PREDICTION MODELS FOR ESTIMATION OF SOIL MOISTURE CONTENT

by

Swathi Gorthi

A thesis submitted in partial fulfillment
of the requirements for the degree

of

MASTER OF SCIENCE

in

Electrical Engineering

Approved:

Dr. Huifang Dou Major Professor	Dr.YangQuan Chen Committee Member
Dr. Jacob Gunther Committee Member	Dr. Mark R. McLellan Vice President for Research and Dean of the School of Graduate Studies

UTAH STATE UNIVERSITY
Logan, Utah

2011

Abstract

Prediction Models for Estimation of Soil Moisture Content

by

Swathi Gorthi, Master of Science

Utah State University, 2011

Major Professor: Dr. Huifang Dou
Department: Electrical and Computer Engineering

This thesis introduces the implementation of different supervised learning techniques for producing accurate estimates of soil moisture content using empirical information, including meteorological and remotely sensed data. The models thus developed can be extended to be used by the personal remote sensing systems developed in the Center for Self-Organizing Intelligent Systems (CSOIS). The different models employed extend over a wide range of machine-learning techniques starting from basic linear regression models through models based on Bayesian framework, decision tree learning, and recursive partitioning, to the modern nonlinear statistical data modeling tools like artificial neural networks. Also, ensembling methods such as bagging and boosting are implemented on all models for considerable improvements in accuracy. The main research objective is to understand, compare, and analyze the mathematical backgrounds underlying and results obtained from different models and the respective improvisation techniques employed.

(85 pages)

Public Abstract

Prediction Models for Estimation of Soil Moisture Content

by

Swathi Gorthi, Master of Science

Utah State University, 2011

Major Professor: Dr. Huifang Dou
Department: Electrical and Computer Engineering

The Center for Self-Organizing Intelligent Systems (CSOIS) has been successful in developing personal remote sensing systems. These systems can be further enhanced to monitor important resources of earth such as soil moisture content. Soil moisture is the major component of the soil in relation to plant growth. Soil water dissolves salts and makes up the soil solution, which is important as medium for supply of nutrients to growing plants.

Usually, measuring such physical quantities would include using derived formulas that mathematically describe the relationships between the parameters involved. With the advancements in computer modeling, other methods such as empirical modeling have evolved which can be used to develop models for measurement of physical quantities such as soil moisture content. In this kind of modeling, empirical data are used to pick a right model, and to calibrate and test it. The relevant data constitutes measured values of predetermined input parameters and the output parameter which is being modeled. The input parameters can be chosen by experience emphasizing the fact that there should be a minimum correlation between them. The important step in such a kind of model development is to choose the techniques to be used in finding an appropriate model. The main objective of the present work lies in understanding the mathematical backgrounds of different advanced techniques

used in empirical model development for measurement of soil moisture content, analyzing the results, and working in the direction of improving those contemporary methods.

Acknowledgments

I express my deep gratitude to my advisor, Dr. HuiFang Dou, and our research group coordinator, Prof. YangQuan Chen, for providing me with this opportunity. I am thankful to all the CSOIS group members and Dr. Bushra Zaman for all the support given by them during my research. I am also thankful to Dr. Gunther and Dr. Moon for answering my questions and guiding me through my research. I feel grateful to USU and UWRL for all the financial support.

I thank all my beloved friends and pals for all the fun and encouragement they gave me during my low times. I extend my gratitude towards my primary school, Sri Gowthami Public School, for providing with the best memories to cheer me up.

Above all, I am thankful to my family for all the confidence and trust they put in me. I wish to include all the names but that will just make the acknowledgments longer than the actual thesis!

Swathi Gorthi

Contents

List of Tables

List of Figures

Acronyms

ABFC	Adaptive Basis Function Construction
AgRISTARS	Agriculture and Resources Inventory Surveys Through Aerospace Remote Sensing
AICC	Akaike's Information Criterion
BIC	Bayesian Information Criterion
CART	Classification and Regression Trees
CSOIS	Center for Self Organizing Intelligent Systems
EF-ABFC	Ensembling Floating Adaptive Basis Function Construction
GCV	Generalized Cross Validation
GPS	Geo-Positioning Systems
LAI	Leaf Area Index
LM	Levenberg-Marquardt
LRS	plus-L minus-R Selection
LST	Land Surface Temperature
MAE	Mean Absolute Error
MARS	Multivariate Adaptive Regression Splines
MESA	Mechatronic and Embedded Systems and Applications
MLP	MultiLayer Perceptron
MSE	Mean Square Error
NDWI	Normalized Difference Water Index
NIR	Near Infra-Red
RMSE	Root Mean Square Error
RVM	Relevance Vector Machines
SAVI	Soil Adjusted Vegetation Index
SDHC	Steepest Descent Hill Climbing
SDR	Standard Deviation Reduction
SFFS	Sequential Floating Forward Selection

SFS	Sequential Forward Selection
SMC	Soil Moisture Content
SMEX02	Soil Moisture Experiments, 2002
STP	Soil Temperature Probes
SVM	Support Vector Machines
UAV	Unmanned Air Vehicles

Chapter 1

Introduction

1.1 Introduction

Precision agriculture is a farming management technique that involves the study of the spatial variations in a crop field using technological tools such as Global Positioning Systems and aerial images. This study can be helpful in estimating fertilizers and other input needs by assessing the local disease and soil conditions in a better way, thus preventing inflexible practices in farming. The benefits of precision agriculture are very valuable in agronomical, environmental, technical and economical perspectives.

Irrigation water management forms a major part of precision agriculture. It involves better assessment of need and availability of soil water level for crop cultivation. The statistical data from the United Nations indicate worldwide, agricultural accounts for 70% of all water consumption, compared to 20% for industry and 10% for domestic use [1]. According to D. L. Miles and I. Broner soil moisture estimation helps in determining the timing of irrigation [2]. Soil moisture is defined as the amount of water level present in the top layers of the soil that interacts with the atmosphere through evaporation and transpiration [3]. Surface soil moisture is the amount of water content present in the top 10cms of the soil layer, whereas the root zone soil moisture is that amount present in the upper 200cms of the soil.

Even though soil moisture is quite small in amount in a specific region, it significantly affects:

- all kinds of hydrological, biological, and biogeochemical processes; and
- weather patterns, runoffs, and erosion; and
- thermal exchange models of land surface and atmosphere.

Power et al. and Stewart et al., in their independent research studies, found strong correlation between crop yield of wheat and soil moisture content availability [4,5]. Machado et al. also associated the crop yield in corn with the soil moisture content [6]. In conclusion, it can be said that estimation of soil moisture can be helpful in irrigation scheduling, crop yield forecasting, drought warnings, and runoff predictions.

1.2 History of Soil Moisture Estimation

Conventionally, in situ soil moisture measurements can be done in either direct or indirect methods. Direct methods involve gravitimetric and volumetric procedures, whereas indirect methods include measurement of water stress under which water is withheld by the soil using the instruments like tensiometers, gypsum blocks, and neutron probes. These methods are cost prohibitive, time and resource consuming. Therefore, many geologists started working on effective mapping techniques, like soil water profiling, instead of relying on above methods for establishing a complete site specific map. Such a research was a principal area of the AgRISTARS Soil Moisture project. Jackson summarized and compared different soil water profile models studied by Schmugge, Kanemasu, Heldrith, Calder, and Saxon [7]. Most of these models incorporate physically-based and empirical factors like precipitation, thermal index, crop water stress, theoretical functions of heat transfer, and crop water stress. These summaries can help further in deciding the input parameters while developing the model of our own.

There are models developed that use the remotely sensed data to profile the soil moisture. However, remote sensing can be used to directly infer soil moisture. Microwave emissivities and infrared data were proven to be highly correlated with the soil moisture. A lot of research has been concentrated in this area during the last two decades. These studies were based on the facts that the reflection spectrum of soil is significantly affected by soil moisture. Especially, the reflectance of soil in the visible and infrared regions is highly related to the soil color, texture, surface roughness and crusting, composition and organic matter, as well as soil moisture.

In particular, many researchers used different spectral regions including gamma radiation, thermal infrared, and passive and active microwave regions. Soil darkens at the wavelengths around 800nm because of absorption of water and internal reflection of radiation. This forms a water absorption band in the wavelength range of 800 - 900nm. Hyperspectral data collected from the multispectral sensors working in the above wavelength range can be used to derive soil moisture measurements. However, all of these remote sensing techniques have their own disadvantages. Microwave radiation has an advantage of penetrating into soil to a wavelength dependant depth, but suffers from low resolution. Multispectral sensors are costly. Of all regions, visible and near infrared are the affordable ones. The remotely sensed data derived from these regions can be used to derive vegetation indices and surface temperature which can in turn be used in developing empirical or theoretical models. Moreover, recent developments in Unmanned Air Vehicles (UAV) studies have reduced the cost of obtaining these visible and near infrared data with no compromise in resolution [8].

"Theoretical models involve complicated scattering phenomena from probabilistic models of soil, vegetation, and terrain whereas empirical models capture relationships among measured variables to estimate geophysical characteristics" [9]. For both kinds of models, in situ data is required to calibrate and validate them. Even though, empirical models are data-driven and are not computationally intensive and complex. With the advancements in machine learning, there are many techniques available nowadays to develop these models.

1.3 Machine Learning and Predictions

"Machine learning, a branch of artificial intelligence, is a scientific discipline that deals with the design and development of algorithms that allow computers to evolve behaviors based on empirical data" [10]. The main objective of machine learning is to estimate the unknown relationship between input and target parameters using known examples. Then, the relationship thus derived can be used in predicting the unknown target values for other values of inputs. The targets can be nominal or numerical. If the targets are nominal, then the problem becomes a classification one while if the target is numerical, the problem is a regression one. The learning task that involves solving such a regression problem is called supervised learning.

The goal of supervised learning is to build a model of the system from the training examples, which can later be used to deduce responses that have yet to be observed. Consider $\{\mathbf{x}_n, y_n\}_{n=1}^{N}$ to be the training dataset with $\boldsymbol{X}$ being the input space and Y being the output space. The objective now is to seek a function $f : X \rightarrow Y$ from a hypothesis space that minimizes the loss associated. The best fit to the underlying function can be chosen as per structural risk minimization or empirical risk minimization. The former chooses a function that controls a bias/variance tradeoff, whereas the latter chooses the one that best fits the training data.

Many algorithms can be used during development of a model in supervised learning, which are developed in different mathematical backgrounds. They are:

- Linear and polynomial regression,
- Basis function construction using adaptive modeling,
- Relevance vector machines using Bayesian framework,
- Support vector machines,
- Regression trees using recursive partitioning or continuous class learning,
- Multilayer perceptrons using neural networks.

Development of models using supervised learning is done following the procedure below:

- After deciding on the input parameters, gather a decent amount of training and test examples;
- Analyze the correlation between the input parameters. Less correlated inputs will give more reliable results;
- Choose one of the algorithms from the above list and decide upon the other parameters required for that algorithm;
- Train the model using the training examples and the chosen algorithm;

- Lastly, check the reliability and accuracy by predicting the target values for the test dataset using the developed model and comparing them with the actual values.

1.4 Research Plan

- Survey of the different empirical models used till date for the estimation of soil moisture content.
- Understand the importance of machine learning techniques among all modeling techniques surveyed.
- Obtain an adequate set of training and test examples.
- Survey of different machine learning algorithms available.
- Study the mathematical backgrounds of surveyed learning algorithms.
- Study different model ensembling techniques along with their pros and cons.
- Implementing the suitable improvement techniques like bagging and boosting for all the different models to make the prediction accuracy better.

1.5 Dataset Used

Bushra's work provided a basis for the present work [11]. The datasets used here are similar to the ones used in that work and they are a part of Soil Moisture Experiments conducted at Ames, Iowa in 2002. The temporal coverage of the data was a one-month period between mid-June and mid-July and the spatial coverage was 41.52°N to 42.2°N, 93.23°W to 93.50°W. The inputs consisted of meteorological parameters such as soil temperature, air temperature and precipitation, vegetation indices such as Soil Adjusted Vegetation Index (SAVI) and Leaf Area Index, and Land Surface Temperature (LST). The output or target parameter is volumetric Soil Moisture Content (SMC).

1.5.1 Study Area

The study area is the Walnut Creek watershed located at Ames, in south-central Iowa, USA. It is a small watershed in the heart of the Corn Belt with an area of about 5,130 hectares and is characterized by fairly level topography and rich soils that developed under prairie and prairie pothole wetlands. More than 80% of this watershed is planted to corn and soybean row crops. Figure 1.1 shows the experimental fields, sampling locations, and topography of the study area [11].

Surface SMC data belonging to 31 sites and 3 dates, i.e. a total of 93 points are available. Some sites had missing data on some dates, such sites were removed from the datasets. So, the points available shrink to 80. These points were used for training the models thus serving as training datasets.

Measurements corresponding to the field number 16, 23, and 24 on three dates were selected as test data. So, there nine points available as test data. These fields were selected as test locations as they had the exact precipitation measurements.

1.5.2 Volumetric Soil Moisture Content

These data result from daily measurements of volumetric SMC (0-6 cm) using a manually inserted probe and handheld reader conducted at 31 moisture sampling sites in the Walnut Creek watershed. The unit for volumetric SMC was cubic meter of water per cubic meter of soil m^3/m^3.

1.5.3 Soil Temperature

Soil temperature was measured using soil temperature probes (STP). The unit for Soil Temperature was degrees Celsius (°C).

1.5.4 Air Temperature

The air temperature for the study area was downloaded from the DAYMET U.S. data center website. The website provides daily surface weather data and climatological summaries based on the latitude and longitude of the location. The hourly precipitation data

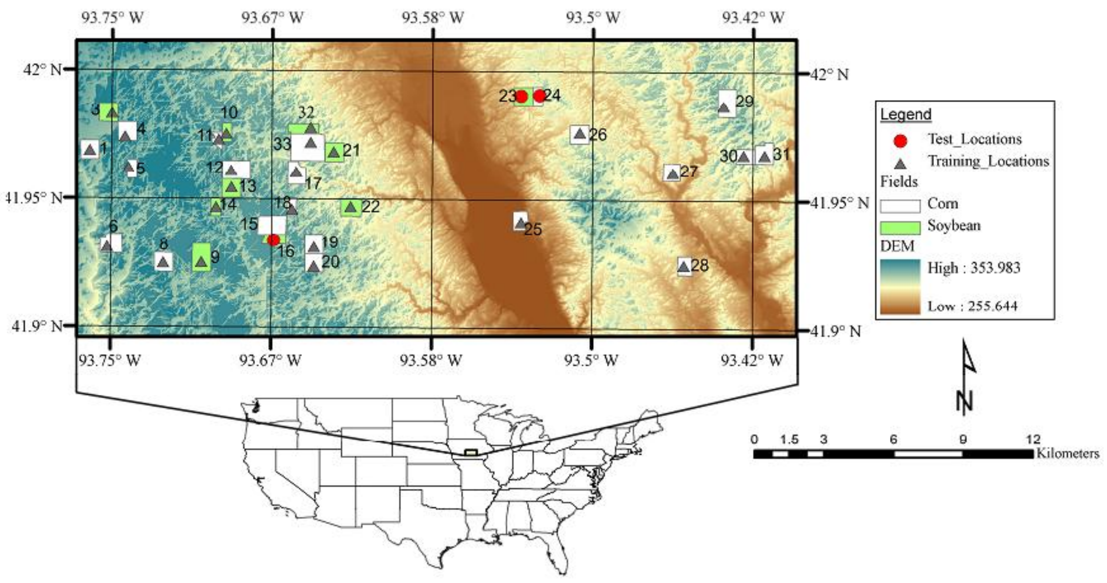

Fig. 1.1: Map of the study area.

was added to get daily data. The precipitation data corresponding to the input points were obtained by creating a spatially interpolated precipitation layer using kriging in ArcGIS.

1.5.5 Precipitation

This data set acquired during SMEX02 from 1 June through 19 August 2002 experiments, includes hourly precipitation data at 20 rain gauge stations distributed throughout the study area. The unit for precipitation was millimeters (mm).

1.5.6 Soil Adjusted Vegetation Index

The spatial layers of SAVI and Normalized Difference Water Index (NDWI) require the reflectance data from visible region and NIR region. These data were obtained from the processed Landsat Thematic Mapper Imagery available along with the SMEX02 datasets. The Landsat imagery was obtained for the days 23rd June, 1st July, and 8th July, 2002. The following equations are used to develop the spatial layers of NDWI and SAVI.

$$SAVI = \frac{(R_{NIR} - R_{RED})(1+L)}{R_{NIR} + R_{RED} + L}, \tag{1.1}$$

$$NDWI = \frac{R_{NIR} - R_{SWIR}}{R_{NIR} + R_{SWIR}}, \tag{1.2}$$

where $R_{NIR}, R_{RED}, R_{SWIR}$ are the apparent reflectance values in the near-infrared, red and short wave infrared wavebands, respectively, and L is the calibration factor.

1.5.7 Leaf Area Index

The spatial layer for LAI is developed using the spatial layer for NDWI as follows:

$$LAI = a \times NDWI + b \times (1 + c \times \exp(d \times NDWI)), \tag{1.3}$$

where $a = 2.88, b = 1.14, c = 0.104, d = 4.1$ are calibration constants [11].

