8-2008

A kernelized genetic algorithm decision tree with information criteria

James Michael Lanning
University of Tennessee

To the Graduate Council:

I am submitting herewith a dissertation written by James Michael Lanning entitled "A kernelized genetic algorithm decision tree with information criteria." I have examined the final electronic copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Business Administration.

Hamparsum Bozdogan, Major Professor

We have read this dissertation and recommend its acceptance:

Accepted for the Council:

Carolyn R. Hodges

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

To the Graduate Council:

I am submitting herewith a dissertation written by James Michael Lanning entitled "Data Mining Visualization: A Kernelized Genetic Algorithm Decision Tree with Information Criteria." I have examined the final electronic copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Business Administration.

Hamparsum Bozdogan, Major Professor

We have read this dissertation
and recommend its acceptance:

J. Wesley Hines

Mary Leitnaker

Russell Zaretzki

Accepted for the Council:

Carolyn R. Hodges, Vice Provost and
Dean of the Graduate School

(Original signatures are on file with official student records)

A Kernelized Genetic Algorithm Decision Tree with Information Criteria

A Dissertation Presented for the

Doctor of Philosophy Degree

The University of Tennessee, Knoxville

James Michael Lanning

August 2008

Dedication

This dissertation is dedicated to my alter ego

(who shall remain nameless)

for not waging war with the natives

and my children

(Sheena, Rocko, and Dakota)

Acknowledgments

First and foremost, I am deeply indebted to my family, especially my parents, Faye Lanning, and my father, Jim Lanning. Where I am today is in no small part due to their love, support, and patience.

I additionally would like to thank my advisor, Dr. Hamparsum Bozdogan for directing me towards this challenging problem and his generous encouragement and patience throughout.

Abstract

Decision trees are one of the most widely used data mining models with a long history in machine learning, statistics, and pattern recognition. A main advantage of the decision trees is that the resulting data partitioning model can be easily understood by both the data analyst and customer. This is in comparison to some more powerful kernel related models such as Radial Basis Function (RBF) Networks and Support Vector Machines.

In recent literature, the decision tree has been used as part of a two-step training algorithm for RBF networks. However, the primary function of the decision tree is not model visualization but dividing the input data into initial potential radial basis spaces.

In this dissertation, the kernel trick using Mercer's condition is applied during the splitting of the input data through the guidance of a decision tree. This allows the algorithm to search for the best split using the projected feature space information while remaining in the current data space. The decision tree will capture the information of the linear split in the projected feature space and present the corresponding non-linear split of the input data space.

Using a genetic search algorithm, Bozdogan's Information Complexity criterion (ICOMP) performs as a fitness function to determine the best splits, control model complexity, subset input variables, and decide the optimal choice of kernel function. The decision tree is then applied to radial basis function networks in the areas of regression, nominal classification, and ordinal prediction.

Keywords and Phrases *Classification Tree; Data Mining; Decision Tree; Genetic Algorithm; Information Criteria; Kernel Function; Kernel Trick; Ordinal Tree; Radial Basis Functions; Regression Tree; and Support Vector Machine*

Contents

List of Figures

List of Tables

Chapter 1

Introduction

> *There is no true interpretation of anything; interpretation is a vehicle in the service of human comprehension. The value of interpretation is in enabling others to fruitfully think about an idea.*
>
> - Andreas Buja

With the low cost of data storage, large and complex data sets have become routinely collected in all fields of business, science, and engineering. Much interest has arisen in developing methods that can discover the implicit and non-trivial relationships that are believed to exist within these data sets.

Phrases such as Knowledge Discovery in Databases, Statistical Learning, Learning from Data, Data Mining, and Machine Learning are often used to describe this process by different fields depending on their philosophy. While each has its strengths and weaknesses, the later usually from working in isolation of other disciplines (statistics, engineering, and computer science, etc), all share the endeavor of find interesting trends or patterns in order to guide decisions about future activities.

