daily solar radiation for
locations in the southeastern US based on daily maximum and minimum air temperature, daily total precipitation daily clear sky radiation and day length for that location. Cook and Wolfe (1991) developed a neural network to predict average air temperatures. In their study, the monthly average of daily maximum temperatures for three months in advance was predicted. Bruton et al. (2000) developed ANN models for estimating daily pan evaporation. The results were compared with those of multiple linear regression and Priestly-Taylor model and they found that the ANN model provided the highest accuracy.

Neural networks have been used for short term ozone concentration prediction (Ruiz-Suarez et al. 1995; Yi and Prybutok 1996) while Boznar et al. (1993) and Chelani et al. (2002a) used neural techniques to predict short term sulfur dioxide (SO₂) concentrations. Gardner and Dorling (1999) used neural networks to predict hourly NOₓ and NO₂ concentrations in London. Hernandez et al. (1992) predicted daily particle concentrations a day in advance with an accuracy of 30%. Other studies that have used neural networks in environmental studies are Coutinho et al. (2010), Slini et al. (2006), Hooyberghs et al. (2005), Diaz-Robles et al. (2008), Patricio and Jorge (2002), Patricio and Jorge (2001), Caselli et al. (2009), Patricio and Jorge (2006), Grivas and Chaloulakou (2006), Chelani et al. (2002b), and Ordieres et al. (2005). Zickus et al. (2002) and Papanastasiou et al. (2007) predicted particulate matter concentrations using neural networks and compared the results with the predictions from other models.
Artificial Neural Networks Approach

ANNs have been applied to an increasing number of real-world problems of varying complexities. They are known for their ability to model highly complicated input-output relationships that are difficult for conventional techniques (Kangas et al. 1995). After learning and abstracting from either measured or simulated data, referred to as training data, through a process called training, neural models provide instant answers to the task learned (Wang et al. 1999; Devabhaktuni et al. 2002). Theoretically, neural models can be considered black-box models, whose accuracy depends on the training data presented. A good collection of training data, i.e. data that is well-distributed, sufficient, and accurately measured/simulated, is suggested for obtaining an accurate neural model (Wang et al. 1999; Zhang et al. 2003).

An ANN operates by creating connections between many different processing elements called neurons. Each neuron takes input signals and produces a single output signal that is typically sent as input to other neurons. The neurons are tightly interconnected and organized into different layers. While the input layer receives the input, the output layer produces the final output. One or more hidden layers are sandwiched in between the input and output layers.
Let $\mathbf{x}$ be an $n$-vector containing the model inputs. In the case of radon modeling, the inputs are latitude and longitude. Let $\mathbf{y}$ be an $m$-vector containing the model outputs, e.g. radon concentration corresponding to $\mathbf{x}$. The relationship between $\mathbf{x}$ and $\mathbf{y}$ is multi-dimensional and nonlinear, and is given by:

$$\mathbf{y} = f(\mathbf{x}). \tag{1}$$

In (1), $f$ represents the functional relationship between $\mathbf{x}$ and $\mathbf{y}$. In this work, $f$ is a neural network (see Figure 3.1), which is derived or modeled through a training process using a set of sample pairs given by:

$$\{(\mathbf{x}_p, \mathbf{d}_p), \ p = 1,..., N\}. \tag{2}$$

In (2), $\mathbf{d}_p$ represents the desired output corresponding to $p^{th}$ training input vector $\mathbf{x}_p$, $N$ is the number of data samples available for training, and $p$ is simply a sample index. Since our work involves modeling of radon data alone, ANN has only one output, i.e. $m = 1$, and $\mathbf{y}$ and $\mathbf{d}$ are vectors of size one (or scalars).

In reality, the neural network also contains model parameters $\mathbf{w}$, referred to as weights, which are first initialized and then adjusted during the training process. As such, (1) can be re-written as:

$$\mathbf{y} = f(\mathbf{x}, \mathbf{w}). \tag{3}$$

In the case of a 3-layer MLP, weight vector $\mathbf{w}$ contains two sets of weights $\mathbf{u}$ (weights between input layer and hidden layer) and $\mathbf{v}$ (weights between hidden layer and output layer). Size of $\mathbf{w}$ depends upon the size of the neural network, e.g. number of hidden neurons. From a theoretical perspective, the definition of $\mathbf{w}$, and how $\mathbf{y}$ is computed through $\mathbf{x}$ and $\mathbf{w}$, determine the structure of the neural network. It is important to note that the neural network in (3) does not represent the original problem (i.e. radon concentration in this particular case), unless the ANN is trained using the available data.
Figure 3.1: Proposed 3-layer MLP architecture for modeling radon concentrations. The network has two inputs (latitude and longitude), one output (radon concentration), and $h$ hidden neurons.

The objective of training is to determine $w^*$ that minimizes the difference between neural model output $y$ and desired output $d$, given by:

$$E(w) = \frac{1}{2} \sum_{p=1}^{N} \sum_{q=1}^{m} (y_{pq}(x_p, w) - d_{pq})^2.$$  \hspace{1cm} (4)

In (4), $y_{pq}(x_p, w)$ is the $q^{th}$ output of the neural network when presented with input $x_p$. In our case, where $m = 1$, equation (4) can be simplified as:

$$E(w) = \frac{1}{2} \sum_{p=1}^{N} (y_p(x_p, w) - d_p)^2.$$  \hspace{1cm} (5)
Owing to the complexity of $E(w)$, iterative methods are typically used to determine $w^*$. In such methods, we begin with an initial assumption $w_{\text{initial}}$, referred to as initial weights, and then iteratively update $w$ as:

$$w_{\text{next}} = w_{\text{now}} + \eta \cdot g.$$  \hspace{1cm} (6)

In (6), $\eta$ is a positive step-size and $g$ is the update direction. In other words, $w_{\text{next}}$ is determined by adjusting the current weights $w_{\text{now}}$ along an update direction $g$. Different training algorithms use different update directions $g$. Experience helps when choosing the neural network, number of hidden layers, number of hidden neurons, and training method. Too small a network could lead to under-learning while too large a network could lead to over-learning (Wang et al. 1999).

Weights $w^*$ of a trained neural network help calculate/estimate the output of the neural network. We define $z_j$ as the output of the $j^{th}$ hidden sigmoid neuron given by:

$$z_j = \frac{1}{1 + \exp(-\sum_{i=1}^{n} u_{ij} \cdot x_i + u_{0j})},$$ \hspace{1cm} (7)

where $x_i$ is the $i^{th}$ input (which is either latitude or longitude in our case), $u_{ij}$ represents weight of the link between $i^{th}$ input neuron and the $j^{th}$ hidden neuron, and $u_{0j}$ is the bias parameter of the $j^{th}$ hidden neuron. In this work, $y$, which is the model output or radon concentration, is calculated as:

$$y = \sum_{j=1}^{b} (z_j \cdot v_j) + v_0.$$ \hspace{1cm} (8)

In (8), $v_0$ is the bias parameter of the output neuron, $v_j$ represents the weight of link between $j^{th}$ hidden neuron and the output neuron, and $b$ is the total number of hidden neurons.
3.1 Data Preparation

Data for radon concentrations and other related statistics for homes have been collected from various county health departments, commercial testing services and university researchers. The original database for about 50,000 observations was created by Kumar et al. (1990) and formed the basis of the Ohio Radon Information System (Heydinger et al. 1990). The database was extended in 1996 and 1997 to about 82,000 observations. New data are being constantly added to the database. This thesis uses 145,996 indoor radon observations.