1.5.8 Land Surface Temperature

This dataset was obtained from the processed Landsat 5 and 7 Thematic Mapper Imagery. The unit for LST is degrees Kelvin (°K).

1.5.9 Evaluation of Models

The models are evaluated by determining the Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) on test data for all the models. The MAE and RMSE are given by

$$MAE = \frac{\sum_{i=1}^{Ntest} \left| y^i_{actual} - y^i_{predicted} \right|}{Ntest}, \tag{1.4}$$

$$RMSE = \sqrt{\frac{\sum_{i=1}^{Ntest} \left| y^i_{actual} - y^i_{predicted} \right|^2}{Ntest}}, \tag{1.5}$$

where $Ntest$ is the size of the test dataset.

Chapter 2
Vector Machines

Relevance vector machines and support vector machines are the two kinds of models used in the present work. This chapter explains the theoretical mathematical backgrounds involved in developing these machines.

2.1 Support Vector Machines

The learning algorithm involved in Support Vector Machines (SVMs) comes under supervised learning. It is a widely used classification algorithm. For a simple binary classification problem, we can say that a support vector machine draws a optimal separating hyperplane between the two classes of data. The hyperplane is chosen following the fact that confidence in predictions improves when a point is far from the separating hyperplane that is the criterion is based on margin maximization of two classes in the case of a binary classification problem.

2.1.1 Simple Binary Classification Problem

Assume for a given binary classification problem, the training set of data is available as $D = \{\mathbf{x}_i, y_i\}_{i=1}^{N}, \mathbf{x} \in \mathbb{R}^n, y \in \{-1, 1\}$ as shown in Fig. 2.1. Now considering the problem to be to develop a linear classifier for the dataset, D, let us use the parameters $\boldsymbol{w}$ and b to write the classifier as

$$f(\mathbf{x}) = sgn(\mathbf{w}^T\mathbf{x} + b), \tag{2.1}$$

such that the argument $\mathbf{w}^T\mathbf{x} + b$ decides the class of the point and the parameters $\boldsymbol{w}$ and b are constrained by Eq. (2.2). The benefit of such a notation is that it allows us to explicitly treat the intercept term b separately from other parameters [12]

$$y^i[|\mathbf{w}^T\mathbf{x}^i + b|] \geq 1, \quad i = 1, \ldots, N. \tag{2.2}$$

As explained by Steve R. Gunn in his technical report [13], the hyperplane that optimally separates the given data is the one that minimizes

$$\phi(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2. \tag{2.3}$$

The solution to the minimization problem with the given constraints will be given by the saddle point of the Lagrange functional

$$\phi(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{N} \alpha_i (y^i[\mathbf{w}^T\mathbf{x}^i + b] - 1), \tag{2.4}$$

where $\boldsymbol{\alpha}$ are the Lagrange multipliers. The dual form of this Lagrange functional is given by

$$\max_{\boldsymbol{\alpha}} \left(\min_{\mathbf{w},b} \phi(\mathbf{w}, b, \boldsymbol{\alpha}) \right). \tag{2.5}$$

The Lagrange multipliers are the solution to the minimization problem

$$\boldsymbol{\alpha}^* = arg \min_{\boldsymbol{\alpha}} \frac{1}{2} \sum_{i=1}^{N} \sum_{j=i}^{N} \alpha_i \alpha_j y_i y_j \mathbf{x_i}^T \mathbf{x_j} - \sum_{k=1}^{N} \alpha_k, \tag{2.6}$$

$$\text{such that} \quad \alpha_i \geq 0 \quad i = 1, \ldots, N \quad \text{and} \quad \sum_{j=1}^{N} \alpha_j y_j = 0.$$

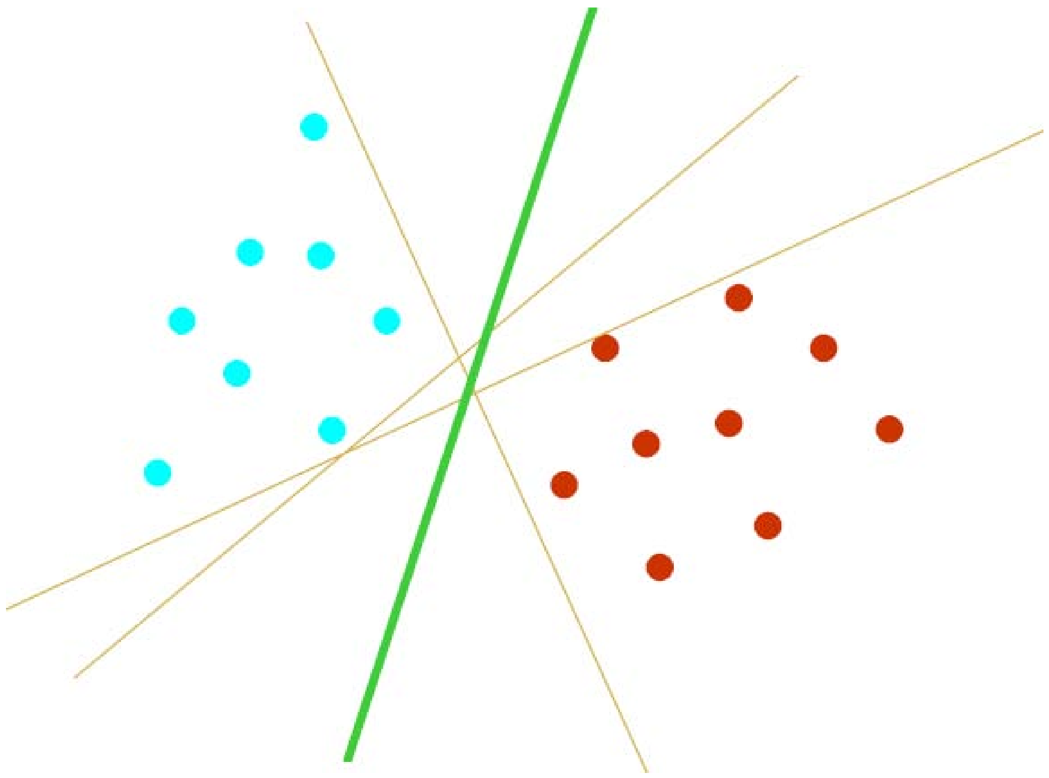

Fig. 2.1: Optimal separating hyperplane.

With the Lagrange multipliers obtained using Eq. (2.6), the parameters $\boldsymbol{w}$ and b can be obtained as in Eq. (2.7) the classifier is obtained as in Eq. (2.1)

$$\mathbf{w}^* = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i$$
$$b^* = -\frac{1}{2}(\mathbf{w}^{*T}(\mathbf{x}_r + \mathbf{x}_s)), \tag{2.7}$$

where $\mathbf{x}_r$ and $\mathbf{x}_s$ are any support vector from each class satisfying

$$\alpha_r, \alpha_s > 0, \quad y_r = -1 \quad \text{and} \quad y_s = 1.$$

The support vectors having nonzero Lagrange multipliers are those points $\mathbf{x}^i$ that satisfy the Kuhn-Tucker conditions given by Eq. (2.8)

$$\begin{aligned} \alpha_i(y^i[\mathbf{w}^T\mathbf{x}^i + b] - 1) &= 0, \quad i = 1, \ldots, N \\ \text{i.e.} \quad y^i[\mathbf{w}^T\mathbf{x}^i + b] &= 1. \end{aligned} \tag{2.8}$$

2.1.2 Support Vector Regression

Regression problems are very common just as the classification ones. In the classification problems, the unknown target variable to be predicted takes a class value being a nominal kind whereas, in a regression problem, the unknown target variable takes a numerical value. Hence, the theory behind the development of support vector machines for regression problems involves some significant changes to be noticed. Note that the argument of the minimization problem in support vector machines developed for classification involved finding a optimal hyperplane based on a distance measure. Analogously, in regression problems support vector machines are introduced with an alternative loss function.

Let us formulate a simple linear regression problem by assuming the availability of the training set of data as $D = \{\mathbf{x}_i, y_i\}_{i=1}^N, \mathbf{x} \in \mathbb{R}^n, y \in \mathbb{R}$. Now, since the problem is a linear one, we can approximate the underlying function by a linear function using the parameters $\boldsymbol{w}$ and b as

$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b. \tag{2.9}$$

Note that the optimal regressors depend on the parameters $\boldsymbol{w}$ and b given by Eq. (2.10) which are derived using Karush-Kuhn-Tucker conditions

$$\begin{aligned} \mathbf{w} &= \sum_{i=1}^{N} \beta_i \mathbf{x}_i \\ b &= -\frac{1}{2}(\mathbf{w}^T(\mathbf{x}_r + \mathbf{x}_s)), \end{aligned} \tag{2.10}$$

where evaluation of $\boldsymbol{\beta}$ is done depends on the type of loss function we use.

- **ϵ-insensitive loss function**

$$L_\epsilon = \begin{cases} 0 & \text{for} \qquad |f(\mathbf{x}) - y| < \epsilon \\ |f(\mathbf{x}) - y| - \epsilon & \text{otherwise,} \end{cases} \tag{2.11}$$

$\boldsymbol{\beta}$ can then be determined using:

$$\boldsymbol{\beta} = arg\min_{\boldsymbol{\beta}} \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\beta_i\beta_j\mathbf{x}_i^T\mathbf{x}_j - \sum_{i=1}^{N}\beta_i y_i, \tag{2.12}$$

$$\text{such that} \quad -C \leq \beta_i \leq C, \quad i = 1, \ldots, N \quad \text{and} \quad \sum_{i=1}^{N}\beta_i = 0,$$

where C is a regularization parameter.

- **Quadratic loss function**

$$L_{quad} = (f(\mathbf{x}) - y)^2, \tag{2.13}$$

$\boldsymbol{\beta}$ can then be determined using

$$\boldsymbol{\beta} = arg\min_{\boldsymbol{\beta}} \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\beta_i\beta_j\mathbf{x}_i^T\mathbf{x}_j - \sum_{i=1}^{N}\beta_i y_i + \frac{1}{2C}\sum_{i=1}^{N}\beta_i^2, \tag{2.14}$$

$$\text{such that} \quad \sum_{i=1}^{N}\beta_i = 0 \quad \text{and C is a regularization parameter.}$$

2.1.3 Nonlinear Regression

Usually, most of the problems we come across in our daily life do not supply a linearly separable data so that, we can follow the above explained methods to develop the models. Such kind of problems appear when the data we have belong to a higher-dimensional space.

The use of kernel comes into picture in such cases, so that they allow to map the data into high-dimensional feature space before feeding them to a linear model in order to increase the computational power of linear machines [14]. Thus, we can extend linear hypotheses to nonlinear ones implicitly using kernel functions so that operations can be performed in the input space rather than the potentially high-dimensional feature space.

The fact behind the use of kernels is that there exists a kernel in input space equivalent to an inner product in inner space

$$K(x, z) = \langle \phi(x), \phi(z) \rangle,$$

only if K is a symmetric positive definite function that satisfies Mercer's conditions. Some of the various kinds of kernels that can be used are tabulated in Table 2.1.

So, the inner product between $\mathbf{x}_i$ and $\mathbf{x}_j$ are all replaced by these kernel functionals and the optimization problems are correspondingly evaluated. There is a great amount of choice to choose a kernel function from many kinds available. The best choice is always better to know on a trial and error basis.

2.1.4 Pros and Cons of SVMs

The advantages of Support Vector Machines are:

- The usage of kernel functional for dealing with higher-dimensional data induces flexibility in drawing the margins while development of the model;
- The implicit nonlinear transformations present in the kernel functionals avoid the requirement of any prior assumptions regarding the underlying functions;
- Any bias in the training samples can be overlooked in SVMs by choosing appropriate regularization parameters, C and kernel size, θ which ultimately makes SVMs robust;
- SVMs provide unique solutions as the optimality problem developed is convex.

The disadvantages of Support Vector Machines are:

Table 2.1: Some useful kernels.

#	Name of kernel	$K_I(\|x-z\|)/K_I(0)$
1	Circular	$\frac{2}{\pi}\arccos\left(\frac{\|x-z\|}{\theta}\right) - \frac{2}{\pi}\frac{\|x-z\|}{\theta}\sqrt{1-\left(\frac{\|x-z\|}{\theta}\right)^2}$ if $\|x-z\| < \theta$
	positive definite in $\mathbb{R}^2$	zero otherwise
2	Spherical	$1 - \frac{3}{2}\frac{\|x-z\|}{\theta} + \frac{1}{2}\left(\frac{\|x-z\|}{\theta}\right)^3$ if $\|x-z\| < \theta$
	positive definite in $\mathbb{R}^3$	
3	Rational quadratic	$1 - \frac{\|x-z\|^2}{\|x-z\|^2+\theta}$
	positive definite in $\mathbb{R}^d$	
4	Exponential	$\exp\left(-\frac{\|x-z\|}{\theta}\right)$
	positive definite in $\mathbb{R}^d$	
5	Gaussian	$\exp\left(-\frac{\|x-z\|^2}{\theta}\right)$
	positive definite in $\mathbb{R}^d$	
6	Wave	$\frac{\theta}{\|x-z\|}\sin\left(\frac{\|x-z\|}{\theta}\right)$
	positive definite in $\mathbb{R}^3$	

- SVM being a nonparametric method suffers from lack of transparency of results;
- Number of support vectors required is a linear function of the size of the training set;
- The predictions are not probabilistic, and hence there is no estimate of the uncertainty associated with the prediction.

2.1.5 Tools Used

A MATLAB toolbox is implemented by Steve R. Gunn for building Support Vector Machines [13]. The functions in this toolbox that are developed for regression purposes have provided the basis for developing the SVM models in the present work.

2.1.6 Results

Figures 2.2 and 2.3 show the predicted values and actual values of the soil moisture content test data and the training data, respectively. The Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) for this model are 0.0499 and 0.0546, respectively.

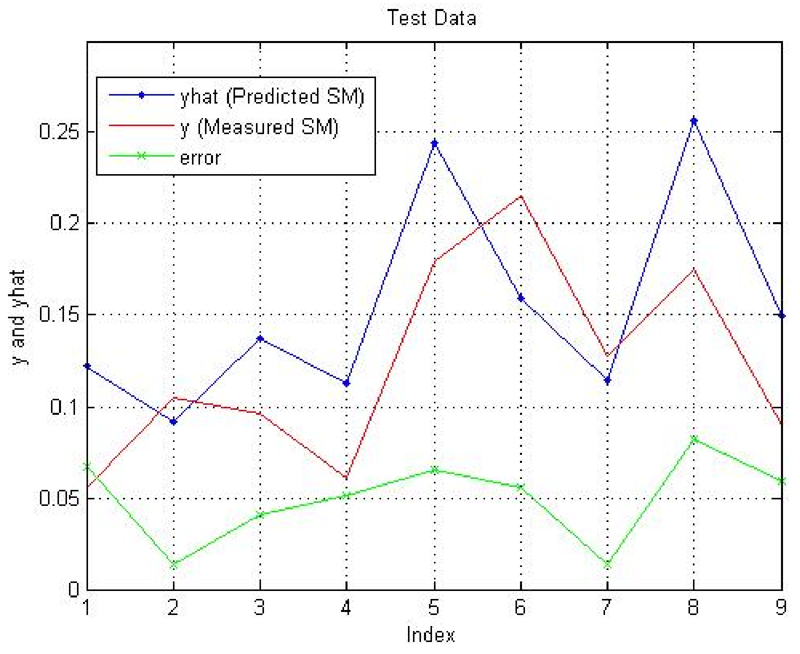

Fig. 2.2: Predicted and actual target values and absolute error for test data using SVMs.

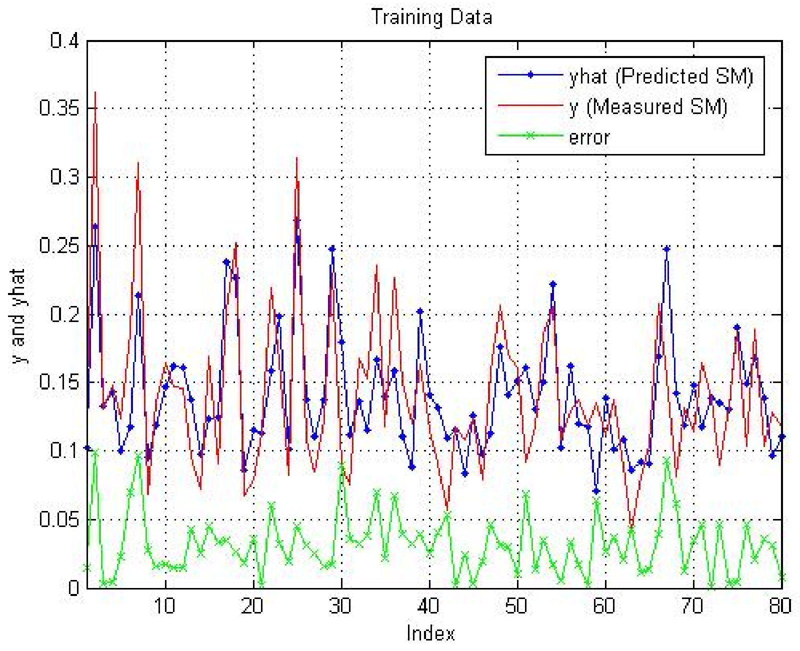

Fig. 2.3: Predicted and actual target values and absolute error for training data using SVMs.