In building a predictive model, understanding a data set's structure through the ability to visualize and interpret its complex nonlinear relationship is an important tool for knowledge extraction. This is especially true with many a customers need to understand possible underlying structures in

their data along with a typical trepidation and suspicion of any analytical or statistical method or model.

While using these patterns to generate a predictive model is important and usually the final goal, often lost for both the customer and the analyst is model interpretation and understanding. This is drawback for many advanced statistical modeling techniques. Extremes include the black-box domain of traditional neural networks and kernel based methods versus the information loss of complex structures when using a more interpretable method such as multivariate linear regression.

The principal focus of this thesis is the development of an advanced non-linear model that is both interpretable for the analyst and customer.

1.1 Dissertation Overview

This dissertation explores the use of kernel based data mining techniques in combination with decision trees as a structure in the visualization of resulting radial basis function models. Through the use of the kernel trick and the application of Dr. Bozdogan's information criteria as a fitness function for goodness-of-fit, a genetic search algorithm handles the variable, kernel, and best model subset selection.

The decision tree is applied to radial basis function networks in the areas of classification, ordinal prediction, and regression.

1.1.1 Kernel Based Methods

Kernel based methods, the kernel trick, is a non-parametric method used for nonlinear data analysis. The kernel trick can develop nonlinear generalizations of any algorithm that can be cast in the term of dot products, implicitly mapping input data, $\mathbf{X}$, into a higher dimension feature space, $\mathbf{F}$, via a nonlinear function.

$$\Phi : \mathbf{X} \longrightarrow \mathbf{F} \tag{1.1}$$
$$\mathbf{x} \longmapsto \phi(\mathbf{x})$$

The similarity measure is defined from the dot product in the feature space $\mathbf{F}$:

$$\mathbf{K}(\mathbf{x}, \mathbf{x}') \triangleq (\phi(\mathbf{x}) \cdot \phi(\mathbf{x}')) = \left\langle \phi(\mathbf{x}), \phi(\mathbf{x}') \right\rangle \tag{1.2}$$

where the kernel function $\mathbf{K}$ is a Mercer kernel such that

1. $\mathbf{K}(\mathbf{x}, \mathbf{x}')$ is continuous,
2. $\mathbf{K}(\mathbf{x}, \mathbf{x}') = \mathbf{K}(\mathbf{x}, \mathbf{x}')$ (Symmetric), and
3. $\mathbf{K}(\mathbf{x}, \mathbf{x}')$ is positive definite.

Theorem 1.1.1 Mercer's Theorem

A symmetric function $\mathbf{K}(\mathbf{x}, \mathbf{x}\prime)$ can be expressed as an inner product

$$\mathbf{K}(\mathbf{x}, \mathbf{x}') = \left\langle \phi(\mathbf{x}), \phi(\mathbf{x}') \right\rangle \tag{1.3}$$

for some ϕ if and only if $\mathbf{K}(\mathbf{x}, \mathbf{x}')$ is positive semi definite, i.e.

$$\int \mathbf{K}(\mathbf{x}, \mathbf{x}') g(\mathbf{x}) g(\mathbf{x}') d\mathbf{x} d\mathbf{x}' \geq 0 \qquad \forall g \tag{1.4}$$

or, equivalently

$$\begin{bmatrix} \mathbf{K}(\mathbf{x}_1, \mathbf{x}_1) & \mathbf{K}(\mathbf{x}_1, \mathbf{x}_2) & \cdots & \mathbf{K}(\mathbf{x}_1, \mathbf{x}_n) \\ \mathbf{K}(\mathbf{x}_2, \mathbf{x}_1) & \ddots & & \\ \vdots & & & \\ \mathbf{K}(\mathbf{x}_n, \mathbf{x}_1) & & & \mathbf{K}(\mathbf{x}_n, \mathbf{x}_n) \end{bmatrix} \text{ is psd for any collection } \{\mathbf{x}_1, \ldots \mathbf{x}_n\}. \tag{1.5}$$

1.1.1.1 Kernel Trick

In recent years, the kernel trick has been successfully introduced into various machine learning algorithms, such as Kernel Principal Component Analysis , Kernel Fisher Discriminant, and Kernel Independent Component Analysis. Achieving a high state of performance in the many areas where it has been applied (Tipping, 2000 and 2001) such as Support Vector Machines (SVM) (Vapnik, 1997).