The database has been compiled over the last 20 years. Copies of databases were requested from laboratories, university researchers, and others to compile a unified database consisting of 145,996 measurements for 1492 zip code areas in Ohio. Though this appears to be a large number, there are possibly still thousands of readings that must have been taken but not yet reported. The majority of data received was from radon testing organizations and have been reported in a series of technical papers, conference proceedings and reports published by Kumar et al. (1989-2010) (Harrell and Kumar 1989; Kumar et al. 1990; Harrell et al. 1991; Heydinger et al. 1991; Harrell et al. 1993; Kumar et al. 1998; Kumar et al. 2001; Ojha et al. 2001; Joshi et al. 2002; Kumar et al. 2003; Kumar et al. 2006; Kumar et al. 2007; Akkala et al. 2010).

The data sets used were diverse and had to be arranged accordingly. All included the zip code area for the tested buildings and most provided the information on the type of room tested, type of radon detection device used and the season of the year the test was conducted. Only a few sources provided information on the building characteristics and this is unfortunate because some of these parameters may have a significant influence on indoor radon levels.
Zip code and county listings were prepared and the following statistics have been calculated for radon measurements in them: arithmetic mean, geometric mean, standard deviation median, quartile one, quartile three, and variance using Matlab programs (Appendix A). In computing these statistics, every radon measurement in a zip code was used without regard for building, room, detection type or season.

The number of radon concentration data points available in each zip code varies largely. As the radon data is heavily skewed towards the higher concentrations, the geometric mean of radon data available at each zip code is used for modeling purposes. In the conventional GM calculation, a reading of zero present in the multiple readings would result in the GM to be zero (although all other readings for the zip code are non-zero). To overcome this problem, the zero readings are replaced with a value 0.1, which is the minimum detectable radon concentration (Kumar et al. 1990).

At the University of Toledo, data collected as well as that being collected from homes across Ohio on a regular basis is organized into a database (see Table 3.1). Each row contains radon concentration for the corresponding zip code, along with other relevant information (e.g. county name, population, etc).

Table 3.1: An illustrative portion of the University of Toledo radon database containing radon concentrations in addition to other relevant information. GM denotes geometric mean. NA indicates non-availability.

<table>
<thead>
<tr>
<th>Zip Code</th>
<th>PO Name</th>
<th>State</th>
<th>Population (1999)</th>
<th>County ID</th>
<th>County Name</th>
<th>Radon (GM)</th>
<th>Uranium (GM)</th>
</tr>
</thead>
<tbody>
<tr>
<td>43019</td>
<td>Fredericktown</td>
<td>OH</td>
<td>9180</td>
<td>69</td>
<td>Richland</td>
<td>NA</td>
<td>5.21</td>
</tr>
<tr>
<td>43021</td>
<td>Galena</td>
<td>OH</td>
<td>5685</td>
<td>20</td>
<td>Delaware</td>
<td>4.01</td>
<td>7.81</td>
</tr>
<tr>
<td>43022</td>
<td>Gambier</td>
<td>OH</td>
<td>3649</td>
<td>41</td>
<td>Knox</td>
<td>7.01</td>
<td>4.34</td>
</tr>
<tr>
<td>43023</td>
<td>Granville</td>
<td>OH</td>
<td>10586</td>
<td>44</td>
<td>Licking</td>
<td>8.85</td>
<td>NA</td>
</tr>
<tr>
<td>43025</td>
<td>Hebron</td>
<td>OH</td>
<td>7889</td>
<td>22</td>
<td>Fairfield</td>
<td>0.00</td>
<td>9.19</td>
</tr>
</tbody>
</table>
In this work, zip codes and the associated GMs of radon concentrations and uranium concentrations are used as the training data. Out of a total of 1492 zip codes across Ohio, radon concentrations are available for only 1262 zip codes and the uranium concentrations are available for 1013 zip codes. The number of zip codes having both radon concentration data and uranium concentration data are 900 which is useful for one of the KBNN techniques. The minimum value of radon concentration is 0.1 pCi/l for zip code 44648, and the maximum is 39 pCi/l for zip code 43930.

It is not meaningful to use zip code information directly as an ANN input during training. Appreciating this, we obtained the latitude and longitude for each of the zip codes, since such a mapping is one-to-one. As such, the neural network to be trained has two input neurons corresponding to latitude and longitude, as shown in Figure 3.1. An advantage of this approach is that the input space, i.e. spatial regions of Ohio, can be viewed as a 2D grid.

3.2 Typical ANN Error Measures

The quality of a trained neural model is tested with an independent set of data (validation data) and the resulting error is called validation error. The available radon data is first randomized and then split into two independent sets of data, namely, training data and validation data. While the former is used to train the neural network, the latter is used to validate the trained model. An appropriate ANN structure may fail to yield an accurate model, unless trained by a suitable training algorithm (Wang et al. 1999). In this work, we used two training algorithms, namely, Backpropagation and quasi-Newton. In both the cases, the number of hidden neurons is varied to obtain a neural model that yields minimal error. A CAD tool, Neuromodeler (Department of Electronics, Carleton University, Ottawa, Canada), is used for training and validation (Appendix B).
For the case of ANN modeling with one output, which is the case of radon modeling, we define a relative error $\delta_k$ for the $k^{th}$ validation data as:

$$\delta_k = \frac{y_k(x_k, w^*) - d_k}{d_{\text{max},k} - d_{\text{min},k}}, k = 1, \ldots, N_v. \tag{9}$$

In (9), $N_v$ is the number of validation data, $y_k(x_k, w^*)$ is the output of the trained ANN model when presented with $x_k$ as input, and $d_{\text{max},k}$ and $d_{\text{min},k}$ are the maximum and minimum values of $d$ respectively. In this work, the minimum value of radon concentration is 0.1 for zip code 44648, and the maximum is 39 for zip code 43930. A quality measure based on $r^{th}$-norm is defined as:

$$M_r = \left[ \sum_{k=1}^{N_v} |\delta_k|^r \right]^{1/r}. \tag{10}$$

When $r = 1$, average test error can be directly calculated from $M_1$ as:

$$E_{\text{avg}} = \frac{M_1}{N_v}. \tag{11}$$

When $n = 2$, the $n^{th}$-norm measure is the Euclidean distance between the neural model prediction and the test data. When $r = \infty$, the $r^{th}$-norm measure is the maximum test error, which is often referred to as worst-case error among the entire validation data, i.e.,

$$E_{\text{worst}} = M_\infty = \max_{k=1, \ldots, N_v} |\delta_k|, k = 1, \ldots, N_v. \tag{12}$$

Based on these two parameters $E_{\text{avg}}$ and $E_{\text{worst}}$, the best ANN model can be identified. When the training data itself is used to validate the model, the resulting $E_{\text{avg}}$ becomes the training error.
3.3 Comparative Error Measures for Evaluating Spatial Interpolation Techniques

Research work done during 1980’s and 1990’s led to the development of many performance measures to evaluate the air-quality models. EPA has laid some guidelines in order to validate and calibrate models in a comprehensive manner. Kumar and Gudivaka (1990) has discussed in detail the statistics relevant to model evaluation and has applied it to heavy gas models. Similarly, Kumar et al. (1993) has used statistical tools to evaluate the prediction of lower flammability distances. Patel and Kumar (1998), Kumar et al. (1999), and Kumar et al. (2006) have indicated that Mean Absolute Error (MAE), Factor of Two (Fa2), Root Mean Square Error (RMSE), Fractional Bias (FB), and Normalized Mean Square Error (NMSE) are important parameters for assessing the performance of air-quality models. Here, we re-visit these parameters for the purpose of comparison with typically used ANN measures (section 4.2). To keep the various formulas simple and relevant to this work, we assume the number of model outputs as one, which is radon concentration.