2.2 Relevance Vector Machines

Predictions done using SVMs are point estimates and are not probabilistic. If we desire to estimate the target variables along with a measure of the uncertainty related with the predictions, we should opt for estimation of the conditional distribution of $p(y/\mathbf{x})$. Bayesian framework introduces a technique to estimate conditional distributions under similar scenarios.

2.2.1 Sparse Bayesian Learning

Considering that we are given with a training set of data $D = \{\mathbf{x}_i, y_i\}_{i=1}^{N}$, the goal is to model y. In order to do that, we can assume that the training set of data is actually sampled from the model with additive noise as explained by Tipping [15]

$$y = f(\mathbf{x}; \mathbf{w}) + \epsilon, \tag{2.15}$$

where $\epsilon \sim \mathcal{N}(0, \sigma^2)$ and $f(\mathbf{x}_n; \mathbf{w}) = \sum_{i=1}^{N} w_i K(\mathbf{x}, \mathbf{x}_i) + w_0$ and $K(\mathbf{x}, \mathbf{x}_i)$ is any kernel function listed in Table 2.1.

Assuming that we have independent, identically distributed samples from the population of the input space, we can write the likelihood of the complete dataset as

$$p(\mathbf{y}/w, \sigma^2) = (2\pi\sigma^2)^{-N/2} \exp\left\{-\frac{1}{2\sigma^2}\|\mathbf{y} - \boldsymbol{\phi}\boldsymbol{w}\|^2\right\}, \tag{2.16}$$

where
$$\begin{aligned} \mathbf{y} &= [y_1, \ldots, y_N]^T \\ \mathbf{w} &= [w_0, \ldots, w_N]^T \\ \boldsymbol{\phi} &= [\boldsymbol{\phi}(\mathbf{x}_1), \ldots, \boldsymbol{\phi}(\mathbf{x}_N)]^T \\ \boldsymbol{\phi}(\mathbf{x}_n) &= [1, K(\mathbf{x}_n, \mathbf{x}_1), \ldots, K(\mathbf{x}_n, \mathbf{x}_N)]^T. \end{aligned}$$

In order to avoid over fitting, we impose some additional constraints on the parameters by defining a prior distribution on it as

$$p(\mathbf{w}/\boldsymbol{\alpha}) = \prod_{i=0}^{N} \mathcal{N}(w_i/0, \alpha_i^{-1}) = \prod_{i=0}^{N} \frac{\alpha_i^2}{2} \exp - \frac{\alpha_i}{2} w_i^2, \tag{2.17}$$

where $\boldsymbol{\alpha}$ is vector of N+1 hyper parameters. The prior $p(\mathbf{w}/\boldsymbol{\alpha})$ is Gaussian and has little preference for sparsity. So to attain the Bayesian consistency, hyper priors are defined over $\boldsymbol{\alpha}$ and noise variance, σ^2 as in Eq. (2.18) which is referred to as hierarchical prior [16]

$$\begin{aligned} p(\boldsymbol{\alpha}) &= \prod_{i=0}^{M} Gamma\,(\gamma_i/a, b) \\ p(\beta) &= Gamma(\beta/c, d), \end{aligned} \tag{2.18}$$

$$\text{where} \quad \beta = \sigma^{-2}, \quad Gamma(z/a, b) = \Gamma(a)^{-1} b^a z^{(a-1)} e^{-ba} \quad \text{and} \quad \Gamma(a) = \int_0^\infty t^{a-1} e^{-t}\, \mathrm{d}t.$$

In order to make the hyper priors noninformative, they are made uniform by setting them to a fixed value mostly 0 or some other small value. This formulation of prior distributions is a type of automatic relevance determination prior explained by Mackay [17]. The name for this model comes actually from the concept of hyper parameters. Individual hyper parameters support groups of weights associated with each input dimension $\boldsymbol{x}$. Relevance vectors are actually the the points present at some distance from the decision boundary and appear more prototypical in character. The weights associated with the other "irrelevant"vectors or input dimensions are made zero by using broad prior over the hyper parameters that allows the posterior probability mass to concentrate at very large values of some of these $\boldsymbol{\alpha}$ variables. The assignment of each hyper parameter to each weight is what gives the Relevance Vector Machines (RVMs) their sparsity properties.

2.2.2 Procedure for Predicting Target Variables

Now that we have all the information regarding the priors and the hierarchical priors, for a given new point $\mathbf{x}_*$, the value for the target variable can be predicted as

$$p(y_*/\mathbf{y}) = \int p(y_*/\mathbf{w}, \boldsymbol{\alpha}, \beta) p(\mathbf{w}, \boldsymbol{\alpha}, \beta/\mathbf{y}) \, \mathrm{d}\mathbf{w} \, \mathrm{d}\boldsymbol{\alpha} \, \mathrm{d}\beta. \tag{2.19}$$

Now, Bayesian inference proceeds by decomposing one of the terms in the integrand of Eq. (2.19), $p(\mathbf{w}, \boldsymbol{\alpha}, \beta)$ as

$$p(\mathbf{w}, \boldsymbol{\alpha}, \beta/\mathbf{y}) = p(\mathbf{w}/\mathbf{y}, \boldsymbol{\alpha}, \beta) p(\boldsymbol{\alpha}, \beta/\mathbf{y}). \tag{2.20}$$

Now, if we further decompose the posterior distribution over weights, combine and simplify the expressions obtained in the process, we can see by inspection that it is actually a Gaussian distribution with posterior mean and covariance given by

$$\begin{aligned} \boldsymbol{\mu} &= \beta \boldsymbol{\Sigma} \boldsymbol{\Phi}^T \mathbf{y} \\ \boldsymbol{\Sigma} &= \left(\beta \boldsymbol{\Phi}^T \boldsymbol{\Phi} + \mathbf{A}\right)^{-1}, \end{aligned} \tag{2.21}$$

where $\quad \mathbf{A} = \mathrm{diag}(\alpha_0, \alpha_1, \ldots, \alpha_N)$.

In order to evaluate $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$, we need to know the proper values of the hyper parameters, $\boldsymbol{\alpha}$ and β that maximize the second factor in Eq. (2.20), which is proportional to $p(\mathbf{y}/\boldsymbol{\alpha}, \beta) p(\boldsymbol{\alpha}) p(\beta)$ [18]. Assuming that the hyper priors to be uniform, the factors $p(\boldsymbol{\alpha})$ and $p(\beta)$ can be neglected thus leaving the term $p(\mathbf{y}/\boldsymbol{\alpha}, \beta)$ to be maximized. It is computable and given by

$$\begin{aligned} p(\mathbf{y}/\boldsymbol{\alpha}, \beta) &= \int p(\mathbf{y}/\mathbf{w}, \beta) p(\mathbf{w}/\boldsymbol{\alpha}) \\ &= (2\pi)^{-\frac{N}{2}} \left|\beta^{-1} + \boldsymbol{\Phi}\mathbf{A}^{-1}\boldsymbol{\Phi}^T\right|^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}\mathbf{y}^T \left(\beta^{-1}\mathbf{I} + \boldsymbol{\Phi}\mathbf{A}^{-1}\boldsymbol{\Phi}^T\right)^{-1} \mathbf{y}\right\}. \end{aligned} \tag{2.22}$$

Optimization of the marginal likelihood function in Eq. (2.22) will not result in a closed form expression for the hyper priors $\boldsymbol{\alpha}$ and β. So, iterative re-estimates for them are calculated through evidence approximation or expectation maximization. The latter technique is the one used by Tipping in his paper [15]. Both of the techniques have been exploited in the present work.

- **Evidence Approximation**

  The maximization of the marginal likelihood is achieved by considering the logarithmic of the evidence (Eq. (2.22)) and differentiating it. The resultant expression is then equated to zero to obtain the reestimation expressions for the hyper priors as [19]

$$\gamma_i = 1 - \alpha_i \Sigma_{ii}, \tag{2.23}$$

$$\alpha_i^{new} = \frac{\gamma_i + 2a}{\mu^2 + 2b}, \tag{2.24}$$

$$\beta^{new} = \frac{N - \sum_i \gamma_i + 2c}{\|\mathbf{y} - \boldsymbol{\Phi}\boldsymbol{\mu}\|^2 + 2d}. \tag{2.25}$$

- **Expectation Maximization**

  Another strategy to maximize the expression in Eq. (2.22) is to treat the weights as hidden variables and maximize the expectation of the logarithm of the marginal likelihood function with respect to the distribution over the weights given the data and hidden variables as explained by Dempster et al. [20]. Using this technique, the updates for the hyper priors will be

$$\gamma_i = 1 - \alpha_i \Sigma_{ii}, \tag{2.26}$$

$$\alpha_i^{new} = \frac{1 + 2a}{\Sigma_{ii} + \mu_i^2 + 2b}, \tag{2.27}$$

$$\beta^{new} = \frac{N + 2c}{\|\mathbf{y} - \boldsymbol{\Phi}\boldsymbol{\mu}\|^2 + \sigma^{2^{old}} \sum_i \gamma_i + 2d}. \tag{2.28}$$

The learning algorithm proceeds by repeated application of the any one of above reestimation rules until some suitable convergence criteria is met with. As we proceed through these iterations, we can see some of the $\boldsymbol{\alpha}$ values approaching infinity, thus confirming that the corresponding weights are zero. So, the corresponding basis functions are pruned and sparsity is realized.

Solving the maximization problem in the above procedure provides us with the values of $\boldsymbol{\alpha}$ and β that maximize the objective function, thus allowing us to compute the predictive distribution using Eq. (2.19) giving

$$p(y_*/\mathbf{y}, \boldsymbol{\alpha}, \sigma^2) = \mathcal{N}(y_*/m, \sigma_*^2), \tag{2.29}$$

$$\text{where} \quad m = \boldsymbol{\mu}^T \boldsymbol{\phi}(\mathbf{x}_*) \quad \text{and} \quad \sigma_*^2 = \sigma^2 + \boldsymbol{\phi}(\mathbf{x}_*)^T \boldsymbol{\Sigma} \boldsymbol{\phi}(\mathbf{x}_*).$$

2.2.3 Pros and Cons of RVMs

The advantages of the Relevance Vector Machines are:

- The basic motivation for the development of these models is to estimate the uncertainty associated with every prediction unlike the support vector machines;
- The number of kernel functions required is dramatically lesser than those required for SVMs even though the performance is similar;
- The fully marginalized probability of a given model and its approximation is a measure of the merit of the model.

The disadvantages of Relevance Vector Machines are:

- There is inverse operation required to be performed on a covariance matrix which may involve an algorithm computational complexity of order N^3, where N is the training sample size.

2.2.4 Tools Used

SparseBayes is a package of MATLAB functions designed by Micheal E. Tipping to implement an efficient learning algorithm for Sparse Bayesian models. The functions in this toolbox are edited appropriately and used for developing the RVM models in the present work. The functions in this toolbox were previously edited by Zaman [11] and they are further edited to make them appropriate for the present work. The *SparseBayes* distribution comes with a basic user manual [21].

2.2.5 Results

Figures 2.4 and 2.5 show the predicted values and actual values of the soil moisture content test data and training data, respectively. The Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) for this model are 0.0279 and 0.0323, respectively.

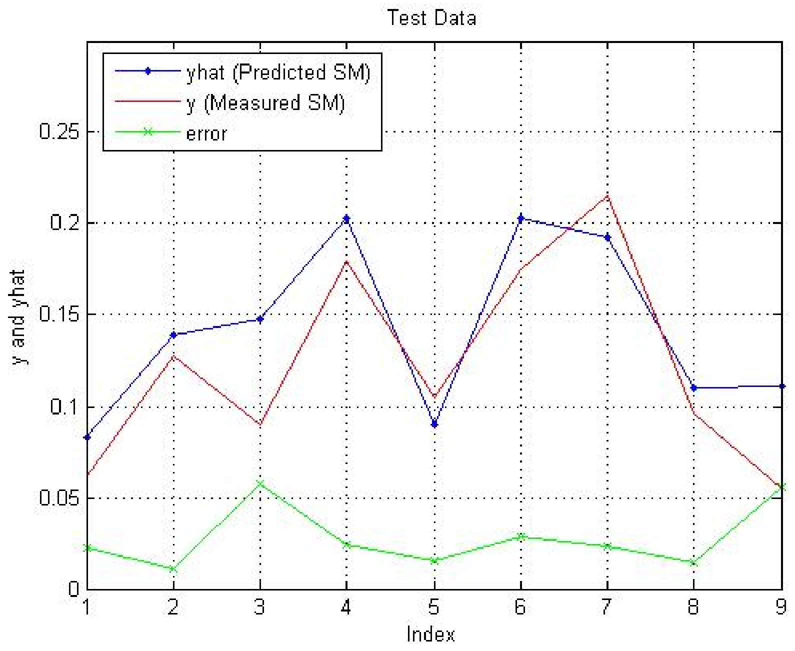

Fig. 2.4: Predicted and actual target values and absolute error for test data using RVMs.

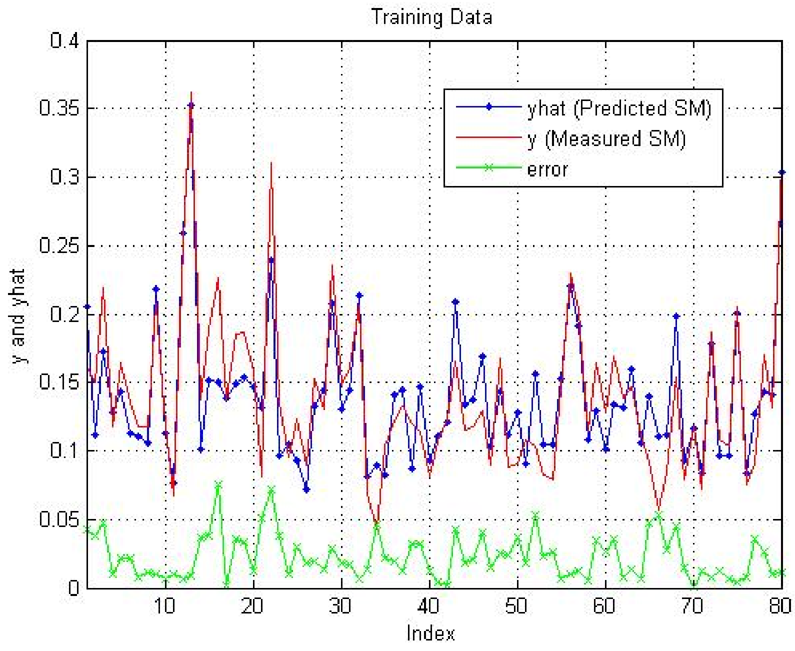

Fig. 2.5: Predicted and actual target values and absolute error for training data using RVMs.

Chapter 3
Tree-Based Learning

Tree-structured classification and regression are the alternative approaches to classification and regression that are not based on assumptions of normality and user-specified model statements. Also, unlike any other nonparametric kernel based methods, the resultant tree structures can be relatively simple functions of the input variables. This chapter explains the mathematical backgrounds of the models using Multivariate Adaptive Regression Splines (MARS) and other continuous learning algorithms like M5 and M5$'$.

3.1 Multivariate Adaptive Regression Splines

This method provides a flexible regression modeling method for high-dimensional data, developed by Friedman [22]. This model works by generating products of spline basis functions, where the data is automatically used to determine the number of basis functions and the parameters associated with them. The motivation behind this model development rises from the recursive partitioning approach. These models are developed in adaptive computation framework. Algorithms indulging in adaptive computation dynamically adjust their strategy to account for the behavior of the function to be approximated. Adaptive algorithms for function approximation have their roots back in recursive partitioning and projection pursuit.

3.1.1 Projection Pursuit Regression

Projection pursuit uses univariate and arbitrary additive functions of linear combinations of the input variables to attain an approximation to the underlying unknown function as

$$\hat{f}(\mathbf{x}) = \sum_{m=1}^{M} \left(\sum_{i=1}^{N} \alpha_{im} \mathbf{x}_i \right). \tag{3.1}$$

The one-dimensional arguments of the functions are not specified. Instead, they are jointly optimized along with the coefficients used in forming the linear combination so that they best fit the data. This kind of approximation can be viewed as low-dimensional expansion of a high-dimensional function. By choosing a proper value for M, close fits for many classes of functions can be developed by using this approximation. Equation (3.1) can be expressed as

$$\hat{f}(\mathbf{x}) = \sum_{j=1}^{J} \hat{g}_j(\mathbf{z}_j), \tag{3.2}$$

where $\mathbf{z}_j$ consists of a preselected subset of the input variables. The above J number of functions used for approximation are in turn obtained through nonparametric methods involving backfitting algorithms like

$$\{\hat{g}_j(\mathbf{x})\}_i^J = arg \min_{g_j} \sum_{i=1}^{N} \left[y_i - \sum_{j=1}^{J} g_j(\mathbf{z}_{ij}) \right]^2 . \tag{3.3}$$

The benefits of this kind of modeling are:

- This method can be used to approximate a wide variety of functions;
- Affine equivariance, that is the solution being invariant of any kind of nonsingular affine transformation of the original input or explanatory variables.

The corresponding disadvantages of this kind of model building are:

- Even simple functions require large M for better approximations;
- When the variables are inter-dependent on each other, the additive forms cannot be separated from these effects.