The following Figure 1.1 illustrates a standard example from the literature (Schölkopf and Smola, 2002) used to explain the idea of mapping the data (input) space into another dot product space known as the feature space. The feature map

$$\begin{aligned} \Phi : \Re^2 &\longrightarrow \Re^3 \\ \phi(x_1, x_2) &= (x_1^2, \sqrt{2}x_1x_2, x_2^2) \end{aligned} \tag{1.6}$$

maps $\Re^2$ data into a linearly separable plane in $\Re^3$ where the decision boundaries will be hyperplanes of the form

$$w_1x_1^2 + w_2\sqrt{2}x_1x_2 + w_3x_2^2 = 0 \tag{1.7}$$

which is the equation of an ellipse.

Using the Mercer kernel trick, the inner product of the feature vectors $\phi(x)$ and $\phi(w)$ in $\Re^3$ can be computed by squaring the inner product of the data vectors $\mathbf{x}$ and $\mathbf{w}$ in $\Re^2$.

$$\begin{aligned} K(\mathbf{x}, \mathbf{w}) &= \langle \phi(\mathbf{x}), \phi(\mathbf{w}) \rangle \\ K(\mathbf{x}, \mathbf{w}) &= w_1^2x_1^2 + 2w_1w_2x_1x_2 + w_2^2x_2^2 \\ K(\mathbf{x}, \mathbf{w}) &= (w_1x_1 + w_2x_2)^2 \\ K(\mathbf{x}, \mathbf{w}) &= (\langle \mathbf{x}, \mathbf{w} \rangle)^2 . \end{aligned} \tag{1.8}$$

Instead of mapping the data via ϕ and computing the inner product, the mapping can be left completely implicit and performed in the data space in one step. As a result, one doesn't need to

know ϕ, just how to compute the Mercer kernel, $K(\mathbf{x}, \mathbf{x}')$.which will be used in place of the Gram matrix $\mathbf{G}$ of all inner products of $\mathbf{X}$. as long as the inner product $\mathbf{K}(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle$ can be evaluated efficiently, where $\mathbf{G}$ is.

$$\mathbf{G} = \begin{bmatrix} \mathbf{x}_1^T\mathbf{x}_1 & \mathbf{x}_1^T\mathbf{x}_2 & \cdots & \mathbf{x}_1^T\mathbf{x}_n \\ \mathbf{x}_2^T\mathbf{x}_1 & \ddots & & \\ \vdots & & & \\ \mathbf{x}_n^T\mathbf{x}_1 & & & \mathbf{x}_n^T\mathbf{x}_n \end{bmatrix}_{nxn} = \mathbf{X}\mathbf{X}^T, \tag{1.9}$$

where $\mathbf{X}_{n\times d}$, containing all the data, is called the design or model matrix.

In some cases, this inner product can be evaluated more efficiently than the feature vector, which can be infinite dimensional in principle allowing nonlinear problem solving with learning algorithms using linear algebra and analytic geometry.

1.1.1.2 Reproducing Kernel Hilbert Space and the Kernel Trick

For completeness, the kernel trick is presented from the Reproducing Kernel Hilbert Space (RKHS) view in this section.

The RKHS is a smooth restricted space in the Hilbert space, which contains many non-smooth functions.

Definition 1.1.2 Hilbert Space is a complete dot product space.