3.3.1 Mean Absolute Error

The $MAE$ expressed as:

$$MAE = \frac{\sum_{k=1}^{N} |y_k(x_k, w^*) - d_k|}{N_k},$$

measures the average magnitude of the errors in a set of estimations. Ideal value of $MAE$ is zero, indicating perfect estimation.
3.3.2 Factor of Two

\( Fa2 \) is defined as the percentage of predictions for which the ratio of predicted value to observed value lies between 0.5 and 2.0, \( i.e., \)

\[
0.5 \leq \frac{y_k(x_k, \mathbf{w}^*)}{d_k} \leq 2.0.
\] (14)

The ideal value for \( Fa2 \) should be 1 (or 100%).

3.3.3 Root Mean Square Error

The \( \text{RMSE} \) given by:

\[
\text{RMSE} = \sqrt{\frac{\sum_{k=1}^{N_v} (y_k(x_k, \mathbf{w}^*) - d_k)^2}{N_v}},
\] (15)

is a quadratic scoring rule that measures the average magnitude of the error. It is an indicator of sensitivity to outliers \( (i.e. \) it indicates the magnitude of extreme errors). The ideal value for \( \text{RMSE} \) is zero.

3.3.4 Fractional Bias

The \( FB \) expressed as:

\[
FB = \frac{\sum_{k=1}^{N_v} (y_k(x_k, \mathbf{w}^*) - d_k)}{\frac{1}{2} \sum_{k=1}^{N_v} (y_k(x_k, \mathbf{w}^*) + d_k)},
\] (16)

is a normalized bias. The \( FB \) varies between +2 and -2 and has an ideal value of zero.

3.3.5 Normalized Mean Square Error

The \( \text{NMSE} \) given by:
\[ NMSE = \frac{\sum_{k=1}^{N_v} (y_k(x_k, \mathbf{w}^*) - d_k)^2}{\frac{1}{N_v} \sum_{k=1}^{N_v} y_k(x_k, \mathbf{w}^*) \sum_{k=1}^{N_v} d_k}, \]  

emphasizes the scatter in the entire data set. Normalization by the denominator term assures
that the \( NMSE \) will not be biased towards models that over predict or under predict. Smaller
values of \( NMSE \) denote better model performance, while the ideal value is zero.

3.4 Results

The universal approximation theory states that standard multilayer feed-forward
networks with a single hidden layer that contains finite number of hidden neurons are
universal approximators (Pinkus 1999). Multiple hidden layers have a greater likelihood for
over-learning, especially in such situations, where data patterns are complex (Snell et al.
2000). In light of this observation, neural networks employed in this work are 3-layer multi-
layer perceptrons or MLP3 networks. This being the first attempt to develop ANN models
for the radon problem, there is no existing knowledge in terms of number of hidden layer
neurons etc. As such, it became essential to train several neural networks with different
number of hidden layer neurons.

Neural networks with one hidden layer having different number of hidden neurons have
been trained using two well-known training methods, namely, Backpropagation and quasi-
Newton. Resulting ANN model errors are presented in Tables 3.2 through 3.5. For instance,
Table 3.2 presents training and validation errors for neural models developed using the
Backpropagation algorithm.
Table 3.2: Training and validation errors of ANN models with varying number of hidden neurons trained using the Backpropagation algorithm. Available data is divided into two sets, i.e. 90% training data and 10% validation data.

<table>
<thead>
<tr>
<th>No. of Hidden Neurons</th>
<th>Training Error (%)</th>
<th>Validation Error</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$E_{\text{avg}}$ (%)</td>
</tr>
<tr>
<td>10</td>
<td>4.98</td>
<td>5.50</td>
</tr>
<tr>
<td>20</td>
<td>4.97</td>
<td>5.46</td>
</tr>
<tr>
<td>30</td>
<td>4.93</td>
<td>5.46</td>
</tr>
<tr>
<td>40</td>
<td>4.88</td>
<td>5.37</td>
</tr>
<tr>
<td>50</td>
<td>4.82</td>
<td>5.31</td>
</tr>
<tr>
<td>60</td>
<td>4.72</td>
<td>5.19</td>
</tr>
<tr>
<td>70</td>
<td>4.70</td>
<td>5.02</td>
</tr>
<tr>
<td>80</td>
<td>4.70</td>
<td>4.96</td>
</tr>
<tr>
<td>90</td>
<td>4.67</td>
<td>4.86</td>
</tr>
<tr>
<td>100</td>
<td>4.65</td>
<td>4.95</td>
</tr>
<tr>
<td>110</td>
<td>4.59</td>
<td>5.00</td>
</tr>
<tr>
<td>120</td>
<td>4.57</td>
<td>5.16</td>
</tr>
<tr>
<td>130</td>
<td>4.57</td>
<td>5.18</td>
</tr>
<tr>
<td>140</td>
<td>4.56</td>
<td>5.19</td>
</tr>
<tr>
<td>150</td>
<td>4.51</td>
<td>5.33</td>
</tr>
</tbody>
</table>

Table 3.3: Training and validation errors of ANN models with varying percentage of training data trained using the Backpropagation algorithm. Based on Table 3.2, the number of hidden neurons is fixed to be 90.

<table>
<thead>
<tr>
<th>% Training Data</th>
<th>% Validation Data</th>
<th>Training Error (%)</th>
<th>Validation Error</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$E_{\text{avg}}$ (%)</td>
<td>$E_{\text{worst}}$ (%)</td>
</tr>
<tr>
<td>90</td>
<td>10</td>
<td>4.67</td>
<td>4.86</td>
</tr>
<tr>
<td>80</td>
<td>20</td>
<td>4.65</td>
<td>4.95</td>
</tr>
<tr>
<td>70</td>
<td>30</td>
<td>4.64</td>
<td>5.26</td>
</tr>
<tr>
<td>60</td>
<td>40</td>
<td>4.62</td>
<td>5.28</td>
</tr>
<tr>
<td>50</td>
<td>50</td>
<td>4.52</td>
<td>5.54</td>
</tr>
</tbody>
</table>
Table 3.4: Training and validation errors of ANN models with varying number of hidden neurons trained using the quasi-Newton algorithm. Available data is divided into two sets, i.e. 90% training data and 10% validation data.

<table>
<thead>
<tr>
<th>No. of Hidden Neurons</th>
<th>Training Error (%)</th>
<th>Validation Error</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$E_{\text{avg}}$ (%)</td>
</tr>
<tr>
<td>10</td>
<td>4.50</td>
<td>5.27</td>
</tr>
<tr>
<td>20</td>
<td>4.45</td>
<td>5.25</td>
</tr>
<tr>
<td>30</td>
<td>4.44</td>
<td>5.19</td>
</tr>
<tr>
<td>40</td>
<td>4.44</td>
<td>5.15</td>
</tr>
<tr>
<td>50</td>
<td>4.40</td>
<td>5.10</td>
</tr>
<tr>
<td>60</td>
<td>4.40</td>
<td>4.88</td>
</tr>
<tr>
<td>70</td>
<td>4.39</td>
<td>4.87</td>
</tr>
<tr>
<td>80</td>
<td>4.37</td>
<td>4.83</td>
</tr>
<tr>
<td>90</td>
<td>4.36</td>
<td>4.81</td>
</tr>
<tr>
<td>100</td>
<td>4.36</td>
<td>4.69</td>
</tr>
<tr>
<td><strong>110</strong></td>
<td><strong>4.34</strong></td>
<td><strong>4.64</strong></td>
</tr>
<tr>
<td>120</td>
<td>4.34</td>
<td>4.67</td>
</tr>
<tr>
<td>130</td>
<td>4.27</td>
<td>4.70</td>
</tr>
<tr>
<td>140</td>
<td>4.27</td>
<td>4.86</td>
</tr>
<tr>
<td>150</td>
<td>4.25</td>
<td>4.90</td>
</tr>
</tbody>
</table>

Table 3.5: Training and validation errors of ANN models with varying percentage of training data trained using the quasi-Newton algorithm. Based on Table 3.4, the number of hidden neurons is fixed to be 110.