3.1.2 Recursive Partitioning

Let the input space, D, be partitioned into M disjoint subregions, $\{R_m\}_1^M$. Then, the model using recursive partitioning takes the form

$$\text{if} \quad \mathbf{x} \in R_m, \text{then} \quad \hat{f}(\mathbf{x}) = g_m\left(\mathbf{x}/\{a_j\}_1^P\right), \tag{3.4}$$

where g_m are parametric functions usually chosen to be constants which facilitates to represent the model as a simple binary tree and the interpretation becomes easier, and P is the number of parameters used to represent those parametric functions.

The procedure of model building now involves, partitioning the input domain, D, into optimal subregions and simultaneously estimating the parametric functions associated with those subregions. This partitioning of subregions represents the fact that, even though the underlying function may strongly depend on a large number of variables globally, the dependence is confined to only a small number of different variables in different regions. This methodology is analogous to subset selection. The partitioning is achieved by recursive splitting of previous subregions, starting from the entire input space. Every time a split is done, a region is divided into two daughter subregions following a goodness of fit criterion on the resulting approximation in Eq. (3.4). The subregions are then combined in the backward stage until an optimal set in terms of lack of fit and number of subregions is obtained.

The advantages of this procedure are:

- This model building uses simple piecewise constant approximation,
- It takes benefit of low local dimensionality of functions,
- The model developed is easy to interpret.

The limitations of this procedure are:

- The approximating function is discontinuous at the subregion boundaries,
- Discontinuities at the boundaries limit the accuracy of the model,
- Approximations become difficult when the dominant interactions involve a small fraction of total number of variables.

In order to make the recursive partitioning theory more related to spline construction, the geometrical view of the regions and splitting can be explained with the arithmetic notions of adding and multiplying. As a starting step to such a notion, we can express the approximation as

$$\hat{f}(\mathbf{x}) = \sum_{m=1}^{M} \alpha_m B_m(\mathbf{x}) \quad \text{where} \quad B_m(\mathbf{x}) = I\left[\mathbf{x} \in R_m\right]. \tag{3.5}$$

The forward stepwise regression procedure for recursive partitioning procedure is outlined in Appendix A.

The limitations of recursive partitioning and projection pursuit can be overcome by making the following changes, which is the primary motivation behind the development of MARS:

- Replacing the indicator function in Eq. (3.5) by a truncated power spline function;
- Not removing the parent basis function after it is split thereby making it and both its daughters eligible for further splitting;
- Restricting the product associated with each basis function to factors involving distinct predictor variables.

Incorporating the above mentioned modifications into the algorithm, the MARS algorithm can be implemented in two parts, one for the forward stepwise part of the MARS strategy and the other part dealing with one at a time backward stepwise procedure to delete less contributing basis functions. Note that deleting the basis functions on a one-at-a-time basis, will not produce zero response as it did for the usual recursive partitioning algorithm because here the subregions formed are not disjoint but overlapping.

The forward stepwise and backward stepwise algorithms for the MARS model development procedure are outlined in Appendix B.

3.1.3 Pros and Cons of MARS

Since MARS is built up on the notion of recursive partitioning and projection pursuit, it also shares their advantages. Apart from those, MARS develops subregions that have first order continuities and also there is no prohibition as such, there should be a single split on one variable which allows to approximate local additive dependencies. The limitation of MARS is that its computational requirements grow rapidly with dimensionality which in turn limits its capability to deal with dimensions above 20.

3.1.4 Tools Used

ARESLab is a MATLAB toolbox for building piecewise-linear and piecewise-cubic regression models using the MARS technique developed by Gints Jekabsons. The functions in this toolbox are edited appropriately and used for developing the MARS models in the present work. The *ARESLab* distribution comes with a basic user manual [23].

3.1.5 Results

Figures 3.1 and 3.2 show the predicted values and actual values of the soil moisture content for test data and training data, respectively. The Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) for this model are 0.0408 and 0.0484, respectively.

3.2 Continuous Class Learning

Many problems dealt in machine learning are related to classification, which involves prediction of a nominal variable. Building trees was firstly used for classification purpose through class learning. Then Quinlan proposed an algorithm called M5 for developing model trees for prediction of numerical continuous variables [24]. These model trees are similar to regression trees build by Classification and Regression Trees (CART), except the fact that regression trees have values at their leaves, whereas model trees have multivariate linear models at their leaves.

These models are built by combining instance-based and model-based learning, a technique proposed by Quinlan [25]. Instance-based learning involves forming prototypes from the training set of data and then a prediction for a new case is made by finding similar prototypes and using their classes in some way. On the other hand, learning methods in model-based techniques construct explicit generalizations of the training cases. This section explains how the model trees are built through M5 and M5$'$ algorithms and also how

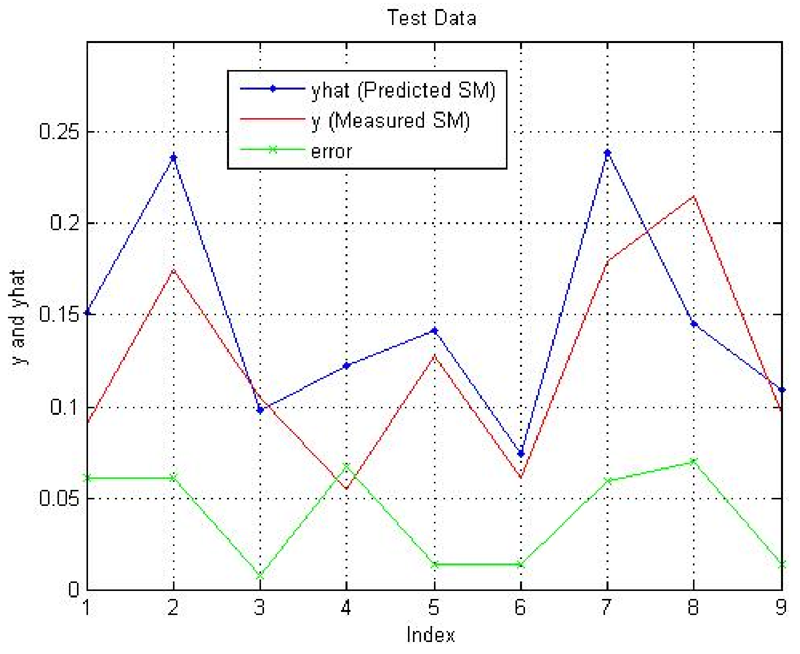

Fig. 3.1: Predicted and actual target values and absolute error for test data using MARS.

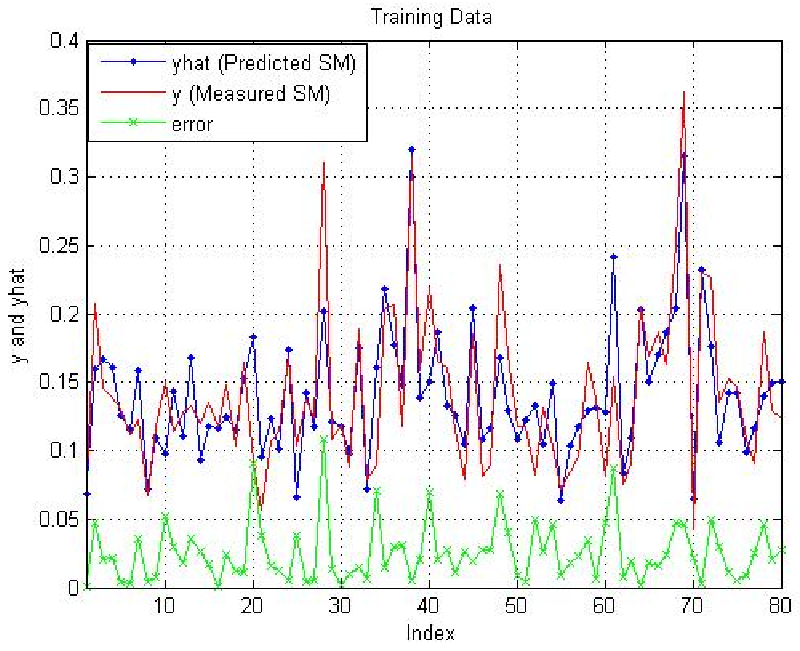

Fig. 3.2: Predicted and actual target values and absolute error for training data using MARS.

predictions are done through model trees by combining instance-based and model-based learning.

3.2.1 M5 Algorithm

Tree-based models are constructed by the divide and conquer method. The algorithm proceeds by assigning the whole training set to a single leaf or by splitting it into subsets in a way similar to recursive partitioning. This process is recursively applied to produce over elaborate structures. To reduce the over size of trees thus formed, they are pruned back. The splitting is done based on the outcomes of a test. Let T denote the set of training examples and T_i denote the subset of cases that have the i^{th} outcome of the potential test. Let $sd(T_i)$ denote the standard deviation of the target values of the examples in T_i. M5 calculates a measure of error in the form of Standard Deviation Reduction (SDR) as denoted in Eq. (3.6) and uses it to decide the best test helpful in the evaluation of the tree developed. The splitting ceases when the class values of all the instances that reach a node vary slightly or only a few instances remain.

$$SDR = sd\,(\mathrm{T}) - \sum_{i} \frac{|\mathrm{T}_i|}{|\mathrm{T}|} \times sd\,(\mathrm{T}_i) \tag{3.6}$$

Details

- In order to avoid underestimation of error, M5 multiplies the average of the absolute difference between the actual and predicted target values by $(n+\nu)/(n-\nu)$, where n is the number of training cases and ν is the number of parameters in the model. This increases the estimated error of model with many parameters constructed from small numbers of cases.

- During the initial splitting procedure, a node is not split if it represents very few examples or its values vary only slightly.

- At each node of the model tree, a multivariate linear model is constructed using standard regression techniques upon a restricted number of attributes referenced by tests.

- Linear models are obtained as above and they are simplified by eliminating parameters to minimize estimated error. M5 algorithm uses greedy search methods to remove variables that contribute very little to the model.

- Pruning of the model tree involves evaluation of each nonleaf node starting from the bottom. The final model at each leaf node is selected to be the one of the simplified linear model or the model subtree depending on which has low estimated error.

- A smoothing process is adopted to improve the prediction accuracy of the model trees. The predicted value for a new case at the appropriate leaf is adjusted so that it reflects the values at the nodes along the path from the root to the leaf. To serve this purpose, the value at that leaf is backed up from the leaf to the root as

$$PV(S) = \frac{n_i \times PV(S_i) + k \times M(S)}{n_i + k}, \tag{3.7}$$

  where S_i is the branch of subtree S followed by the new case, n_i is the number of training cases at S_i, $PV(S_i)$ is the predicted value at S_i, $M(S)$ is the value given by the model at S, k is the smoothing constant.

3.2.2 M5′ Algorithm

The drawbacks of the model trees developed by M5 are of very big size which is overcome in M5′ algorithm developed by Wang and Witten [26]. In order to improve the speed of the algorithm in M5′, the splits at all nodes are made binary; that means they involve either a continuous valued attribute or a synthetic binary one. This is based on the proof that the best split at a node for an enumerated variable with k values is one of the $k-1$ positions obtained by ordering the average class values for each enumerated variable. To avoid the automatic favoring of the attributes with large different values in M5′, the SDR is multiplied by a factor (β) that is unity for every binary split and decays exponentially as the number of different values increases.

The decision tree inducing algorithm used by M5′ algorithm is same as the one used by its previous version except the following changes:

- The tree is built by minimizing the intra-subset variation in the class values down each branch instead of maximizing the information gain at each interior node;
- While pruning back to an interior node, consideration is given to replacing that node by a regression plane instead of a constant value.

The problem of missing values can come into picture, when the examples are to be divided into subsets based on the value of the attribute which is selected for splitting. To account for the missing values, a further modification is made to SDR in Eq. (3.6) as below

$$\mathrm{SDR} = \frac{m}{|T|} \times \beta(j) \times \left[sd\,(\mathrm{T}) - \sum_{i \in \{L,R\}} \frac{|\mathrm{T}_i|}{|\mathrm{T}|} \times sd\,(\mathrm{T}_i) \right], \tag{3.8}$$

where m is the number of examples without missing values for that attribute, T is the set of examples that reach this node, $\beta(j)$ is the correction factor calculated for the original attribute to which this synthetic attribute corresponds using Eq. (3.9). T_L, T_R are sets that result from splitting on this attribute

$$\beta = e^{7 \times \frac{2-k}{n}}, \tag{3.9}$$

where k is the number of samples of values of the original enumerated attribute and n is the total number of examples.

The algorithm for M5′ is outlined in Appendix C.

3.2.3 Pros and Cons of Continuous Class Learning

The advantages of continuous class learning are:

- Interpretation of the models developed using this procedure is easy;

- It requires little data preparation such as data normalization;
- The models developed are robust;
- No prior assumptions of underlying model parameters are required.

The disadvantages of continuous class learning are:

- Use of heuristic search algorithms such as greedy algorithms do not guarantee the optimal results;
- Overfitting is a common problem.

3.2.4 Tools Used

M5primelab is a MATLAB toolbox for building regression trees and model trees using M5$'$ method developed by Gints Jekabsons [27]. The functions in this toolbox are edited appropriately and used for developing the model trees in the present work. The *M5primelab* distribution comes with a basic user manual [27].

3.2.5 Results

Figures 3.3 and 3.4 show the predicted values and actual values of the soil moisture content for test data and training data, respectively. The Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) for this model are 0.0411 and 0.0536, respectively.

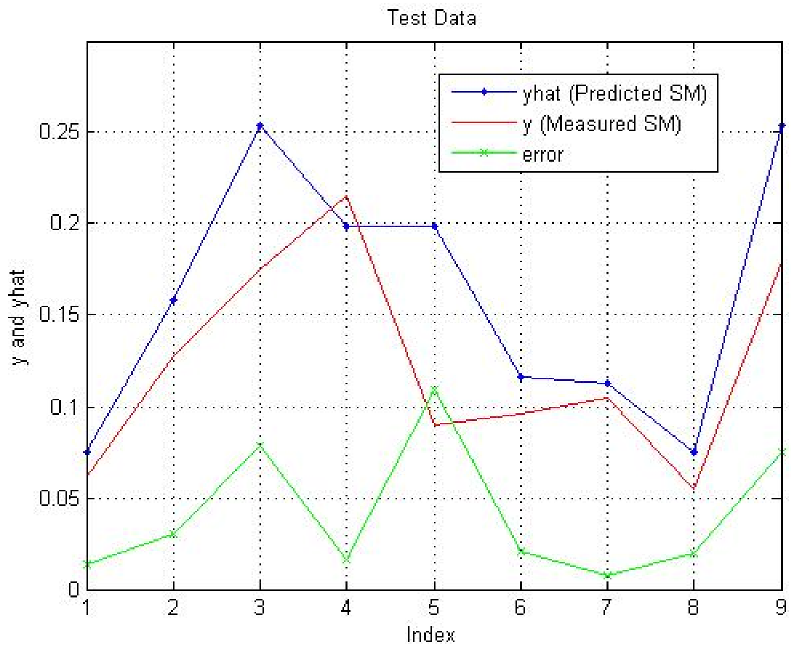

Fig. 3.3: Predicted and actual target values and absolute error for test data using model trees.

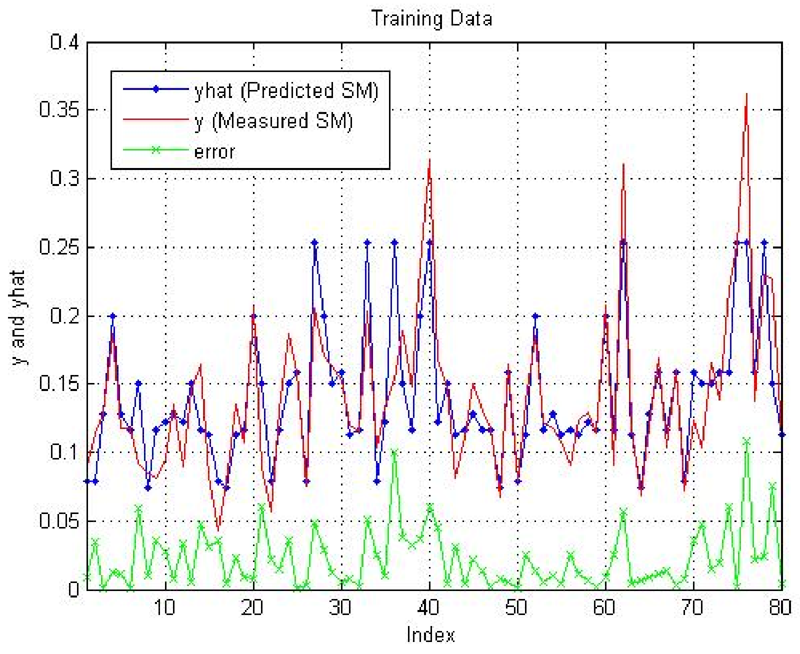

Fig. 3.4: Predicted and actual target values and absolute error for training data using model trees.

Chapter 4

Polynomial Regression and Neural Networks

This chapter explains other miscellaneous methods such as polynomial regression and neural networks used in the present work for the development of prediction models for the estimation of soil moisture content. The mathematical backgrounds of full polynomial, sparse polynomial based regression and Adaptive Basis Function Construction (ABFC) are explained in the first section. The theoretical background of constructing multilayer perceptrons, a kind of neural networks is explained in second section.