The Hilbert space is an infinite-dimensional Euclidean vector space with an inner product $\langle \cdot, \cdot \rangle$ that obeys the following conditions

$$\begin{aligned} \langle x + y, w \rangle &= \langle x, w \rangle + \langle y, w \rangle \\ \langle ax, w \rangle &= a \langle x, w \rangle \\ \langle x, w \rangle &= \langle w, x \rangle \\ \langle x, x \rangle &\geq 0 \\ \langle x, x \rangle &= 0 \longrightarrow x = 0. \end{aligned} \tag{1.10}$$

From $\langle \cdot, \cdot \rangle$ we get a norm $\|\cdot\|$ via $\|x\| = \langle x, x \rangle^{\frac{1}{2}}$ Adding all limit points of Cauchy sequences to the space yields a Hilbert Space, which is a complete inner product space.

Given a kernel $K(\mathbf{x}, \mathbf{x}')$, the RKHS is defined as

$$\mathbf{H} = f(x) = \sum_{i=1}^{m} \alpha_i k(x_i, x') \tag{1.11}$$

with the following dot product

$$\langle f, g \rangle = \sum_i \sum_j \alpha_i \beta_i k(x_i, x_j) \tag{1.12}$$

which implies the reproducing property

$$\begin{aligned} \langle k(\cdot, x), f \rangle &= f(x) \\ \big\langle k(\cdot, x), k(\cdot, x') \big\rangle &= k(x, x') \end{aligned} \tag{1.13}$$

Define a reproducing kernel map as

$$\Phi : x \longrightarrow k(\cdot, x)$$

where to each point x in the original space is the associated function $k(\cdot, x)$.

Then from the reproducing property,

$$\big\langle \Phi(x), \Phi(x') \big\rangle = \big\langle k(\cdot, x), k(\cdot, x') \big\rangle = k(x, x') \tag{1.14}$$

which is the kernel trick.

1.1.2 Information Measure of Complexity (ICOMP)

The choice of the best radial function and subset selection is not simple nor automatic. In this paper, an Informational Measure of Complexity (ICOMP) criterion of Bozdogan (1988, 1990, 1994,

2000, 2004) is applied for feature variable selection, best model subset selection, and goodness-of-fit for the predicted model.

Bozdogan's information criteria used in this thesis is based on the Kullback-Leibler information (or distance). This type of criteria include Akaike's Information Criteria (AIC) which measures loss of information as a lack of fit (maximized likelihood function) plus a lack of model parsimony (two times the number of estimated parameters).

- Akaike's (1973) information criterion (AIC):

$$AIC = -2\log L(\widehat{\theta}) + 2(\#\ model\ parameters) \tag{1.15}$$

where $L(\widehat{\theta})$ is the maximized likelihood function.

The Informational Measure of Complexity (ICOMP) criteria extends the AIC criteria with a penalty due to increased complexity of the system. Examples include

- Bozdogan's (1990) Consistent AIC with Fisher Information Matrix (CAICF):

$$CAICF = -2\log L(\widehat{\theta}) + k[\log(n) + 2] + \log\left|\widehat{F}\right| \tag{1.16}$$

where $\left|\widehat{F}\right|$ denotes the determinant of the estimated Fisher information and n is the number of observations.

Specifically, the method in this thesis will use

- Bozdogan's (1990) new information complexity (ICOMP) criterion:

$$ICOMP(Cov\beta) = -2\log L(\widehat{\theta}) + 2C_1(\widehat{F}^{-1}(\widehat{\theta})), \tag{1.17}$$

where $C_1(\bullet)$ is a maximal information theoretic measure of complexity of $\widehat{F}^{-1}$ defined by

$$C_1(\widehat{Cov}(\widehat{\beta})) = \frac{q}{2}\log\left[\frac{tr(\widehat{F}^{-1}(\widehat{\theta}))}{q}\right] - \frac{1}{2}\log|\widehat{F}^{-1}(\widehat{\theta})|, \tag{1.18}$$

and where $q = dim(\widehat{F}^{-1}) \equiv rank(\widehat{F}^{-1})$.

In the literature, cross-validation based criteria are used for model selection and optimization. Due to the high dimensionality of the Kernel Method's feature space, these type of criteria are too time consuming and costly. In the results of this thesis, it suffices to use and score ICOMP.