<table>
<thead>
<tr>
<th>% Training Data</th>
<th>% Validation Data</th>
<th>Training Error (%)</th>
<th>Validation Error</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$E_{\text{avg}}$ (%)</td>
<td>$E_{\text{worst}}$ (%)</td>
</tr>
<tr>
<td>90</td>
<td>10</td>
<td>4.34</td>
<td>4.64</td>
</tr>
<tr>
<td>80</td>
<td>20</td>
<td>4.30</td>
<td>4.73</td>
</tr>
<tr>
<td>70</td>
<td>30</td>
<td>4.27</td>
<td>4.94</td>
</tr>
<tr>
<td>60</td>
<td>40</td>
<td>4.20</td>
<td>5.05</td>
</tr>
<tr>
<td>50</td>
<td>50</td>
<td>4.12</td>
<td>5.19</td>
</tr>
</tbody>
</table>

A close inspection of the results presented in Table 3.2 shows that the training error decreases with increasing number of hidden neurons (and this phenomenon is confirmed by
Table 3.4 as well). However, the validation error initially decreases with hidden neurons, and then increases. Such behavior can be attributed to over-learning. Tables 3.3 and 3.5 confirm that more the training data, less the validation error, which is natural. From Tables 3.2 and 3.4, it can be concluded that training by quasi-Newton leads to relatively more accurate models as compared to that by Backpropagation. Figure 3.2 shows output of the ANN model when presented with validation data.

![Image](image.png)

Figure 3.2: A screen-shot of the Neuromodeler software showing the validation data (magenta line) and the radon concentration values as predicted by the neural model (yellow line).
Chapter 4

Knowledge Based Neural Networks Approach

In the ANN approach, the fact that radon gas is associated with the geological occurrence of uranium and other radioactive elements/products has not been considered. Radon is formed from uranium by the decay chain shown in Figure 4.1. As such, houses and other structures built above uranium-bearing rocks or sediments may have higher indoor radon levels (Harrell et al. 1993). This fact is corroborated by the uranium concentration map of Ohio (Figure 4.2) with the radon concentration map (Figure 2.1). The observation represents knowledge, which could be used for further improving ANN model accuracies.

Figure 4.1: Decay series of uranium with daughter products and half-lives, showing the formation of radon from uranium. This chain helps us in understanding dependency of radon concentrations on those of uranium.
Figure 4.2: Radiometric map of Ohio showing the uranium concentration in soil (Source: http://radon.utoledo.edu).

In this section, a brief overview of the various techniques and network structures available for embedding knowledge into the neural network is presented. By inserting the additional knowledge into the neural network structure, the empirical connections between the parameters can be refined/adjusted as part of the overall neural network training process. This technique enhances neural model accuracy, especially for the data not seen
during training (generalization capability), and produces neural models which are more accurate in terms of lowering the validation error.

4.1 Review of Knowledge Based Neural Networks

The basic idea of the knowledge based methods is to embed the existing knowledge as a part of the overall neural network structure, as can be seen in Figure 4.3. The structure contains the original data (knowledge) along with additional knowledge.

![Figure 4.3: Structure of a Knowledge Based Neural Network, which embeds additional knowledge into the neural network, as a part of the training process.](image)

The additional knowledge is usually in the form of data to which the output can be correlated. The KBNN model can directly learn from this data. During the training process, the weights of the neural networks are all automatically adjusted such that the overall KBNN model matches the training data.

The Prior Knowledge Input (PKI) method was proposed in (Watson et al. 1998). In this method, the knowledge inputs are used directly as additional inputs to the neural network model along with the original network inputs. Here, the input-output relationship learned by the neural network is between the knowledge inputs and the variable being modeled.

The neural network consists of an additional input to represent the available knowledge (see Figure 4.4) and represents the mapping between the inputs and the output. The quality
of this mapping is enhanced by including the additional inputs to the neural network. The PKI model improves the accuracies but retains the speed of the MLP models.

Figure 4.4: The structure of a PKI method. The knowledge is used as an additional input to the model.

The Source Difference Model (SDM) method, proposed in (Watson and Gupta 1996) is one of the earlier methods for embedding knowledge into the neural network structure. The overall structure consists of an empirical or approximate neural model (referred to as coarse model). A second neural network that represents the difference between the actual radon concentration data and approximate model (known as the difference model) is shown in Figure 4.5.

Using the difference as the desired output of the neural network results in a smaller range of the outputs as compared to the original problem, and hence, a simpler input-output relationship. The difference model training therefore fine-tunes the predictions of the coarse model and helps capture the data trends more accurately. After training, given input $\mathbf{x}$, the coarse model predicts the approximate concentration values and the difference model estimates the difference between actual radon concentrations and the empirical model outputs. The sum of the coarse model output and the difference model output is the final output of the overall SDM model.
The principle of SDM method. The difference between the coarse model and the expected output is modeled for fine-tuning the estimations.

The Space-Mapped Neural Network (SMNN) approach was proposed in (Bandler et al. 1994). The Space-Mapping (SM) technique (Bandler et al. 1999) combines the computational efficiency of coarse models with the accuracy of fine models. This technique establishes a mathematical link between the inputs of the SM and the coarse models.

The neural network module maps the original problem input-space $\mathbf{x}$ (i.e., SM model input-space) into a coarse model input-space $\mathbf{x}_{\text{coarse}}$. The coarse model then produces the overall output $\mathbf{y}$, which should match the expected output. The objective of SM is to find an appropriate mapping from the SM model input-space $\mathbf{x}$ to the coarse model input-space $\mathbf{x}_{\text{coarse}}$. Once such mapping is found, the coarse model can be used for fast and accurate simulations. Figure 4.6 shows the structure of a SMNN model.
4.2 Proposed Methodology

Motivated by the above discussed approaches, three approaches were applied for modeling radon and uranium concentrations using KBNNs, namely PKI, SDM, and SMNN. In the KBNNs used in this work, knowledge in the form of radon concentration data and uranium concentration data is embedded into the neural network structure.

In the PKI approach we used a 3 layer MLP with uranium concentration as an additional input along with the latitude and longitude to train the neural model (see Figure 4.7). By doing this, the correlation between the radon and uranium concentrations is established by the neural model through the training process. This input-output relationship is given by:

\[ y_{P_{KI}} = f_{P_{KI}}(x, x_{\text{knowledge}}, w). \] (18)

In this case, \( y_{P_{KI}} \) is the output of the PKI model (i.e. radon concentration), \( x \) is the input vector having latitude and longitude, and \( x_{\text{knowledge}} \) is the knowledge input (i.e. uranium concentration), \( f_{P_{KI}} \) is the functional relationship that is established during the training process, and \( w \) is the weights of the neural model. This approach successfully utilizes the available knowledge in terms of uranium concentration data. However, this approach requires the uranium concentration data to be available for all the points where radon concentration data is available and those places where the radon concentrations need to be predicted. This imposes a limitation on the number of available training data as any missing uranium concentration data would result in that data element to be treated as an unknown data point.

The training of the PKI model is as follows:

Step 1: Identify the zip codes that have both the radon and uranium concentration data and obtain the latitude and longitude information for these zip codes.
Step 2: Train the PKI model with inputs as latitude, longitude, and uranium concentration vs. output as the radon concentration.