4.1 Polynomial Regression

Polynomial regression fits a nonlinear model to the data, but then as a statistical problem it is linear, as in the regression function is linear in unknown parameters to be estimated for the model. Also, polynomial regression usually fits the data to the model using least squared fit. The least squares method minimizes the variance of the unbiased estimators of the parameters of the model under the conditions of the Gauss-Markov theorem.

In these methods, basis functions are used to approximate a function. Usually, the basis functions used have fixed degrees in nonadaptive methods and otherwise for adaptive methods. In nonadaptive methods, the model approximates the unknown function as a linear combination of basis functions from a dictionary of valid basis functions. In adaptive methods, the basis functions are adapted to the data.

4.1.1 Full Polynomials

The models using low order full polynomials for approximating an unknown function are the mostly used and the easiest to develop compared to other kinds of regressions. For example, a second degree polynomial can defined as

$$f(\mathbf{x}) = a_0 + \sum_i a_i \mathbf{x}_i + \sum_i \sum_j a_{ij} \mathbf{x}_i \mathbf{x}_j. \tag{4.1}$$

A polynomial regression model can be represented in Eq. (4.2), where a_i's are model parameters are estimated usually using least squares method as shown in Eq. (4.3)

$$F(\mathbf{x}) = \sum_{i=1}^{k} a_i f_i(\mathbf{x}), \tag{4.2}$$

where k is the number of basis functions used in the model, and $f_i(\mathbf{x})$ are the basis functions of input $\mathbf{x}$.

$$\mathbf{a} = arg \min_{\mathbf{a}} \sum_{j=1}^{n} \left(y_j - F(\mathbf{x}_j)\right)^2, \tag{4.3}$$

where $\mathbf{a} = (a_1, \dots, a_k)^T$ are the model's parameters.

For a d-dimensional input and a polynomial of fixed degree p, the total number of basis functions in the size of dictionary of full polynomial is given by

$$m = \prod_{i=1}^{p} (1 + d/i). \tag{4.4}$$

The advantages of using full low-order polynomials is the requirement of low number of data points and computational efficiency. One of the important limitations of this rigorous procedure is overfitting of data to the model. The drawback of using low-order polynomials is the inability of them to approximate highly nonlinear behaviors. This is overcome by using high-order polynomials. However, use of higher-order polynomials increases the size of dictionary, and in turn the computational efficiency.

4.1.2 Sparse Polynomials

The importance of "sparse"polynomials lies in avoiding the overfitting of the data to the model and reducing the resulting errors in the regions that contain sparse data. The sparse polynomials are selected using subset selection methods whose goal is to find a subset of functions from a fixed predetermined dictionary basis functions that provide the best predictive performance of the model. So all the following benefits of feature selection apply to the sparse polynomial models [28]:

- Alleviating the effect of the “curse of dimensionality,”
- Enhancing generalization,
- Speeding up learning process,
- Improving model interpretability.

If the size of dictionary is m, the total number of possible subsets become 2^m. For reasonably big values of m, the search procedure for finding out the right subset by searching through all subsets becomes a cumbersome procedure. To make this search more practical, heuristic search algorithms can be used to find out the best subset. Some of the search algorithms that can be employed are Sequential Forward Selection (SFS), Steepest Descent Hill Climbing (SDHC), and Sequential Floating Forward Selection (SFFS).

All the search algorithms proceed iteratively adding functions to the model to get the highest performance increase according to a chosen criteria. The criteria can be chosen from any of the following: Small Sample Corrected Akaike’s Information Criterion (AICC), Bayesian Information Criterion (BIC), and Generalized Cross Validation (GCV). The expressions for these criteria in terms Mean Square Error (MSE), size of the set of training examples (N), and number of basis function used (k) are tabulated in Table 4.1.

Sequential Forward Selection and Sequential Floating Forward Selection

These sequential algorithms add or remove features sequentially, but have a tendency to become trapped in local minima. Both of these algorithms come under the scheme called maximum relevance selection which selects features that correlate strongest to the classification variable. SFS performs best when the optimal subset has a small number of features. When the subset is near the empty set, a large number of states can be potentially evaluated while when the subset is near the full set, the region examined by SFS is narrower

Table 4.1: Search criteria.

#	Name of criteria	Expression
1	AICC	$N \log MSE + 2k + \frac{2k(k+1)}{(N-k+1)}$
2	BIC	$\log MSE + k \log N$
3	GCV	$N \log MSE - 2N \log (1 - \frac{k}{N})$

since most of the features are already been selected. To state this more clearly, we can say that the search space is drawn like an ellipse to emphasize the fact that there are fewer states towards the full and empty sets.

SFS methods are an extension to Plus-L Minus-R Selection (LRS) algorithms with flexible backtracking capabilities. Rather than fixing the values of number of features to added and removed at every step, i.e. L and R values, these floating methods allow those values to be determined from the data. Sequential floating forward selection starts from an empty set and after each forward step, it performs backward steps as long as the objective function increases. In iteration of search, forward phase is done only once while the number of times the backward phase is carried is determined dynamically.

Steepest Descent Hill Climbing

Hill climbing is a mathematical optimization technique that belongs to the family of local search. It proceeds by randomly selecting a solution from the search space and iteratively makes small changes to attain the neighborhood solution. Steepest hill climbing differs from the simple hill climbing in the way the node is selected. In the former, all successors are compared and the closest to the solution is chosen, whereas in the latter the first closer node is chosen. Note that the solution chosen using this technique is close to the optimal, but it is not guaranteed ever.

4.1.3 Adaptive Basis Function Construction

The backdrop of polynomial regression lies in subset selection, where the search for the best subset is carried in exponential orders as proportional to the number of independent input parameters. To alleviate this extraordinary searches, basis function are constructed adaptively from the data. The required basis functions are automatically iteratively constructed using heuristic searches instead choosing a subset from a restricted finite user defined dictionary. So note that in this approach, the basis function dictionary is infinite and polynomials of arbitrary complexity are generated brining the desired flexibility to the model building process. In one way, we can say that the ultimate aim of Adaptive Basis Function Construction is to overcome some of the limitations associated with subset selection.

Generally, a basis function in a polynomial regression model can be defined as a product of original input variables each with an individual exponent as

$$f_i(\mathbf{x}) = \prod_j \mathbf{x}_j^{r_{ij}}, \tag{4.5}$$

where $\mathbf{r}$ is a $k \times d$ matrix of nonnegative integer exponents such that r_{ij} is the exponent of the j^{th} variable in the i^{th} basis function. Finding a best subset can now be defined as finding the best matrix $\mathbf{r}$ with the best combination of nonnegative integer values of its elements, that is

$$\mathbf{r} = arg \min_{\mathbf{r}} \left\{ J\left(\prod_j \mathbf{x}_j^{r_{ij}} \right) \right\} \quad i = 1, 2, \ldots, k, \tag{4.6}$$

where $J(.)$ is an evaluation criterion that evaluates the predictive performance of the regression model which corresponds to the set of basis functions. Following are the five components of heuristic search procedure explained with respect to the ABFC approach [29].

Initial State Generally, the simplest model is chosen to be the initial state. Here, the model with a single basis function is the simplest one and it corresponds to the intercept term with all the exponent terms for the basis function being 0s.

State Transition Operators Subset selection approach employs only add/delete operators for state transition while in ABFC approach, modified form of the usual add/ delete operators are used which facilitate the modification on the level of individual exponents. The basic idea behind using such modified operators is to employ operators that add only simplest functions. The following are the four operators used:

- Add a new linear basis function with only one of the exponents as one and all others zeros;
- Add an exact copy of already existing basis function with one of exponents increased by 1;
- Decrease one of the exponents in one of the existing basis functions by 1;
- Delete one of the existing basis functions.

Search Strategy Since both complication and simplification algorithms are used, we can employ both forward and backward type searches.

Evaluation Measure The evaluation measures used for ABFC approach are the same as in subset selection approaches.

Termination Condition Some of the widely used termination conditions are:

- User defined pre-specified number of iterations;
- User pre-specified size of model;
- No further improvement of the model can be made with the available state transition operators.

4.1.4 Floating Adaptive Basis Function Construction

This is a special case of ABFC approach [30]. It uses SFFS search procedure starting from a simplest model. SFFS is one among the most efficient heuristic floating search algorithms. In ABFC, sometimes when the data is of low dimensionality, the search algorithm may get stuck in a local minimum too early in the search, returning a too simple and underfitted model. F-ABFC approach addresses this issue by employing an additional recursion of the state-transition operators which introduces another hyper parameter for the model.

The recursion of operators reduces the number of local minima in the state space, which helps in the searching procedure and ultimately results in a better model.

4.1.5 Pros and Cons of Adaptive Construction of Basis Functions

The advantages of ABFC modeling are:

- The computational considerations for ABFC search procedure are better than those of subset selection;
- Use of four operators instead of the conventional add/delete operators helps in making the building procedure easier and more efficient;
- The models of best predictive performance are usually located relatively near to the initial state. Hence, only a small fraction of the whole infinite state space can be explored.

The disadvantages of ABFC modeling are that they can become computationally demanding when the number of input variables or the size of training set get very large. Selection bias occurs when the same dataset is used for all the tasks such as model building, selection of best model and steering the direction of search procedure and this selection bias increases with the noise present in the dataset. Selection instability is another issue, which is related to the fact that small perturbations in the data can make the model building process result in vastly different models.

4.1.6 Tools Used

FABFC is a MATLAB toolbox implementing adaptive modeling techniques through Floating-Adaptive Basis Function Construction, developed by Gints Jekabsons [31]. The functions in this toolbox are edited appropriately and used for developing the adaptive models discussed in this section.

4.1.7 Results

Figures 4.1 and 4.2 show the predicted values and actual values of the soil moisture content for test data and training data, respectively. The Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) for this model are 0.0389 and 0.0453, respectively.

4.2 Neural Networks

A neural network is a mathematical model consisting of interconnected neurons (nodes) that dupe the functioning of biological neurons. They are used as nonlinear statistical tools in modeling complex relationships between inputs and outputs. The neurons are arranged in different layers with random weighted connections between layers. It is an adaptive system that changes it structure based on the external or internal information that flows through the network during the training phase. A typical neural network with four input nodes, three output nodes, and one hidden layer with six nodes is shown in Fig. 4.3.

The functioning is basically similar to flow diagrams in networks, like every node sums up the weighted inputs and transfers it to all the other nodes connected to it with the respective weights multiplied (feed-forward). The development of the model involves two phases: training and testing. Training phase essentially selects one model from all the available ones that optimizes the cost function evaluated on the obtained training dataset. In other words, the objective of training phase is to determine a mapping from the set of training dataset to the set of possible weights. Test phase involves the assessment of the performance of the model using different evaluation procedures or a test dataset, if available. The cost function is usually the mean-squared error that represents the mismatch between the mapped functional output and the dataset.

Different training procedures have evolved to be used during the first phase of neural network development. The kind of feed-forward neural network that uses back propagation algorithm for training is called Multilayer Perceptron network (MLP). These are the most commonly used neural network models. The MLP networks have only one hidden layer as shown in Fig. 4.4, and the activation function at the nodes are usually hyperbolic tangent

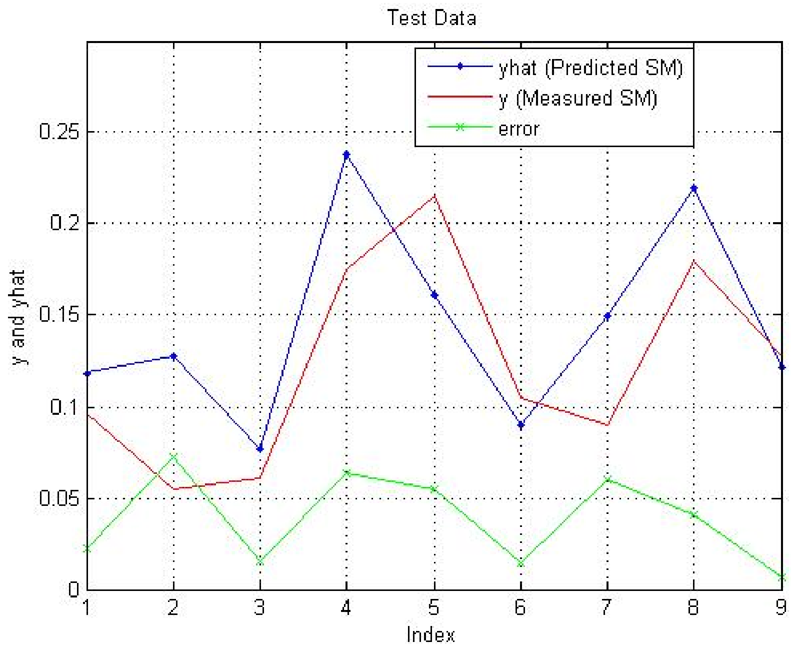

Fig. 4.1: Predicted and actual target values and absolute error for test data using ABFC.

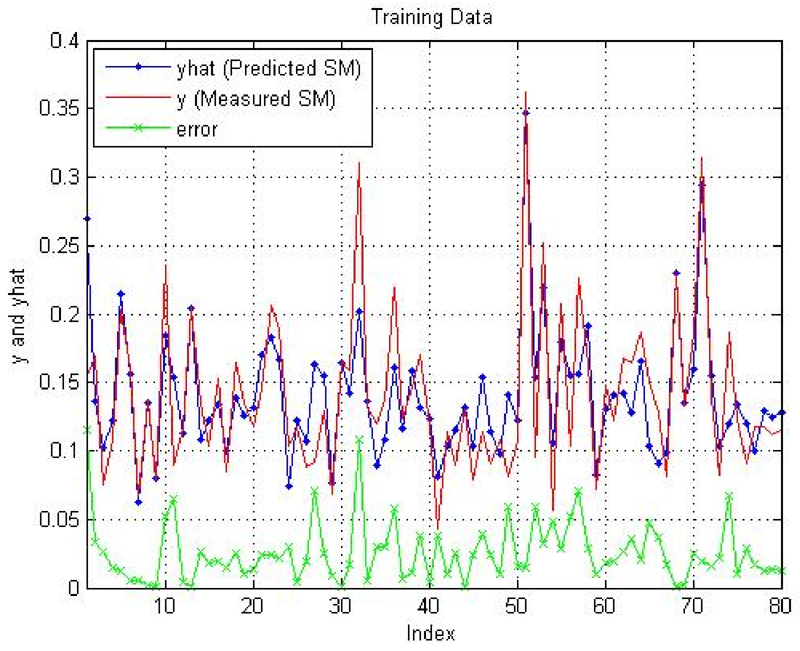

Fig. 4.2: Predicted and actual target values and absolute error for training data using ABFC.

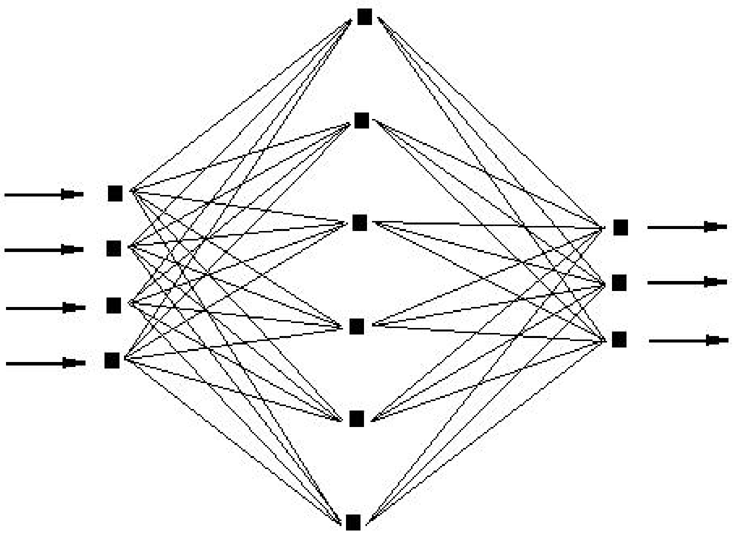

Fig. 4.3: A typical neural network.

or linear activation functions (f, F).

$$\hat{y}(w, W) = F_i \left(\sum_{j=1}^{q} W_{ij} h_j(w) + W_{i0} \right) \quad = F_i \left(\sum_{j=0}^{q} (W_{ij} f_j \left(\sum_{l=1}^{m} w_{jl} z_l + w_{j0} \right) + W_{i0}) \right) \tag{4.7}$$

4.2.1 Back-Propagation Algorithm

This algorithm minimizes the cost function to find the weights $(\theta = [w, W])$ using gradient descent. Let T denote the training dataset $\{\mathbf{x}_i, y_i\}_{i=1}^{N}$. The cost function in the form of mean square error criterion is

$$V_N(\theta, T) = \frac{1}{2N} \sum_{t=1}^{N} \left[y_t - \hat{y}_{t/\theta}\right]^T \left[y_t - \hat{y}_{t/\theta}\right]. \tag{4.8}$$

The weights can be found as

$$\hat{\theta} = arg \min_{\theta} V_N(\theta, T). \tag{4.9}$$

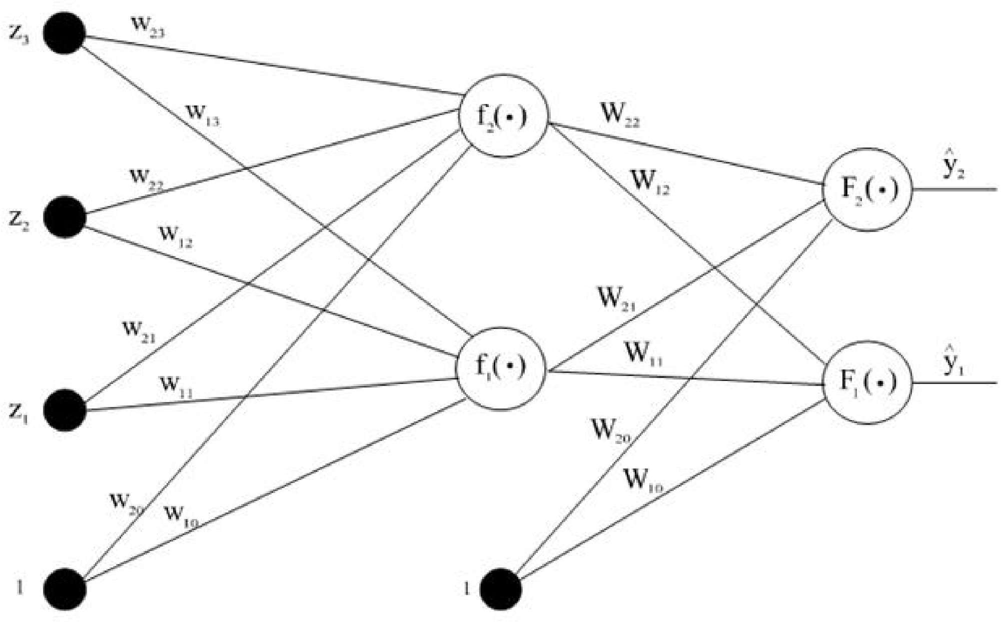

Fig. 4.4: A feed-forward MLP network with weights and activation functions.