1.2 Approach and Strategy

Tree-structured decision models are nonparametric approaches that divide the data space into regions in which a class, classification; constant, regression; or a model is assigned. While not based on assumptions of normality and user-specified model statements, the resulting tree-structured predictors can be easy to use and be generated by relatively simple functions of the input variables.

The method involving decision trees to initialize the centers and radii for RBF networks was first suggested by (M. Kubat and I. Ivanova,1995) in the context of classification rather than regression with further elaboration of the idea appearing in (Kubat, 1998). The method has been reviewed and built upon by (Orr, 1999). Radial basis function network (RBFN) (Broomhead and Lowe, 1988; Moody and Darken, 1989) is a type of artificial network for applications to problems of supervised learning e.g. regression, classification and time series prediction.

The tree generated in this thesis is not used to generate a pool of possible radial basis locations that are superimposed upon a regression or classification tree's node regions. The tree in this dissertation, much more like a model tree, determines the best radial basis selection at each split for a radial basis function network. However, each tree node is not a model but an individual radial basis function resulting in a structure that is dendogram in nature, describing the clustering of the data space in relation to the training output target. Furthermore, the final goal is not to discard the tree during the optimization of the radial basis functions, but retain the decision tree for customer visualization.

The strategy presented is more in line with the current strategies in kernel methods to find sparse approximations for Gaussian process regression and classification to meet computational limitations (Rasmussen, 1996-2005 ,Seeger, 2004; Smola, 2001; and Tipping, 2001). Where the goal in the previously mentioned is to find a suitable reduce ranked approximation of the kernel feature space using only a subset of latent variables, the goal in this thesis is to use the kernel method to

determine the resulting RBF network's basis functions of the input data space using the reduced ranked approximation.

Gaussian process's have traditionally used the marginal likelihood to learn the parameters of the kernel function and use the radial basis functions centered on the training inputs, as in the Relevance Vector Machine (RVM) (Tipping, 2001), the decision tree is dividing the data space allowing the individual radial basis function's parameter's to be learned.

Kernel methods have been applied to regression trees (Geurts, 2006). By applying the kernel clustering method to the output space instead of the input predictor space, they tend to suffer the main problem of kernel methods in interpretation. The mapping between the kernel and data space is not 1 to 1, or translation invariant. There are also Gaussian process trees, but this is more in line with model trees with each node being an individual Gaussian process.

1.3 Dissertation Organization

The structure of the following dissertation is as follows.

In Chapter 2, the background material first describes linear basis functions in the context of the regression function. This is a basic lead in to the radial basis function networks. Regularization is then described in the context of ordinary least squares (ridge regression) which will be used in solving for the weights of the RBF network. The reader is then introduced to the kernel trick in the context of least squares. Applying the kernel trick using linear algebra and analytic geometry allowing nonlinear problem solving with learning algorithms is then illustrated.

Chapter 3 explains the structure of the variable selection genetic algorithm with the idea of using decision trees for locating the centers of the basis functions. The standard greedy search algorithm is replaced with a triple layer genetic algorithm to handle the increased search space. Modifications for allowing multiple predictors for splitting the tree nodes and selecting the prediction operators are presented. The genetic algorithm will use information criteria as the fitness function to control the complexity of the regression tree during the subset of the variables in the growth and pruning of the regression tree.

Chapter 4 presents the modification to the decision tree's algorithm using Gaussian Processes to allow the application of the kernelized decision tree to function as a regression tree. Bayesian linear regression is presented from both the weight and function space viewpoints. The application is illustrated with two data sets, the Boston Housing and Auto-mpg data.

Chapter 5 builds upon the material presented in Chapter 4 to build a Gaussian Process classification tree with kernelized linear discriminant analysis (LDA) through multivariate regression. Flexible Discriminant Analysis which allows LDA to be performed as a multi-response linear regression using optimal scores to represent the classes is presented. The application of the kernelized LDA with respect to nominal classification is illustrated with two data sets, the Vowel and Wine data.