Step 3: Compute the PKI model output with latitude, longitude, and uranium concentration as inputs, which should match the expected values of radon concentration.

The neural network used in the PKI method is a 3-layer MLP and the number of hidden neurons was varied to identify the best neural network that gives the least average error $E_{avg}$.

Figure 4.7: PKI method using 3 layer MLP network.

The SDM involves training in two phases. The first phase of the training employs a coarse neural network model trained using the latitude, longitude pairs versus the radon concentration data. For the first network which models the radon concentration, the neural model establishes a relation between the input $\mathbf{x}$ and output $y_{\text{coarse}}$, given by:

$$y_{\text{coarse}} = f(\mathbf{x}, \mathbf{w}_{\text{coarse}}),$$

(19)
where \( \mathbf{x} \) is the input (i.e., latitude and longitude), \( y_{\text{coarse}} \) is the output of the coarse model, \( f \) is the input-output relationship that is modeled by the first neural model, and \( w_{\text{coarse}} \) is the weights of the coarse neural model. This model is now trained and can be used to approximately estimate the radon concentration values. The second phase of training employs another neural network (also known as the difference model), and is done with the same inputs as the coarse model (i.e., latitude and longitude) versus the difference between the coarse model output and the expected concentration values, as shown in Figure 4.8. This relationship is given by:

\[
y_{\text{difference}} = \delta(x, w_{\text{difference}}),
\]

(20)

where \( y_{\text{difference}} \) is the output of the difference model, \( x \) is the same input as the first model (i.e., latitude and longitude), \( \delta \) is the input-output relationship modeled by the difference model, and \( w_{\text{difference}} \) is the weights of the difference model. Note that knowledge in terms of uranium concentrations is not used in this technique. While estimating the missing concentration of a zip code, the coarse model is used to estimate the approximate values of the radon concentration \( y_{\text{coarse}} \) value and the difference model will estimate the difference \( y_{\text{difference}} \). The final value of the predicted concentration \( y \) is given by:

\[
y = y_{\text{coarse}} + y_{\text{difference}}.
\]

(21)

In (21), \( y \) is the final output of the SDM model, which is the radon concentration. Since we are using two neural models in the SDM technique, we need to identify the best neural model that gives the least validation error \( E_{\text{avg}} \). This is achieved by varying the number of neurons in each of the models. The neural model having the least validation error is chosen for the coarse and the difference model.

The training procedure of the SDM model is as follows:
Step 1: Identify the zip codes that have radon concentrations available and obtain the latitude, longitude information for these zip codes.

Step 2: Train a coarse model with latitude and longitude as inputs vs. radon concentration as output.

Step 3: Compute the coarse model output with latitude and longitude as inputs to find the approximate values of estimated radon concentrations.

Step 4: Compute the difference between the approximate outputs and the expected radon concentration values.

Step 5: Train the difference model with the difference computed in Step 4.

Step 6: Compute the difference model output with latitude and longitude as inputs.

Step 7: Calculate the sum of outputs of the coarse model and the difference model, which should match the expected radon concentration values.
Figure 4.8: The structure of the SDM technique used to train the neural model in two stages.

In the SMNN approach, two neural models are used. Both these networks are connected such that the output of the SM (first) network is the input to the coarse (second) network. The inputs to this model are the latitude, longitude, and uranium concentrations, and the output is a second order vector having intermediate inputs latitude’ and longitude’. The relation between the input and output is given by:

\[ \mathbf{x}_{\text{coarse}} = f_{\text{SM}}(\mathbf{x}, \mathbf{x}_{\text{knowledge}}, \mathbf{w}_{\text{SM}}), \]  

(22)
where \( \mathbf{x} \) is the input, \( \mathbf{x}_{\text{knowledge}} \) is the knowledge input, \( \mathbf{x}_{\text{coarse}} \) is the output of the first model, \( \mathbf{f}_{\text{SM}} \) is the input-output relationship that is modeled by the space mapping model, and \( \mathbf{w}_{\text{SM}} \) is the weights of the SM model. A mathematical relationship is identified between the input space of the SM network \( (\mathbf{x}, \mathbf{x}_{\text{knowledge}}) \) and that of the coarse model \( \mathbf{x}_{\text{coarse}} \). The knowledge in terms of uranium data is used as a part of the input to the SM neural network to model the intermediate inputs \( \mathbf{x}_{\text{coarse}} \). The intermediate input layer acts as a sandwich between the original input and output dividing the modeling complexity of relationships into two stages.

The SMNN model used for modeling radon concentrations is shown in Figure 4.9.

The coarse model is trained with the radon concentrations as the output and \( \mathbf{x}_{\text{coarse}} \) as the input, which is the radon concentration. The model equation is given by:

\[
\mathbf{y} = \mathbf{f}_{\text{coarse}} (\mathbf{x}_{\text{coarse}}, \mathbf{w}_{\text{coarse}}),
\]

where \( \mathbf{y} \) is the final output of the SMNN technique (i.e. the radon concentration), \( \mathbf{x}_{\text{coarse}} \) is the intermediate input (i.e. latitude’ and longitude’), \( \mathbf{w}_{\text{coarse}} \) is the weights of the coarse neural network, and \( \mathbf{f}_{\text{coarse}} \) is the input-output relationship that is modeled by the coarse model.

Hence, the final equation of the SMNN model is:
\[ y = f_{\text{coarse}}(f_{\text{SM}}(x, x_{\text{knowledge}}, w_{\text{SM}}), w_{\text{coarse}}). \] (24)

The training procedure of the SMNN model is done as follows:

Step 1: Identify the zip codes that have both the radon and uranium concentration data, and obtain the latitude, longitude information for them.

Step 2: Train an MLP3 network with available radon concentration data as input vs. outputs as corresponding latitude and longitude.

Step 3: Use the trained model to predict the values of latitude’ and longitude’ using radon concentration as input.

Step 4: Train the SM model with inputs as latitude, longitude, and uranium concentration vs. outputs as latitude’ and longitude’.

Step 5: Train the coarse model with latitude’ and longitude’ as inputs vs. radon concentration as output.

Step 6: Compute the output of the coarse model with latitude’ and longitude’ as inputs, which should match the expected radon concentration values.

Similar to SDM, the SMNN technique uses two neural models. Hence for each model, we need to identify the network that gives the most accurate prediction. For this purpose, the number of hidden neurons in each of the models is varied and the model that gives the least validation error \( E_{\text{avg}} \) is identified.

After identifying the best of PKI, SDM, and SMNN models, we compare them with the conventional interpolation techniques and the previous ANN model which uses only the radon concentration data for estimation. This is done by using the comparative error measures used for evaluating spatial interpolation techniques for environmental data given in Section 3.2.
4.3 Results

Various knowledge-based approaches exploiting knowledge in terms of radon and uranium concentrations have been applied to model and predict the radon concentrations in places where it is not measured directly. The available data is split into training and validation data for the purpose of validating the proposed knowledge based neural models. In this case, it has been split into 90% training data and 10% validation data. The models have been trained using the training data and have been validated using the validation data. The resulting errors have been used as the evaluating criteria for ranking them. Among these models, the model that produces the least $E_{\text{avg}}$ is identified as the best model for that technique.

In the PKI method, several models are generated by varying the number of neurons in the hidden layer (hidden neurons). From these models, the model that gives the least $E_{\text{avg}}$ is identified as the best PKI model for predicting the missing radon concentrations. Table 4.1 shows the training and validation errors of the several models using PKI method with varying number of hidden neurons.

Table 4.1: Training and validation errors of the PKI model with varying number of hidden neurons.