The cost function in the form of Minimum Mean Square Error (MMSE) is minimized in the algorithm by propagating the error backwards through the network and adjusting the weights accordingly. The weights are thus iteratively updated as

$$\hat{\theta}^{(i+1)} = \hat{\theta}^{(i)} + \mu^{(i)} f^{(i)}, \tag{4.10}$$

where $\theta^{(i)}$ represents the present iterate, $f^{(i)}$ gives the search direction, and $\mu^{(i)}$ is the step size.

All the computations involved are ordered in a way to take advantage of the structure of the neural network. The algorithm is implemented in the following steps [32].

Feed-Forward Calculation The training input is propagated through the network and the corresponding activations at all the nodes are generated in the process.

Back-Propagation to the Output Layer The gradients of all outputs and hidden neurons are generated through the back propagation of the activations of all the neurons.

Back-Propagation to the Hidden Layer The back-propagation error in each hidden layer is computed taking into account all possible backward paths.

Weight Updates The weights are updated by subtracting a fraction of them from the previous weights.

4.2.2 Levenberg-Marquardt (LM) Method

This method of optimizing the cost function is more rapid and robust [33]. LM method is a compromise between the Gauss-Newton Algorithm and the method of Gradient Descent employed in back-propagation algorithm. The algorithm is implemented in the following steps [33].

Step 1: Select the initial parameter vector $\theta^{(0)}$ and an initial value $\lambda(0)$.

Step 2: Determine the search direction from $\left[R\left(\theta^{(i)}\right)+\lambda^{(i)}I\right]f^{(i)}=-G\left(\theta^{(i)}\right)$, I being a unit matrix. Here, G denotes the gradient of the criterion with respect to the weights and R is the Gauss-Newton approximation to the Hessian.

Step 3: Compute

$$r^{(i)}=\frac{V_N(\theta,T)-V_N(\theta^{(i)}+f^{(i)},T)}{V_N(\theta,T)-L^{(i)}(\theta^{(i)}+f^{(i)})}, \tag{4.11}$$

$$\text{where}\quad L^{(i)}(\theta^{(i)}+f)=\sum_{t=1}^{N}\left(y_t-y_{t/\theta}-f^T\left[\frac{\partial\hat{y}_{t/\theta}}{\partial\theta}\|_{\theta=\hat{\theta}}\right]\right)^2$$
$$=V_N(\theta^{(i)},T)+f^TG(\theta^{(i)})+\frac{1}{2}f^TR(\theta^{(i)})f.$$

If predicted decrease is close to actual decrease, let the search direction approach the Gauss-Newton search direction while increasing the step size. That is if $r(i)>0.75$ implies that $\lambda^{(i)}=\lambda^{(i)}/2$.

Step 4: If predicted decrease is far from the actual decease let the search direction approach the gradient direction while decreasing the step size. That is if $r(i) < 0.25$ implies that $\lambda^{(i)} = 2\lambda^{(i)}$.

Step 5: If $V_N(\theta^{(i)} + f^{(i)}, T) < V_N(\theta^{(i)}, T)$, then accept $\theta^{(i+1)} = \theta^{(i)} + f^{(i)}$ as new iterate and let $\lambda^{(i+1)} = \lambda^{(i)}$ and $i = i + 1$.

Step 6: If the stopping criterion is not satisfied, go to step 2.

4.2.3 Pros and Cons of Neural Networks

The advantages of neural networks are:

- No restrictive assumptions,
- Can overcome autocorrelation between input features,
- Robust and flexible.

The disadvantages of neural networks are:

- Risk of overfitting,
- Requires definition of architecture,
- Long processing time,
- Large training samples required.

4.2.4 Tools Used

Neural Network Based System Identification Toolbox is developed by Magus Norgaard, Department of Automations, Technical University of Denmark. The toolbox contains a large number of functions for training and evaluation of MLP networks. The toolbox provides six different model structures. Two of them, *NNARX* and *NNOE*, have been used in the present work. The former is a simple auto-regressive type neural network model while the latter is a Neural Network Output Error model. Both of them use Levenberg Marquardt method for training the parameters. Mostly, a test dataset is used to cross-validate the trained model. The toolbox also provides functions like *NNVALID* and *NNEVAL* to validate the models. This distribution comes with a basis user manual [33].

4.2.5 Results

Figures 4.5 and 4.6 show the predicted values and actual values of the soil moisture content for test data and training data, respectively. The Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) for this model are 0.0677 and 0.0856, respectively.

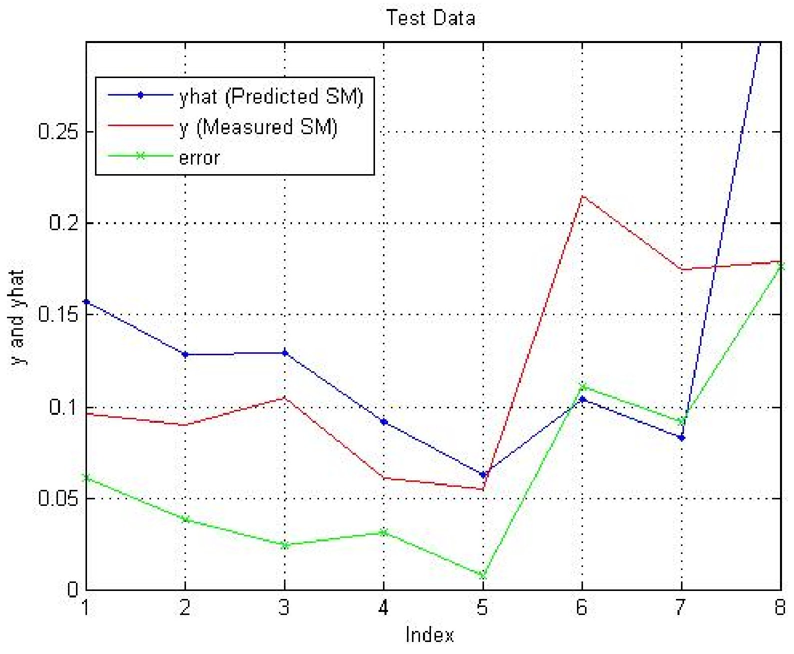

Fig. 4.5: Predicted and actual target values and absolute error for test data using neural networks.

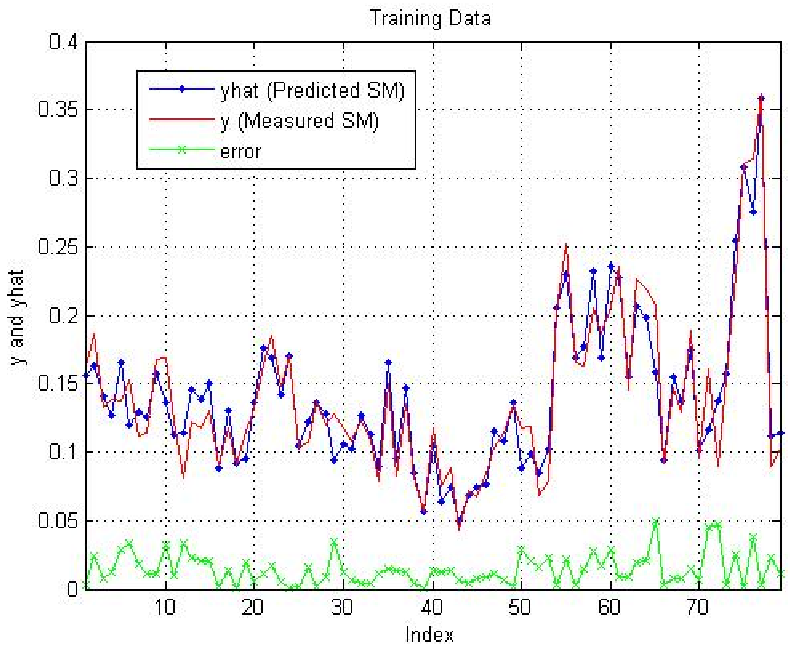

Fig. 4.6: Predicted and actual target values and absolute error for training data using neural networks.

Chapter 5
Ensembling Methods

This chapter explains some ensembling techniques such bagging, variance optimized bagging, and boosting for combining models built on same learning techniques. Every model suffers from selection bias and selection instability when the same dataset is used for different tasks like model building, determining the search criteria, and selection of final best model. Ensembling models may avoid such kind of problems. These techniques also improve prediction accuracies.

The first section goes through an ensembling technique applied to adaptively constructed basis functions which is similar to the bagging procedure, but differs in the way the best model is chosen. The latter sections give the details about the other ensembling techniques and present the results with different base learning algorithms such as Relevance Vector Machines and Adaptively Constructed Basis Functions.

5.1 Ensembling Floating Adaptive Basis Function Construction (EF-ABFC)

In order to deal with the selection bias and selection instability of the models developed by adaptively constructed basis functions, ν-fold Cross-Validation (CV) over the entire search process is used [34]. All other features such as the evaluation criteria are followed the same way as in the ABFC model building process. The training data is re sampled with a CV-type re sampling. During this resampling, the whole training dataset is randomly divided into ν disjoint subsets. Then ν overlapping training datasets are constructed by dropping out a different one of these ν subsets during each iteration. The post-evaluation is done using the validation dataset left out during each iteration, to select the best models of each iteration. This process is repeated ν times each time with different CV fold serving as the the validation dataset and all other as training datasets.

By combining the ν models using a simple unweighted model averaging as indicated in Eq. (5.1), the issue of selection instability is resolved. Note that prior to combining, all the models need to be refitted to the whole training set in order to compensate for the smaller training set used during individual model building. Such a combining of models can smooth out the effect of erratic models that overfit the data thus attaining more model stability.

$$\hat{y}_{comb} = (1/\nu)\sum_{i=1}^{\nu} \hat{y}_i \tag{5.1}$$

The disadvantages of this kind of modeling are:

- When large datasets are available, the positive effects of EF-ABFC might diminish;
- Increased complexity of the algorithm may lead to inaccurate and overfitted models.

Results

Figures 5.1 and 5.2 show the predicted values and actual values of the soil moisture content for test data and training data, respectively. The Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) for this model are 0.0376 and 0.0431, respectively.

5.2 Bagging Predictors

Bagging is a technique developed by Breiman [35]. Both stability and prediction accuracy can be improved using this technique. It can be applied over classification as well as regression problems. The multiple versions of models are formed by bootstrapping the dataset and using them as training sets for building different models. In the case of a classification problem, a plurality vote is taken to predict a nominal target variable while in the case of a regression problem, average of all the versions is taken as a prediction for a numerical target variable. The name bagging came from bootstrap aggregating. The more general form of bagging can be seen in the Perturb and Combine methods developed by Breiman, in which regression technique is applied to several perturbations of the original data and the results are combined to obtain a single regression model.

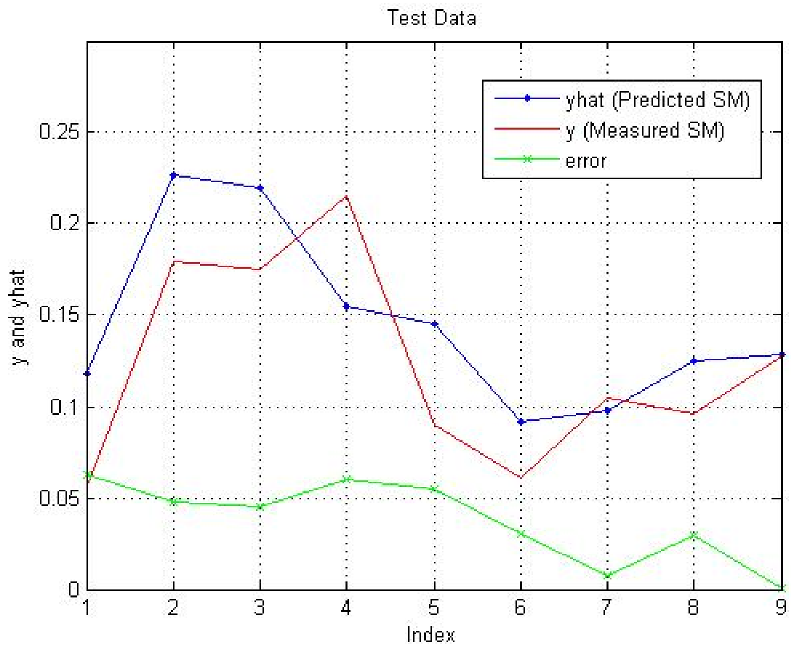

Fig. 5.1: Predicted and actual target values and absolute error for test data using EF-ABFC.

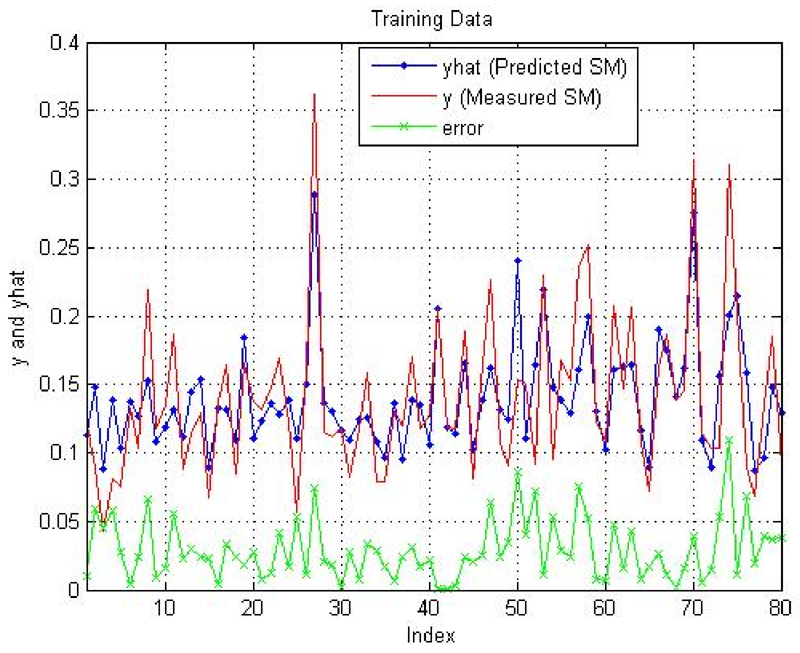

Fig. 5.2: Predicted and actual target values and absolute error for training data using EF-ABFC.

Let $\mathcal{L}$ denote the actual learning set and $\mathcal{L}_{\mathcal{B}}$ denote bootstrapped samples from $\mathcal{L}$ and $\psi(\mathbf{x}, \mathcal{L})$ denote the model developed using the input variables $\mathbf{x}$, learning set $\mathcal{L}$ and learning algorithm ψ. The algorithm follows as below.

Step 1: The whole dataset is divided into test set $\mathcal{T}$ and learning set $\mathcal{L}$, the share of the former being usually 10% of the whole dataset.

Step 2: A bootstrap sample $\mathcal{L}_{\mathcal{B}}$ is selected from $\mathcal{L}$ and a prediction model is built using this bootstrap sample as the training dataset following a specific learning algorithm such as ABFC, RVM, or SVM. This is repeated T times so that T predictors $\psi_1(\mathbf{x}), \psi_2(\mathbf{x}), \ldots, \psi_T(\mathbf{x})$.

Step 3: Now for all $(\mathbf{x}_n, y_n) \in \mathcal{T}$, the bagged predictor is $\hat{y_n} = \text{avg}_k \psi_k(\mathbf{x}_n)$ and the squared error $e_B(\mathcal{L}, \mathcal{T})$ is $\text{avg}_n(y_n - \hat{y}_n)^2$.