Chapter 6 then extends Chapter 5 to allow for the classification of ordinal responses. Utilizing the fact that Flexible Discriminant Analysis allows the use of the multi-response linear regression with an indicator membership matrix, the nominal classification tree is modified to handle ordinal attribute data by determining the expected class count according to cumulative nested grouping.

Chapter 7 closes the dissertation with the discussion of the results and conclusions.

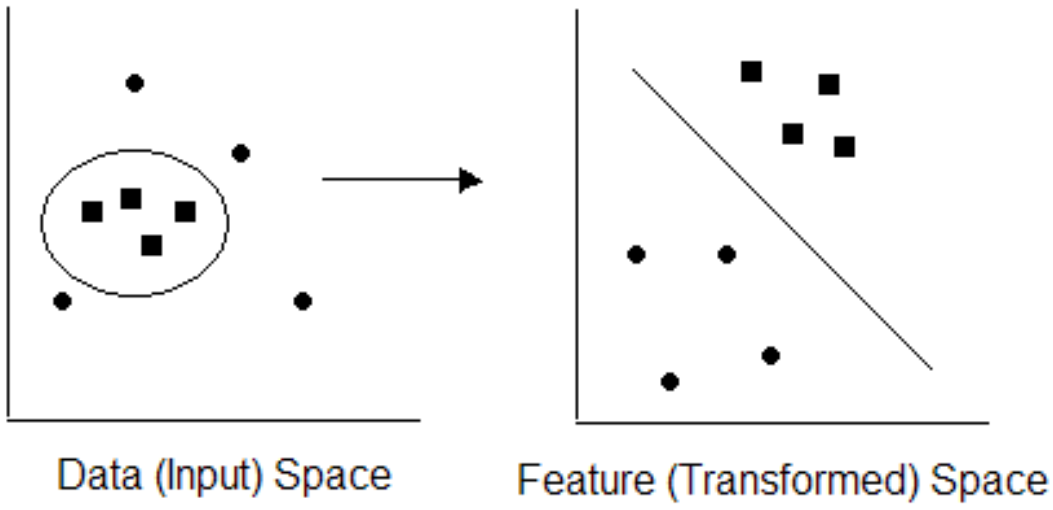

Figure 1.1: Mapping the training data into a higher-dimensional feature space with an existing separating hyperplane

Chapter 2

Background Material

Figure 2.1 illustrates the traditional Radial basis function networks (RBF) (Broomhead and Lowe, 1988; Moody and Darken, 1989) with a single hidden layer given by the model

$$\widehat{f}(\mathbf{X}) = \widehat{\mathbf{y}} = \sum_{j=1}^{m} \widehat{w}_j H_j(\mathbf{X}) \tag{2.1}$$

which is linear in the weights, $\{w_j\}_{j=1}^{m}$.

A RBF network is a type of artificial network for applications to problems of supervised learning e.g. regression, classification and time series prediction. When used for classification, the weighted output is filtered through a link transform function such that

$$\widehat{f}(\mathbf{X}) = \widehat{\mathbf{y}} = sign\left(\sum_{j=1}^{m} \widehat{w}_j H_j(\mathbf{X})\right). \tag{2.2}$$

A characteristic feature of RBF networks is the radial nature of the hidden unit transfer function $\{H_j(\mathbf{X})\}_{j=1}^{m}$, which is monotonic for non-negative numbers. It depends only on the distance between the input x and the center c of each hidden unit, scaled by a metric or smoothing parameter h.

Figure 2.1 can also illustrate a generalized mixture model (GMM) or when the link function is sigmoidal a support vector machine (SVM). What is different between the structures is not the model being estimated but the algorithm used in the estimation.