<table>
<thead>
<tr>
<th>No. of Hidden Neurons</th>
<th>Training Error (%)</th>
<th>Validation Error</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$E_{\text{avg}}$ (%)</td>
<td>$E_{\text{worst}}$ (%)</td>
</tr>
<tr>
<td>10</td>
<td>0.90</td>
<td>0.97</td>
<td>18.45</td>
</tr>
<tr>
<td>20</td>
<td>0.87</td>
<td>0.93</td>
<td>18.31</td>
</tr>
<tr>
<td>30</td>
<td>0.84</td>
<td>0.90</td>
<td>19.62</td>
</tr>
<tr>
<td><strong>40</strong></td>
<td><strong>0.81</strong></td>
<td><strong>0.88</strong></td>
<td>18.86</td>
</tr>
<tr>
<td>50</td>
<td>0.79</td>
<td>0.91</td>
<td>19.98</td>
</tr>
<tr>
<td>60</td>
<td>0.78</td>
<td>1.05</td>
<td>18.72</td>
</tr>
<tr>
<td>70</td>
<td>0.76</td>
<td>1.04</td>
<td>19.79</td>
</tr>
</tbody>
</table>
Table 4.1 shows that in the PKI method, the neural model having 40 hidden neurons is the one which gives the least prediction error.

In the case of SDM, there is a coarse model and a difference model. The coarse model of the SDM method is a 3 layer MLP model that has two inputs (i.e. latitude and longitude) and one output (i.e. radon concentration). The difference model has two inputs (i.e. latitude and longitude), and one output (i.e. the difference between the coarse model output and the expected values of radon concentration). The predictions from both of these models are added to obtain the predictions of the overall SDM technique.

To identify the best SDM model, we need to find the best coarse and difference models. Since the training data for the difference model is dependent on the coarse model output, the best coarse model is first identified. This is done by varying the number of hidden neurons in the coarse model and choosing the model that gives the least $E_{avg}$. Then the training data for the difference model is obtained based on the predictions of the best coarse model. The difference between the best coarse model output and the corresponding expected output is used as the training data for the difference model. Further, the best difference model is identified by varying the number of hidden neurons in the model and choosing the model that gives the least $E_{avg}$. Table 4.2 presents the training and validation errors of the coarse model.

<table>
<thead>
<tr>
<th>80</th>
<th>0.76</th>
<th>1.01</th>
<th>19.68</th>
</tr>
</thead>
<tbody>
<tr>
<td>90</td>
<td>0.76</td>
<td>1.02</td>
<td>18.99</td>
</tr>
<tr>
<td>100</td>
<td>0.75</td>
<td>1.21</td>
<td>19.19</td>
</tr>
</tbody>
</table>
Table 4.2: Training and validation errors of the coarse model used in the SDM technique with varying number of hidden neurons.

<table>
<thead>
<tr>
<th>No. of Hidden Neurons</th>
<th>Training Error (%)</th>
<th>Validation Error</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$E_{avg}$ (%)</td>
</tr>
<tr>
<td>10</td>
<td>4.50</td>
<td>5.27</td>
</tr>
<tr>
<td>20</td>
<td>4.45</td>
<td>5.25</td>
</tr>
<tr>
<td>30</td>
<td>4.44</td>
<td>5.19</td>
</tr>
<tr>
<td>40</td>
<td>4.44</td>
<td>5.15</td>
</tr>
<tr>
<td>50</td>
<td>4.40</td>
<td>5.10</td>
</tr>
<tr>
<td>60</td>
<td>4.40</td>
<td>4.88</td>
</tr>
<tr>
<td>70</td>
<td>4.39</td>
<td>4.87</td>
</tr>
<tr>
<td>80</td>
<td>4.37</td>
<td>4.83</td>
</tr>
<tr>
<td>90</td>
<td>4.36</td>
<td>4.81</td>
</tr>
<tr>
<td>100</td>
<td>4.36</td>
<td>4.69</td>
</tr>
<tr>
<td><strong>110</strong></td>
<td><strong>4.34</strong></td>
<td><strong>4.64</strong></td>
</tr>
<tr>
<td>120</td>
<td>4.34</td>
<td>4.67</td>
</tr>
<tr>
<td>130</td>
<td>4.27</td>
<td>4.70</td>
</tr>
<tr>
<td>140</td>
<td>4.27</td>
<td>4.76</td>
</tr>
<tr>
<td>150</td>
<td>4.25</td>
<td>4.78</td>
</tr>
</tbody>
</table>

From Table 4.2, the best coarse model in the SDM model is identified as the one having 110 hidden neurons. Fixing the coarse model, now we vary the number of hidden neurons to identify the best difference model and the best overall SDM model. Table 4.3 shows the training and validation errors of the difference model used in the SDM method along with the validation errors for the overall SDM model.
Table 4.3: Training and validation errors of the difference model used in the SDM technique and the overall SDM method. The number of hidden neurons for the coarse model is fixed as 110.

<table>
<thead>
<tr>
<th>No. of Hidden Neurons</th>
<th>Training Error (%)</th>
<th>Validation Error</th>
<th>Validation Error</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( E_{\text{avg}} ) (%)</td>
<td>( E_{\text{worst}} ) (%)</td>
<td>( E_{\text{avg}} ) (%)</td>
</tr>
<tr>
<td>10</td>
<td>3.17</td>
<td>3.85</td>
<td>40.54</td>
</tr>
<tr>
<td>20</td>
<td>3.10</td>
<td>3.67</td>
<td>40.03</td>
</tr>
<tr>
<td>30</td>
<td>3.07</td>
<td>3.61</td>
<td>39.48</td>
</tr>
<tr>
<td>40</td>
<td>3.04</td>
<td>3.55</td>
<td>40.27</td>
</tr>
<tr>
<td>50</td>
<td>3.02</td>
<td>3.49</td>
<td>39.21</td>
</tr>
<tr>
<td>60</td>
<td>3.01</td>
<td>3.46</td>
<td>39.13</td>
</tr>
<tr>
<td>70</td>
<td>2.98</td>
<td>3.44</td>
<td>39.55</td>
</tr>
<tr>
<td>80</td>
<td>2.96</td>
<td>3.42</td>
<td>38.68</td>
</tr>
<tr>
<td>90</td>
<td>2.95</td>
<td>3.43</td>
<td>40.01</td>
</tr>
<tr>
<td>100</td>
<td>2.93</td>
<td>3.44</td>
<td>39.89</td>
</tr>
</tbody>
</table>

From Table 4.3, it can be seen that the best overall model is obtained when the difference model has 70 hidden neurons. Hence, the best SDM model has 110 hidden neurons in the coarse model and 70 hidden neurons in the difference model.

In the case of SMNN technique also, there are two models. Hence, the best model for the space mapping model and the coarse model need to be identified. Fixing this model as the best SM model, the best coarse model is similarly identified by varying the number of hidden neurons and selecting the model which gives the least \( E_{\text{avg}} \). Table 4.4 gives the training and validation errors of the SM model in the SMNN technique by varying the number of hidden neurons.
Table 4.4: Training and validation errors of the SM model used in SMNN technique with varying number of hidden neurons.