Step 4: The random division of the data into $\mathcal{L}$ and $\mathcal{T}$ is repeated n times and the errors are averaged to give $\bar{e}_B$.

It is understood that the aggregated predictor is actually the average over $\mathcal{L}$ of $\psi(\mathbf{x}, \mathcal{L})$ that is $E_{\mathcal{L}\psi(\mathbf{x},\mathcal{L})}$. It can be proven easily that

$$\left[E_{\mathcal{L}\psi(\mathbf{x},\mathcal{L})}\right]^2 \leq E_{\mathcal{L}\psi^2(\mathbf{x},\mathcal{L})}. \tag{5.2}$$

It can be thus understood that the difference of the mean-squared prediction error for a prediction with the square of the mean value of the prediction is equal to the variance of the predictor. The larger this variance is relative to the mean squared prediction error of the predictor, the greater the percentage reduction of the mean squared prediction error will be if the predictor is replaced by its mean [36]. Hence, the predictors with relatively large variance can be improved by bagging. Hence, we can conclude that bagging shows reasonable improvement when the base learner is rather unstable. It can degrade the performance otherwise.

The number of bootstrap replicates, T required is not an issue. Usually, a 10 to 25 number of replicates are enough as the work of Breiman proves. Also, usually it is helpful if we make the size of a bootstrap sample $\mathcal{L}_{\mathcal{B}}$ equal to the size of the actually learning set, $\mathcal{L}$.

Results

Figures 5.3 and 5.4 show the predicted values and actual values of the soil moisture content for test data and training data, respectively, obtained by bagging RVMs. The Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) for this model are 0.0935 and 0.1034, respectively.

5.3 Variance Optimized Bagging

This section explains a technique developed by Derbeko et al. for aggregating an ensemble of bootstrap predictors [37]. Weights are chosen for forming the linear combination of predictors through minimum variance computations using quadratic programming. The idea is actually borrowed from Markowitz Mean-Variance Portfolio Optimization. It has been observed that this technique can even produce improvements in ensembling those models that are deteriorated by bagging. The procedure thus developed was termed as Variance Optimized Bagging, in short as Vogging.

In this technique, the goal is to find optimized weights for obtaining a weighted average of the predictors developed as in bagging. Similar to bagging, T bootstrap samples are produced from the whole learning set $\mathcal{L}$ available and T predictors are developed using each of those bootstrap samples as training sets for each model. Let the predictors be denoted as $\psi_i \quad i = 1, 2 \ldots, T$, and let the bootstrap samples be denoted as $B_i \quad i = 1, 2, \ldots, T$, each of size n_B. Let $A_j(B_i)$ denote the empirical accuracy obtained by the predictor ψ_j on the bootstrap sample B_i given by Eq. (5.3). Let $\bar{A}_j = \frac{1}{T-1} \sum_{i \neq j} A_j(B_i)$ be the average empirical accuracy over all the other bootstrap samples.

$$\psi_j \text{ on } B_i = \frac{1}{n_B} \sum_{y \in B_i} (y - \hat{y}) \tag{5.3}$$

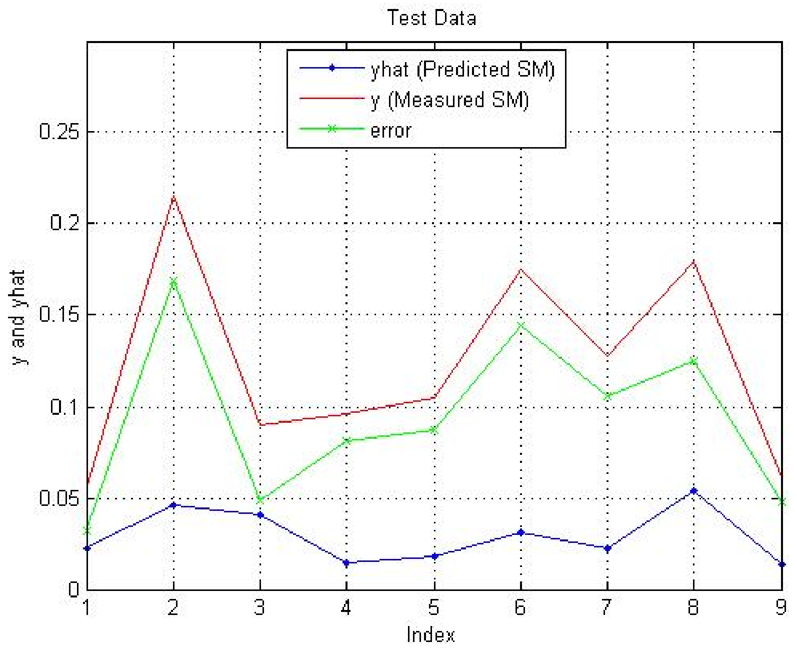

Fig. 5.3: Predicted and actual target values and absolute error for test data using bagging.

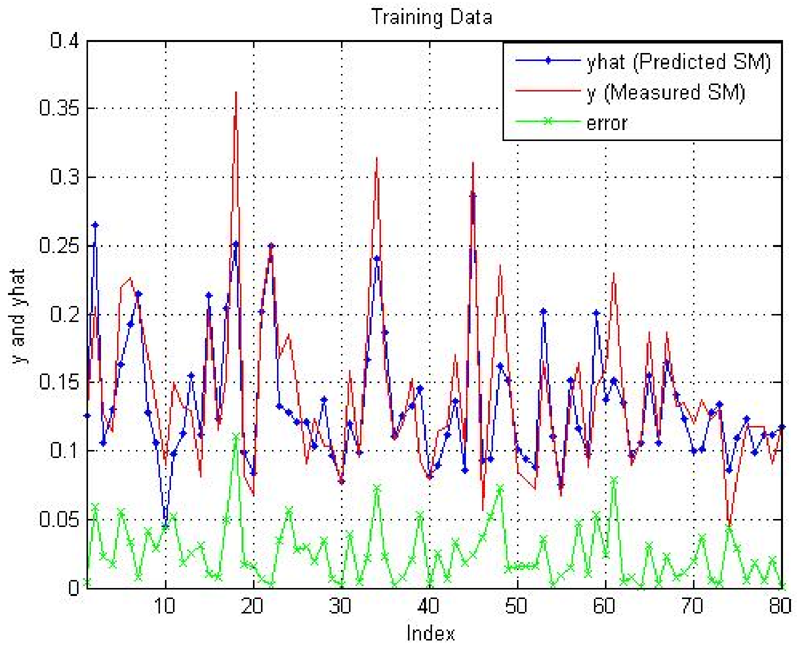

Fig. 5.4: Predicted and actual target values and absolute error for training data using bagging.

The empirical covariance, Q is given by Eq. (5.4).

$$Q = \frac{1}{T-1}\sum_{j=1}^{T}(A_j - \bar{A})(A_j - \bar{A})^T, \tag{5.4}$$

$$\text{where} \quad A_j = [A_j(B_1), \ldots, A_j(B_T)] \quad j = 1, \ldots, T \text{ and } \quad \bar{A} = \frac{1}{T}\sum_{j=1}^{T} A_j.$$

Now, for each $a_i \in U\left[\min_j \bar{A}_j, \max_j \bar{A}_j\right] \quad i = 1, 2, \ldots, k$, the following quadratic problem is solved with linear constraints.

$$\text{Minimize over } \mathbf{w} : \min_{\mathbf{w}} \frac{1}{2}\mathbf{w}^T Q\mathbf{w}; \tag{5.5}$$

$$\text{subject to} : \left(\bar{A}_1, \bar{A}_2, \ldots, \bar{A}_T\right)^T \mathbf{w} \geq a_i; \tag{5.6}$$

$$\text{and} \quad \sum_j w_j = 1, \quad \mathbf{w} \geq 0. \tag{5.7}$$

The vogging weight vector is chosen out of these k sequences of weights as $\mathbf{w}_{i^*}$ with $i^* = arg\max_i \frac{a_i' - p_0}{\sigma_i}$ where (a_i', σ_i) is the mean-variance pair corresponding to a_i and p_0 is the mean of all the responses obtained using all the predictors. The algorithm for Vogging is outlined in Appendix D.

5.3.1 Tools Used

The quadratic problem in Eq. (5.5) can be solved by convex optimization using *cvx*, a modeling system for disciplined convex programming. Disciplined convex programming is a methodology for constructing convex optimization problems proposed by Michael Grant and Stephen Boyd. *cvx* is implemented in MATLAB, effectively turning it into an optimization modeling language. This distribution comes with a basic user manual [38].

5.3.2 Results

Figures 5.5 and 5.6 show the predicted values and actual values of the soil moisture content for test data and training data, respectively, obtained by applying this technique on RVMs. The Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) for this model are 0.1115 and 0.1217, respectively.

5.4 Boosting

Boosting is another technique used generally for improving the accuracy of any given learning algorithm. Drucker developed boosting for regression purposes [39] by modifying the *Adaboost.R* algorithm previously developed by Freund and Schapire [40]. It is very much similar to bagging in its its implementation except that, in bagging each training example is equally likely to be picked while bootstrapping. On the other hand, the probability of a particular example being in the training set of a particular machine depends on the performance of the prior models on that example. It accomplishes this by maintaining a distribution of weights over the training set which are increased proportional to the difference between the actual and predicted values of the target variables. So in simple words, this algorithm tracks and makes note of all the ill-predicted values and increases their probability to be chosen into another bootstrap sample thus allowing the base learner to concentrate more on harder examples.

The algorithm for Boosting is outlined in Appendix E.

Results

Figures 5.7 and 5.8 show the predicted values and actual values of the soil moisture content for test data and training data, respectively, obtained by boosting RVMs. The Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) for this model are 0.0205 and 0.0264, respectively.

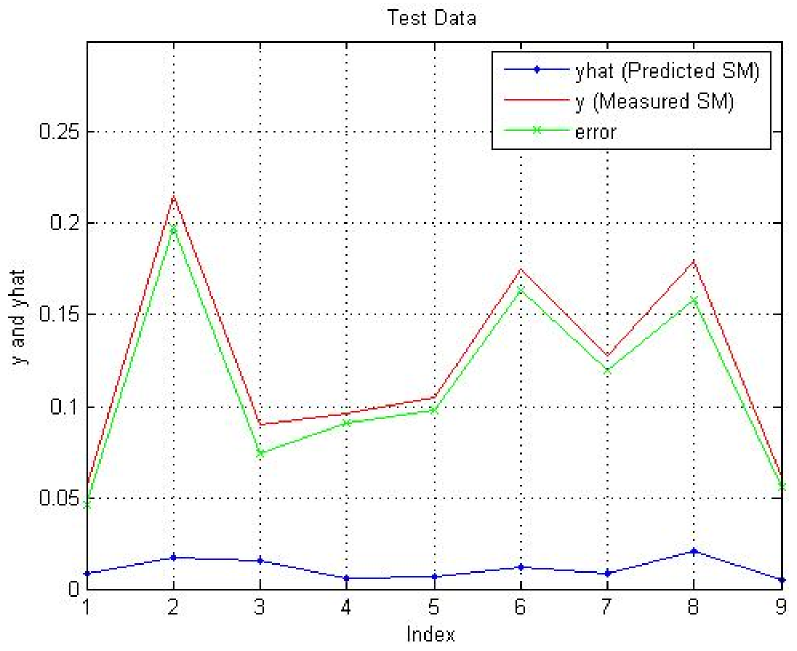

Fig. 5.5: Predicted and actual target values and absolute error for test data using vogging.

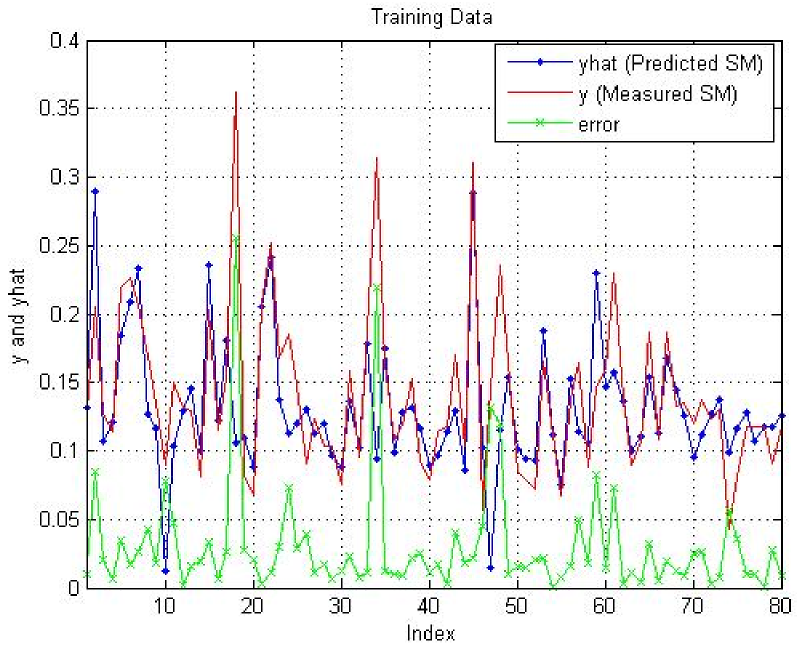

Fig. 5.6: Predicted and actual target values and absolute error for training data using vogging.

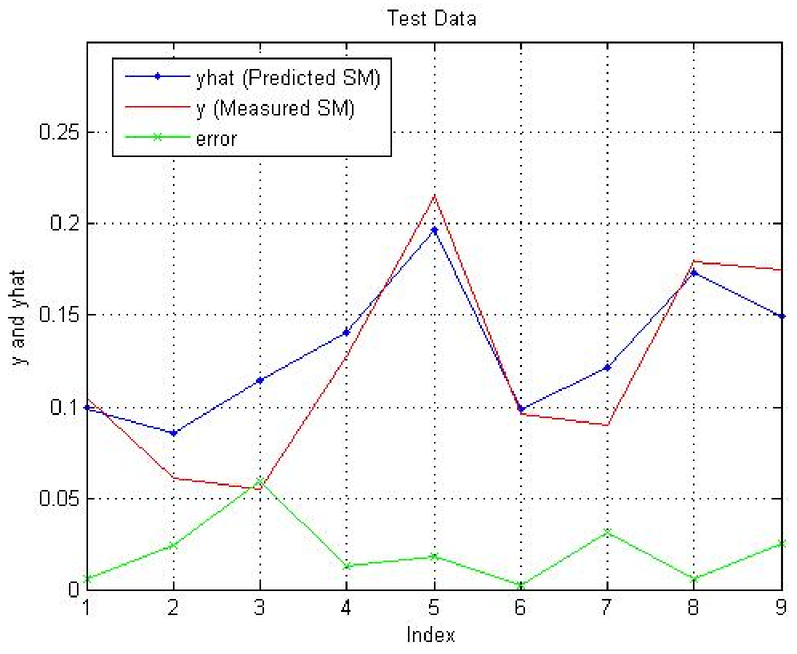

Fig. 5.7: Predicted and actual target values and absolute error for test data using boosting.

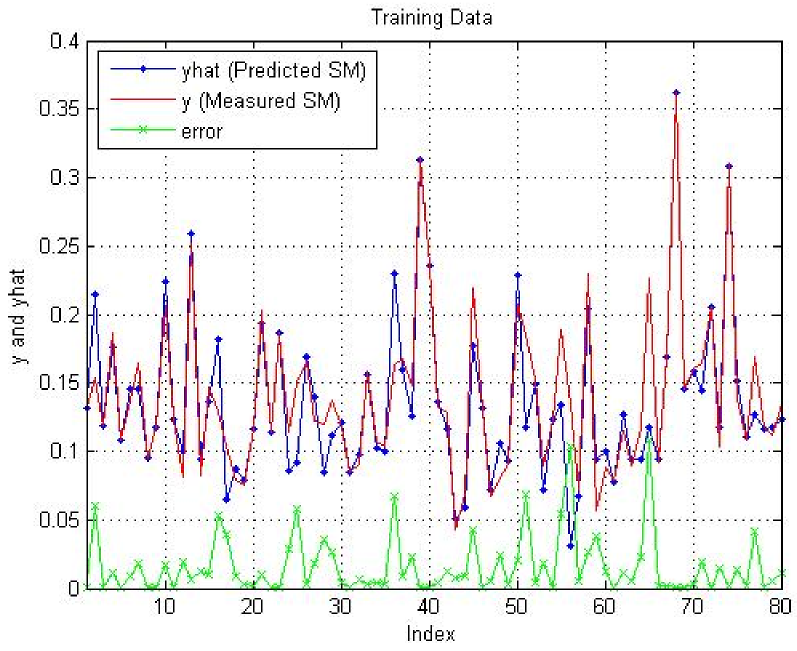

Fig. 5.8: Predicted and actual target values and absolute error for training data using boosting.

Chapter 6

Conclusion

6.1 Summary of Results

One of the main goals of developing all these models is to produce reliably good estimates of Soil Moisture Content using the most easily available attributes such as the meteorological factors and vegetation indices. This information is very helpful to farmers and water researchers to understand the approximate conditions of the fields and guides them to decide more efficiently about farming practices such as irrigation and application of fertilizers.