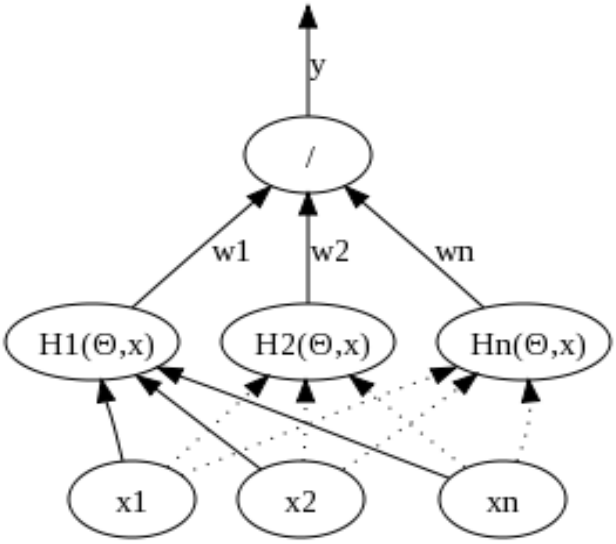

Figure 2.1: Radial Basis Function Network

The RBF network is traditionally trained in two stages, first defining and optimizing the radial basis functions and then determining the weights in the connections between the hidden units through minimizing the sum of squared errors typically using a steepest decent algorithm.

The SVM utilizes the kernel trick; and by restricting the solution to a maximized separating plane defined by support vectors, is solved through a constrained quadratic optimization problem.

The GMM is solved in a similar manner to the RBF network. The parameters are usually optimized through the Expectation Maximization (EM) Algorithm; and the weights solved through the use of maximum likelihood estimation.

This chapter reviews the background material of the RBF network building from the linear regression model. Included in the section on RBF networks, regularization and the EM algorithm are discussed. The final section discusses the method of the kernel trick as applied to SVMs. The kernel trick is also applied to least squares regression which will be used to solve for the weights of the RBF network in this dissertation. For more details and information of the following sections, the reader is referred to (Hastie et al, 2001) and (Written et al, 2000) for information on statistical data mining; and for kernel methods applications, the reader is referred to (Shawe-Taylor et al, 2004).

2.1 Linear Basis Functions

Suppose that the data for the joint distribution $\Pr(\mathbf{X}, \mathbf{y})$ arose from the statistical model

$$\mathbf{y} = f(\mathbf{X}) + \epsilon, \tag{2.3}$$

where the random error has $E[\epsilon] = 0$ and is independent of $\mathbf{X}$.

The goal is to find an approximation $\widehat{f}(\mathbf{X})$ to the unknown true function $f(\mathbf{X})$ for predicting $\mathbf{y}$ given the values of $\mathbf{X}$ where $\mathbf{X} \in \mathbb{R}^p$ is a real valued input vector and $Y \in \mathbb{R}$ is a real valued random output variable with a joint distribution $\Pr(\mathbf{X}, \mathbf{y})$. This goal is solved using a *Loss function*

$L(\mathbf{y}, f(\mathbf{X}))$ for penalizing errors during prediction. A common loss function is the squared error loss

$$L(\mathbf{y}, f(\mathbf{X})) = (\mathbf{y} - f(\mathbf{X}))^2. \tag{2.4}$$

The criterion then for choosing f is the *Expected Mean Squared Error* (EMSE):.

$$\begin{aligned} EMSE(f) &= E[(\mathbf{y} - f(\mathbf{X}))^2] \\ EMSE(f) &= \int (\mathbf{y} - f(\mathbf{X}))^2 \Pr(dx, dy). \end{aligned} \tag{2.5}$$

Conditioning on $\mathbf{X}$

$$EMSE(f) = E_X E_{Y|X}[(\mathbf{y} - f(\mathbf{X}))^2 \mid \mathbf{X}] \tag{2.6}$$

and minimizing $EMSE(f)$ point wise

$$f(\mathbf{X}) = \arg\min_c E_{Y|X}[(\mathbf{y} - c)^2 \mid \mathbf{X} = \mathbf{x}], \tag{2.7}$$

the solution is

$$f(\mathbf{X}) = E[\mathbf{y} \mid \mathbf{X} = \mathbf{x}]. \tag{2.8}$$

This is known as the *regression function* where the best prediction of $\mathbf{y}$ at any point $\mathbf{X} = \mathbf{x}$ is the conditional mean, when best is measured by average squared error.