<table>
<thead>
<tr>
<th>No. of Hidden Neurons</th>
<th>Training Error (%)</th>
<th>Validation Error</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$E_{\text{avg}}$ (%)</td>
</tr>
<tr>
<td>10</td>
<td>1.22</td>
<td>1.66</td>
</tr>
<tr>
<td>20</td>
<td>1.21</td>
<td>1.62</td>
</tr>
<tr>
<td>30</td>
<td>1.21</td>
<td>1.57</td>
</tr>
<tr>
<td>40</td>
<td>1.21</td>
<td>1.54</td>
</tr>
<tr>
<td>50</td>
<td>1.20</td>
<td><strong>1.52</strong></td>
</tr>
<tr>
<td>60</td>
<td>1.20</td>
<td>1.53</td>
</tr>
<tr>
<td>70</td>
<td>1.20</td>
<td>1.56</td>
</tr>
<tr>
<td>80</td>
<td>1.19</td>
<td>1.57</td>
</tr>
<tr>
<td>90</td>
<td>1.19</td>
<td>1.57</td>
</tr>
<tr>
<td>100</td>
<td>1.18</td>
<td>1.59</td>
</tr>
</tbody>
</table>

From Table 4.4, the best space mapping model in the SMNN technique is identified as the one having 50 hidden neurons. Fixing this, the number of hidden neurons is varied to identify the best coarse model. Table 4.5 shows the training and validation errors of the coarse model.

Table 4.5: Training and validation errors of the coarse model used in SMNN technique with varying number of hidden neurons. The number of hidden neurons in the space mapping model is fixed to be 50.

<table>
<thead>
<tr>
<th>No. of Hidden Neurons</th>
<th>Training Error (%)</th>
<th>Validation Error</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$E_{\text{avg}}$ (%)</td>
</tr>
<tr>
<td>10</td>
<td>0.45</td>
<td>0.71</td>
</tr>
<tr>
<td>20</td>
<td>0.39</td>
<td>0.65</td>
</tr>
<tr>
<td>30</td>
<td>0.35</td>
<td>0.61</td>
</tr>
<tr>
<td>40</td>
<td>0.33</td>
<td>0.60</td>
</tr>
<tr>
<td>50</td>
<td>0.32</td>
<td>0.56</td>
</tr>
<tr>
<td>60</td>
<td><strong>0.31</strong></td>
<td><strong>0.54</strong></td>
</tr>
<tr>
<td>70</td>
<td>0.31</td>
<td>0.55</td>
</tr>
<tr>
<td>80</td>
<td>0.30</td>
<td>0.59</td>
</tr>
</tbody>
</table>
From Table 4.5, it can be seen that the neural model with 60 hidden neurons is the best coarse model for the SMNN technique. Hence, the best SMNN model is one having 50 hidden neurons in the space mapping model and 60 hidden neurons in the coarse model.

After obtaining best models for each of the techniques PKI, SDM, and SMNN, their performance is compared with the previous techniques (ANN model and conventional interpolation methods), as shown in Table 4.6.

Table 4.6: Performance of the ANN and KBNN models developed in comparison with the conventional interpolation techniques.

<table>
<thead>
<tr>
<th>Interpolation Technique</th>
<th>MAE</th>
<th>F_a2</th>
<th>RMSE</th>
<th>FB</th>
<th>NMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kriging</td>
<td>1.19</td>
<td>0.84</td>
<td>2.04</td>
<td>0.06</td>
<td>0.41</td>
</tr>
<tr>
<td>Radial Basis Function</td>
<td>1.21</td>
<td>0.86</td>
<td>2.14</td>
<td>0.06</td>
<td>0.45</td>
</tr>
<tr>
<td>Inverse Distance Weighting</td>
<td>1.24</td>
<td>0.85</td>
<td>2.21</td>
<td>0.06</td>
<td>0.48</td>
</tr>
<tr>
<td>Local Polynomial Interpolation</td>
<td>1.46</td>
<td>0.70</td>
<td>2.39</td>
<td>-0.05</td>
<td>0.96</td>
</tr>
<tr>
<td>Global Polynomial Interpolation</td>
<td>1.58</td>
<td>0.65</td>
<td>2.55</td>
<td>-0.45</td>
<td>1.07</td>
</tr>
<tr>
<td>Artificial Neural Networks</td>
<td>0.79</td>
<td>0.86</td>
<td>0.9</td>
<td>0.01</td>
<td>0.08</td>
</tr>
<tr>
<td>Prior Knowledge Input</td>
<td>0.39</td>
<td>0.98</td>
<td>1.50</td>
<td>0.04</td>
<td><strong>0.03</strong></td>
</tr>
<tr>
<td>Source Difference Model</td>
<td>1.04</td>
<td>0.91</td>
<td>1.21</td>
<td>-0.01</td>
<td>0.09</td>
</tr>
<tr>
<td>Space Mapped Neural Network</td>
<td><strong>0.33</strong></td>
<td><strong>0.99</strong></td>
<td><strong>0.59</strong></td>
<td><strong>0.01</strong></td>
<td>0.07</td>
</tr>
</tbody>
</table>

As shown in Table 4.6, the SMNN technique gives the least $MAE$, $F_a2$, $RMSE$, and $FB$, while the PKI method gives the least $NMSE$. This shows that the SMNN model with 50 hidden neurons in the SM model and 60 hidden neurons in the coarse model gives the best interpolation results compared to the other KBNN techniques, the ANN technique, and the conventional interpolation techniques.
Chapter 5

Conclusions and Future Work

5.1 Conclusions

For the first time, this thesis demonstrates the use of ANNs for modeling radon concentrations in Ohio. The available radon concentration data was reformatted to suit ANN training by translating the zip code information into latitude and longitude data. A comparison of the ANN model with conventional interpolation techniques shows that the proposed technique results in relatively better accuracies, as shown in Table 4.6.

The best neural model, *i.e.* ANN model with 110 hidden neurons, trained using quasi-Newton is chosen as the best MLP3 approach for the purpose of estimating radon concentrations in zip codes with no available data. From Tables 3.3 and 3.5, it can be seen that more training data results in a lesser validation error. Based on this observation, it is recommended that all available data be used for training to accurately predict the radon concentrations in the 230 zip codes with no available data.

In the second phase of the thesis, various knowledge based neural models have been implemented to predict the radon concentrations exploiting the knowledge available in terms of radon and uranium concentration data in an attempt to embed knowledge into the otherwise statistical learning black-box ANN models. One of the techniques *i.e.* the PKI
method is a simple extension of the MLP3 network used previously for the estimation of radon concentrations. The SDM is based on the use of two neural models – one coarse and the other model to fine-tune the predictions of the coarse model. The SMNN technique is based on the manipulation of the input space for one neural model by the other model. Table 4.6 gives the comparison of the performance of these techniques to conventional interpolation techniques and the MLP3 approach. It can be seen that the SMNN gives the best performance in terms of $MAE$, $Fa2$, $RMSE$, and $FB$; whereas the PKI method gives the best performance in term of $NMSE$. Hence, the SMNN method is more reliable in the scenario of predicting missing radon concentrations in Ohio. It can also be seen that all the knowledge based approaches are more efficient as compared to rest of the techniques such as RBF and Kriging. Also, these techniques are proven to be better than the simple MLP3 approach used in the first phase of the thesis.

5.2 Future Work

Future work in this direction could include exploiting any other knowledge relevant to the radon concentration data, apart from the uranium concentration data. Exploring further advanced techniques based on ANN modeling could lead to better prediction results. Furthermore, several other techniques exist in literature that have not yet been explored for application to this problem, e.g. Support Vector Machines, which is being used extensively in the field of image processing for reconstruction of images. This could be a potential technique as our problem is also similar. A user interface could also be developed which can be used to visualize the radon concentrations across Ohio, automatically calculate various statistics such as GM from the raw data, and consecutively, perform predictions using a technique from a list containing several techniques such as LPI, Kriging, SDM, or SMNN.
References


NeuroModeler, Dep Electron, Carleton University, 1125 Colonel By Drive, Ottawa, K1S5B6, Ontario, Canada.