The present work has also given a chance to explore different kinds of empirical modeling techniques and understand the mathematical backgrounds on which they are built. This work provides a survey of all important and mostly used nonparametric methods ranging from the basic polynomial regression to the modern tools like neural networks. The results obtained are quite satisfactory and encouraging

The Mean Square Error (MSE) and Root Mean Square Error (RMSE) for test data for all the models developed in the present work are summarized in Table 6.1.

Of all the models used in the present work, RVM proved to be the best one with reasonable training and test errors. The training error for neural networks were better than any other model, but their test errors were not at all satisfactory. Other models that gave reasonable results of all are ABFC and MARS.

The ensembling techniques like Bagging and Boosting proved to produce better predictors for the models of RVM and ABFC. Sometimes, it has been observed that Vogging deteriorated the results contrary to what it is supposed to be producing.

Table 6.1: Summary of results.

#	Model	MAE	RMSE
1	Support Vector Machines	0.0499	0.0546
2	Relevance Vector Machines	0.0279	0.0323
3	Multivariate Adaptive Regression Splines	0.0279	0.0484
4	Model trees using M5′ algorithm	0.0411	0.0536
5	Adaptive Basis Function Construction	0.0389	0.0453
6	Multilayer Perceptron Networks	0.0677	0.0856
7	Ensembling Adaptive basis Function Construction	0.0376	0.0431
8	Bagging RVMs	0.0935	0.1034
9	Variance Optimized Bagging RVMs	0.1115	0.1217
10	Boosting RVMs	0.0205	0.0264

6.2 Future Work

Further work in this direction might involve using the data derived from Unmanned Air Vehicles unlike the satellite data used in this present work. Many other input parameters which are nominal may also be allowed while building the model. Also, spatio-temporal maps of the retrieved soil moisture content measurements might be developed using more sophisticated models.

References

[1] "Worldometers about world statistics updates in real time 2011. water consumed this year," http://www.worldometers.info.water/.

[2] "Estimating soil moisture," http://www.ext.colostate.edu/pubs/crops/04700.html.

[3] A. L. Kaleita, L. Tian, and H. Yao, "Soil moisture estimation from remotely sensed data," in *American Society of Agricultural Engineers Annual International meeting*, Las Vegas, NV, July 27-30, 2003.

[4] J. F. Power, P. L. Brown, T. J. Army, and M. G. Klages, "Phosphorus responses by dryland spring wheat as influenced by moisture supplies," *Agronomy Journal*, vol. 53, pp. 106–108, 1961.

[5] C. M. Stewart, A. B. McBratney, and J. H. Skerritt, "Site specific durum wheat quality and its relationship to soil properties in a single field in northern south wales," *Precision Agriculture*, vol. 3, pp. 155–168, 2002.

[6] S. Machado, E. D. Bynum, T. L. Archer, R. J. Lascano, L. T. Wilson, L. Borodvsky, E. Scarra, D. M. Nesmith, and W. Xu, "Spatial and temporal variability of corn grain yield: site specific relationships to biotic and abiotic factors," *Precision Agriculture*, vol. 2, pp. 359–376, 2000.

[7] T. J. Jackson, "Soil water modeling and remote sensing," *IEEE Transactions on Geosciences and Remote Sensing*, vol. 24, pp. 37–46, 1986.

[8] S. Gorthi and H. Dou, "Prediction models for estimation of soil moisture content," in *2011 ASME /IEEE International Conference on Mechatronic and Embedded Systems and Applications (MESA2011)*, Washington, DC, August 28-31, 2011.

[9] S. Ahmad, A. Kalra, and H. Stephen, "Estimating soil moisture using remote sensing data: A machine learning approach," *Advances in Water Resources*, vol. 33, pp. 69–80, 2010.

[10] "Machine learning," http://en.wikipedia.org/wiki/Machine_learning.

[11] B. Zaman, *Remotely Sensed Data Assimilation Technique to Develop Machine Learning Models for Use in Water Management.* Ph.D. dissertation, Utah State University, Logan, UT, 2010.

[12] N. Andrew, "Support vector machines lecture notes," http://cs229.stanford.edu/notes/cs229-notes3.pdf.

[13] S. R. Gunn, "Support vector machines for classification and regression," http://users.ecs.soton.ac.uk/srg/publications/pdf/SVM.pdf.

[14] M. G. Genton, “Classes of kernels for machine learning: A statistics perspective,” *Journal of Machine Learning*, vol. 2, pp. 299–312, 2001.

[15] M. E. Tipping, “Sparse bayesian learning and the relevance vector machines,” *Journal of Machine Learning*, vol. 1, pp. 211–244, 2001.

[16] M. E. Tipping, *Bayesian Inference: An Introduction to Principles and Practice in Machine Learning*, ser. Advanced Lectures on Machine Learning, O. Bousquet, U. V. Luxburg, and G. Ratsch, Eds. New York: Springer, 2004.

[17] D. Mackay, *Bayesian methods for backpropagation networks*, ser. Model of Neural Networks, E. Domany, J. L. V. Hemmen, and K. Shulten, Eds. New York: Springer, 1994.

[18] T. Fletcher, “Relevance vector machines explained,” www.cs.ucl.ac.uk/staff/T.Fletcher/.

[19] D. Mackay, “Bayesian interpolation,” *Neural Computations*, vol. 4, pp. 415–447, 1992.

[20] A. P. Dempster, N. M. Laird, and D. B. Rubin, “Maximum likelihood from incomplete data via the expectation maximization algorithm,” *Journal of Royal Statistical Society*, vol. 39, pp. 1–38, 1977.

[21] M. E. Tipping, *SparseBayes: An Efficient Matlab Implementation of the Sparse Gayesian Modelling Algorithm (Version 2.0)*, Mar. 2009.

[22] J. Friedman, “Multivariate adaptive regression splines,” *The Annals of Statistics*, vol. 19, pp. 1–141, 1991.

[23] G. Jekabsons, *ARESLab: Adaptive Regression Splines Toolbox for Matlab/Octave (Version 1.5)*, Institute of Applied Computer Systems, Sept. 2010.

[24] J. R. Quinlan, “Learning with continuous classes,” in *Proceedings of 5th Australian Joint Conference on Artificial Intelligence*, A. A and S. L, Eds., pp. 343–348. Singapore: World Scientific, 1992.

[25] J. R. Quinlan, “Combining instance-based and model-based learning,” in *Proceedings of Machine Learning ’93*, U. P, Ed., pp. 236–243. San Mateo: Morgan Kaufmann, 1993.

[26] Y. Wang and I. H. Witten, “Induction of model trees for predicting continuous classes,” http://www.cs.waikato.ac.nz/pubs/wp/1996/uow-cs-wp-1996-23.pdf.

[27] G. Jekabsons, *M5primeLab: M5’ regression tree and model tree toolbox for Matlab/Octave*, Institute of Applied Computer Systems, Sept. 2010.

[28] A. J. Miller, *Subset Selection in Regression*, V. Isham, N. Keiding, T. Louis, N. Reid, R. Tibshirani, and H. Ton, Eds. New York: Chapman & Hall/CRC, 2002.

[29] G. Jekabsons, *Machine Learning*, ch. Adaptive Basis Function Construction: An Approach for Adaptive Building of Sparse Polynomial Regression Models, pp. 127–156. Latvia: InTech, 2010.

[30] G. Jekabsons and J. Lavendels, "Polynomial regression modelling using adaptive construction of basis functions," in *Proceedings of IADIS International Conference, Applied Computing*, pp. 269–276, 2008.

[31] G. Jekabsons, *F-ABFC and EF-ABFC : A Matlab/Octave tool for implementing Adaptive Regression Modelling*, Mar. 2010.

[32] R. Rojas, *Neural Networks*, ch. The Backpropagation Algorithm, pp. 151–183. Berlin: Springer-Verlag, 1996.

[33] M. Norgaard, *Neural Network Based System Identification Toolbox*, Department of Automation.

[34] G. Jekabsons, "Ensembling adaptively constructed polynomial regression models," *International Journal of Intelligent Systems and Technologies*, vol. 3, pp. 56–61, 2008.

[35] L. Breiman, *Machine Learning*, no. 24, ch. Bagging Predictors, pp. 123–140. Boston: Kluwer Academic Publishers, 1996.

[36] C. D. Sutton, *Handbook Statistics*, vol. 24, ch. Classification and Regression Trees, Bagging and Boosting. Amsterdam: Elsevier B. V, 2005.

[37] P. Derbeko, R. El-Yaniv, and R. Meir, "Variance optimized bagging," in *Proceedings of European Conference on Machine Learning*, pp. 60–71, 2002.

[38] M. Grant and S. Boyd, *CVX Users' Guide (Version 1.21)*, Apr. 2011, http://cvxr.com/cvx/cvx_usrguide.pdf.

[39] H. Drucker, "Improving regressors using boosting techniques," in *Proceedings of Fourteenth International Conference on Machine Learning*, D. H. Fisher and J. M. Kauffmann, Eds., pp. 107–115, 1997.

[40] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," in *Machine Learning: Proceedings of Thirteenth International Conference*, pp. 148–156, 1996.

Appendices

Appendix A

Recursive Partitioning Algorithm

Algorithm A.1 Recursive Partitioning Algorithm

```
B$_1(\mathbf{x}) \leftarrow 1$
For $M = 2$ to $M = M_{max}$ do: lof* $\leftarrow \infty$
    For $m = 1$ to $M - 1$ do:
        For $\nu = 1$ to $n$ do:
            For $t \in \{x_{\nu j}/B_m(\mathbf{x}_j) > 0\}$
                $g \leftarrow \sum_{i \neq m} a_i B_i(\mathbf{x}) + a_m B_m(\mathbf{x}) H\left[x_\nu - t\right] + a_M B_m(\mathbf{x}) H\left[-(x_\nu - t)\right]$
                lof $\leftarrow \min_{a_1,\ldots,a_M}$ LOF(g)
                if lof<lof*,then lof* $\leftarrow$ lof; $m^* \leftarrow m$;$\nu^* \leftarrow \nu$; $t^* \leftarrow t$ end if
            end for
        end for
    end for
    $B_M(\mathbf{x}) \leftarrow B_{m^*} H\left[-(x_{\nu^*} - t^*)\right]$
    $B_{m^*}(\mathbf{x}) \leftarrow B_{m^*} H\left[x_{\nu^*} - t^*\right]$
end for
end algorithm
```

Appendix B

MARS Algorithms

B.1 MARS-Forward Stepwise

Algorithm B.1 MARS-Forward Stepwise

```
$B_1(\mathbf{x}) \leftarrow 1; M \leftarrow 2$
Loop until $M > M_{max}$ :lof* $\leftarrow \infty$
    For $m = 1$ to $M - 1$ do:
        For $v \notin v(k, m)$ such that $1 \leq k \leq K_m$
            For $t \in \{x_{\nu j}/B_m(\mathbf{x}_j) > 0\}$
                $g \leftarrow \sum_{i=1}^{M-1} a_i B_i(\mathbf{x}) + a_M B_m(\mathbf{x}) H[x_\nu - t] + a_{M+1} B_m(\mathbf{x}) H[-(x_\nu - t)]$
                lof $\leftarrow \min_{a_1,\ldots,a_M}$ LOF(g)
                if lof<lof*,then lof* $\leftarrow$ lof; $m^* \leftarrow m$;$\nu^* \leftarrow \nu$; $t^* \leftarrow t$ end if
            end for
        end for
    end for
    $B_M(\mathbf{x}) \leftarrow B_{m^*} H[-(x_{\nu^*} - t^*)]$
    $B_{M+1}(\mathbf{x}) \leftarrow B_{m^*} H[x_{\nu^*} - t^*]$
    $M \leftarrow M + 2$
end for
```

B.2 MARS-Backwards Stepwise

Algorithm B.2 MARS-Backwards Stepwise

```
$J^* = (1, 2, \ldots, M_{max}); K^* \leftarrow J^*$
lof* $\leftarrow \min_{a_j/j \in J^*} LOF(\sum_{j \in J^*} a_j B_j(\mathbf{x}))$
For $M = M_{max}$ to 2 do: $b \leftarrow \infty; L \leftarrow K^*$
    For $m = 2$ to $M$ do: $K \leftarrow L - \{m\}$
        lof$\leftarrow \min_{a_k/k \in K} LOF(\sum_{k \in K} a_k B_k(\mathbf{x}))$
        if lof < v, then b$\leftarrow lof; K^* \leftarrow K$end if
        if lof <lof*, tehn lof* $\leftarrow$lof;$J^* \leftarrow K$ end if
    end for
end for
end algorithm
```

Appendix C

M5′ Pseudocode

Algorithm C.1 M5′

```
M5′ (examples)
    SD=sd {examples}
    for each $k$-valued enumerateed attributes
        convert into $k$-$1$ synthetis binary attributes
    root=new_node ; root.examples = examples
    split (root); prune (root)
    print_tree($root$)
split(nodes)
    if sizeof(node.examples)$<4$ or sd(node.examples)$<0.05\times$SD
        node.type=LEAP
    else
        node.type= INTERIOR
        for each old continuous and new binary attribute
            for all possible split positions of the attribute
                calculate the attribute's SDR
            node.attribute=attribute with maximum SDR
            split (node.left); split(node.right)
prune(node)
    if node = INTERIOR then
        prune(node.left_child); prune(node.right_child)
        node.model = linear_regression(node)
        if subtree_.error(node) > error(node) then
            node.type=LEAP
subtree_left.r=node.right
    l=node.left;r=node.right
    if node = INTERIOR then
        return$\left(\frac{\text{sizeof(l.examples)}\times\text{subtree\_error(l)}+\text{sizeof(r.examples)}\times\text{subtree\_error(r)}}{\text{sizeof(node.examples)}}\right)$
    else return error(node)
```

Appendix D

Variance Optimized Bagging

Algorithm D.1 Variance Optimized Bagging

```
Input:
    Number of bagged classifiers,$T$
    Number of efficient frontier points, $k$
    Training examples : $S = \{\mathbf{x}_n, y_n\}_{n=1}^{N}$
    Choose the base learning algorithm $\mathcal{H}$
Training:
    Generate $T$ bootstrap samples $B_1, \ldots, B_T$ from $S$
    Train $T$ models $h_1, h_2, \ldots, h_T$ such that $h_j \in \mathcal{H}$ is trained over $B_j$
    Evaluate $\bar{A}_j = \frac{1}{T-1} \sum_{i \neq j} A_j(B_i)$ for all $i = 1, \ldots, T$
    Evaluate $Q = \frac{1}{T-1} \sum_{j=1}^{T} (A_j - \bar{A})(A_j - \bar{A})^T$
            where $A_j = [A_j(B_1), \ldots, A_j(B_T)] \quad j = 1, \ldots, T$
            and $\quad \bar{A} = \frac{1}{T} \sum_{j=1}^{T} A_j$
    Choose $k$ uniformly spread points $a_1, \ldots, a_k$ in $[\min_j \bar{A}_j, \max_j \bar{A}_j]$
    Solve $k$ instances of the following Quadratic problem for all $i = 1, \ldots, k$. Let $\mathbf{w}_i$ and
    $(a'_i, \sigma_i)$ be the resulting weight vector and mean-variance pair corresponding to $a_i$
                $\min_{\mathbf{w}} \frac{1}{2} \mathbf{w}^T Q \mathbf{w}$
            such that $\left(\bar{A}_1, \bar{A}_2, \ldots, \bar{A}_T\right)^T \mathbf{w} \geq a_i$
                and $\quad \sum_j w_j = 1, \quad \mathbf{w} \geq 0$
    Let $p_0$ be the mean of all the responses obtained using all the predictors. Output:
    Vogging weight vector $\mathbf{w}_{i^*}$ with $i^* = arg\max_i \frac{a'_i - p_0}{\sigma_i}$
```

Appendix E

Boosting

Algorithm E.1 Adaboost.R2 Algorithm

```
Input:
    Training examples : ${\mathbf{x}_n, y_n}_{n=1}^N$
    Choose the base learning algorithm
    Number of iterations (T)
Initialize:
    iteration number, $t = 1$
    Distribution $D_t(i) = \frac{1}{N} \quad i = 1, \ldots, N$
    Average loss function, $L_t^{avg} = 0$
Iterate while $L_t^a vg < 0.5$ or $t \leq T$
    Obtain a regression model, $f_t(x) \to y$ using the base learning algorithm and distribution $D_t$
    Calculate loss for each training example as: $l_t(i) = |f_t(x_i) - y_i|$
    Calculate the loss function $L_t(i)$ for each training example as : $L_t(i) = \frac{l_t(i)}{MRES}$
    where                                        $MRES = \max_i l_t(i)$
    Calculate an average loss as: $L_t^{avg} = \sum_{i=1}^N L_t(i) D_t(i)$
    Set $\beta_t = \frac{L_t^{avg}}{1 - L_t^{avg}}$
    Update distribution $D_t$ as: $D_{t+1}(i) = \frac{D_t(i)\beta_t^{(1-L_t(i))}}{Z_t}$
    where $Z_t$ is a normalization factor chosen such that $D_{t+1}$ will be a distribution.
    set $t = t + 1$
Output:
    Final hypothesis,
        $f_{final}(x) = inf \left[ y \in Y : \sum_{t:f_t(x) \leq y} \log \frac{1}{\beta_t} \geq \frac{1}{2} \sum_t \log \frac{1}{\beta_t} \right]$
```