Appendix A

Source Code for Statistics Computation of Zip Codes and Counties

A.1 Source Code for Computing Various Zip Code Statistics from Raw Data

clear all;

A = load('U:\RADON_DATA\zip_raw_data.csv');
zips = A(:,1);
radon = A(:,2);
N = length(zips);
count = 1;
for i=2:N
    if zips(i) ~= zips(i-1)
        count = count + 1;
    end
end

temp = zips(1);
j=1;
unique_zip = zeros(count,1);
Median = zeros(count,1);
AM = zeros(count,1);
GM = zeros(count,1);
Quart1 = zeros(count,1);
Quart3 = zeros(count,1);
Max_values = zeros(count,1);
Min_values = zeros(count,1);
SDev = zeros(count,1);
Int_QRange = zeros(count,1);
Simple_Range = zeros(count,1);

sum = 0.00;
prod = double(1.00);
n=1;

for i=1:N-1
    if temp == zips(i+1)
        if i+1 == N
            i=i+1;
            unique_zip(j,1) = zips(i);
            GM(j,1) = geomean(radon(i-n+1:i));
            AM(j,1) = mean(radon(i-n+1:i));
            Min_values(j,1) = min(radon(i-n+1:i));
            Max_values(j,1) = max(radon(i-n+1:i));
            SDev(j,1) = std(radon(i-n+1:i));
            Median(j,1) = median(radon(i-n+1:i));
            A = sort(radon(i-n+1:i));
            Quart1(j,1) = A(floor((n)/4));
            Quart3(j,1) = A(floor((3*(n))/4));
            Int_QRange(j,1) = Quart3(j,1) - Quart1(j,1);
            Simple_Range(j,1) = Max_values(j,1) - Min_values(j,1);
        else
            n = n+1;
        end
    else
        unique_zip(j,1) = zips(i);
        GM(j,1) = geomean(radon(i-n+1:i));
        AM(j,1) = mean(radon(i-n+1:i));
        Min_values(j,1) = min(radon(i-n+1:i));
        Max_values(j,1) = max(radon(i-n+1:i));
        SDev(j,1) = std(radon(i-n+1:i));
        Median(j,1) = median(radon(i-n+1:i));
        A = sort(radon(i-n+1:i));
        Quart1(j,1) = A(floor((n)/4));
        Quart3(j,1) = A(floor((3*(n))/4));
        Int_QRange(j,1) = Quart3(j,1) - Quart1(j,1);
        Simple_Range(j,1) = Max_values(j,1) - Min_values(j,1);
    end
end

if i+1 == N
    unique_zip(j,1) = zips(i+1);
    GM(j,1) = radon(i+1);
    AM(j,1) = radon(i+1);
    Min_values(j,1) = radon(i+1);
    Max_values(j,1) = radon(i+1);
    SDev(j,1) = radon(i+1);
    Median(j,1) = radon(i+1);
    Quart1(j,1) = radon(i+1);
    Quart3(j,1) = radon(i+1);
    Int_QRange(j,1) = Quart3(j,1) - Quart1(j,1);
    Simple_Range(j,1) = Max_values(j,1) - Min_values(j,1);
end

title={'ZIP','AM','GM','Minimum','Maximum','Standard Deviation','Median','Quartile1','Quartile3','Int Quartile Range','Simple Range'};
xlswrite('zip_code_data.xls',title,'A1:K1');
xlswrite('zip_code_data.xls',unique_zip,'Sheet1','A2');
A.2 Source Code for Computing Various County Statistics from Raw Data

```matlab
clc; clear all;

A = importdata ('U:\RADON_DATA\county_raw_data.csv');
county = upper(A.textdata(:,1));
radon = A.data(:,1);
N = length(county);
count = 1;
for i=2:N
    if ~strcmp(county(i), county(i-1))
        count = count + 1;
    end
end

temp = county(1);
j=1;
unique_county = cell(count, 1);
Median = zeros(count, 1);
AM = zeros(count, 1);
GM = zeros(count, 1);
Quart1 = zeros(count, 1);
Quart3 = zeros(count, 1);
Max_values = zeros(count, 1);
Min_values = zeros(count, 1);
SDev = zeros(count, 1);
Int_QRange = zeros(count, 1);
Simple_Range = zeros(count, 1);

sum = 0.00;
prod = double(1.00);
n=1;

for i=1:N-1
    if strcmp(temp, county(i+1))
        if i+1 == N
            unique_county(j, 1) = county(i+1);
            GM(j, 1) = geomean(radon(i-n+1:i+1));
            AM(j, 1) = mean(radon(i-n+1:i+1));
            Min_values(j, 1) = min(radon(i-n+1:i+1));
```
Max_values(j,1) = max(radon(i-n+1:i+1));
SDev(j,1) = std(radon(i-n+1:i+1));
Median(j,1) = median(radon(i-n+1:i+1));
A = sort(radon(i-n+1:i+1));
Quart1(j,1) = A(ceil((n)/4));
Quart3(j,1) = A(ceil(3*(n)/4));
Int_QRange(j,1) = Quart3(j,1) - Quart1(j,1);
Simple_Range(j,1) = Max_values(j,1) - Min_values(j,1);
else
    n = n+1;
end
else
    unique_county(j,1) = county(i);
    GM(j,1) = geomean(radon(i-n+1:i+1));
    AM(j,1) = mean(radon(i-n+1:i+1));
    Min_values(j,1) = min(radon(i-n+1:i+1));
    Max_values(j,1) = max(radon(i-n+1:i+1));
    SDev(j,1) = std(radon(i-n+1:i+1));
    Median(j,1) = median(radon(i-n+1:i+1));
    A = sort(radon(i-n+1:i+1));
    Quart1(j,1) = A(ceil((n)/4));
    Quart3(j,1) = A(ceil(3*(n)/4));
    Int_QRange(j,1) = Quart3(j,1) - Quart1(j,1);
    Simple_Range(j,1) = Max_values(j,1) - Min_values(j,1);
    n = 1;
    temp = county(i+1);
    j=j+1;
    if i+1 == N
        unique_county(j,1) = county(i+1);
        GM(j,1) = radon(i+1);
        AM(j,1) = radon(i+1);
        Min_values(j,1) = radon(i+1);
        Max_values(j,1) = radon(i+1);
        SDev(j,1) = radon(i+1);
        Median(j,1) = radon(i+1);
        Quart1(j,1) = radon(i+1);
        Quart3(j,1) = radon(i+1);
        Int_QRange(j,1) = Quart3(j,1) - Quart1(j,1);
        Simple_Range(j,1) = Max_values(j,1) - Min_values(j,1);
    end
end

end

end

title={'COUNTY NAME','AM','GM','Minimum','Maximum','Standard Deviation','Median','Quartile1','Quartile3','Int Quartile Range','Simple Range'};
xlswrite('county_data.xls',title,'A1:K1');
xlswrite('county_data.xls',unique_county,'A2');
xlswrite('county_data.xls',AM,'B2');
xlswrite('county_data.xls',GM,'C2');
xlswrite('county_data.xls',Min_values,'D2');
xlswrite('county_data.xls',Max_values,'E2');
xlswrite('county_data.xls',SDev,'F2');
xlswrite('county_data.xls',Median,'G2');
xlswrite('county_data.xls',Quart1,'H2');
xlswrite('county_data.xls',Quart3,'I2');
xlswrite('county_data.xls',Int_QRange,'J2');
xlswrite('county_data.xls',Simple_Range,'K2');
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 Creation 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.

![Training Window Screenshot]

Figure B.3: Screenshot of the training window of the *Neuromodeler* software, while training the fine model of the SMNN technique.
B.4 Testing Window of the *Neuromodeler* Software.

![Testing Window](image)

**Figure B.4:** Screenshot of the testing window of the Neuromodeler software, validating the fine model of the SMNN technique.