2.1.1 Linear Model

A linear regression model, assumes that the regression function $E[\mathbf{y} \mid \mathbf{X}]$ is linear in the inputs $\mathbf{X}_{n\times p}$. The real-valued output $\mathbf{y}$ is predicted by the functional relation

$$\widehat{f}(\mathbf{X}) = \widehat{\mathbf{y}} = \widehat{w}_0 + \sum_{j=1}^{p} \widehat{w}_j \mathbf{X}_j, \tag{2.9}$$

where $\widehat{w}_0$ is the intercept, also known as *bias*. A constant variable 1 is often included in $\mathbf{X}$, $\widehat{w}_0$ included in the vector of **w**eight coefficients $\widehat{w}$, and the linear model written in the vector form as

an inner product:

$$\widehat{f}(\mathbf{X}) = \widehat{\mathbf{y}} = \sum_{j=1}^{p} \widehat{w}_j \mathbf{X}_j = \mathbf{X}^T \mathbf{w}. \tag{2.10}$$

Here $\widehat{\mathbf{y}}$ is an $n \times 1$ vector; in general $\mathbf{Y}$ can be a $n \times q$ matrix, in which case $\widehat{\mathbf{W}}$ would be a $p \times q$ matrix of coefficients.

A popular method to estimate the parameters, $\mathbf{w}$, is *ordinary least squares*, in which the weight coefficients are chosen as to minimize the residual sum of squares (minimizing the $EMSE$). Plugging the linear model for $f(\mathbf{X})$ into equation (2.5), we have

$$\begin{aligned} EMSE(f) &= E[(\mathbf{y} - \mathbf{X}^T\mathbf{w})^2] \\ EMSE(f) &= (\mathbf{y} - \mathbf{X}\mathbf{w})^T(\mathbf{y} - \mathbf{X}\mathbf{w}), \end{aligned} \tag{2.11}$$

and differentiating with respect to $\mathbf{w}$, we have

$$\begin{aligned} \frac{dEMSE}{d\mathbf{w}} &= -2\mathbf{X}^T(\mathbf{y} - \mathbf{X}\mathbf{w}) \\ \frac{d^2EMSE}{d\mathbf{w}d\mathbf{w}^T} &= -2\mathbf{X}^T\mathbf{X}. \end{aligned} \tag{2.12}$$

Assuming $\mathbf{X}$ is non singular and $\mathbf{X}^T\mathbf{X}$ is therefore positive definite, the first derivative is set to zero to obtain the unique solution

$$\widehat{\mathbf{w}} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}. \tag{2.13}$$

Another more general method for estimation is *maximum likelihood estimation* where the values of the density parameters, θ, are those for which the probability of the observed sample is largest. Least squares for the additive error model , with $\epsilon \sim N(0, \sigma^2)$, is equivalent to maximum likelihood using the conditional likelihood

$$\Pr(\mathbf{y} \mid \mathbf{X}, \theta) = \mathbf{N}(f(\theta = \mathbf{w}, \mathbf{X}), \sigma^2). \tag{2.14}$$

The log-likelihood of the data is then

$$l(\theta) = \sum_{i=1}^{n} \log \Pr_{\theta}(y_i) \tag{2.15}$$
$$l(\theta) = -\frac{n}{2}\log(2\pi) - n\log\sigma - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(\mathbf{y} - f(\theta, \mathbf{X}))^2$$

in which the only term involving the density parameters θ is the last, so that the likelihood is minimized by minimizing the residual sum of squares.

2.1.2 Basis Functions

The regression function $E[\mathbf{y} \mid \mathbf{X}]$ is often nonlinear and non additive in $\mathbf{X}$. A popular approach to address non linearity is to replace the inputs $\mathbf{X}$ with *linear basis functions*, which are transformations of $\mathbf{X}$. The model is expressed as a linear combination of a set of m fixed functions

$$\widehat{f}(\mathbf{X}) = \widehat{\mathbf{y}} = \sum_{j=1}^{m} \widehat{w}_j B_j(\mathbf{X}). \tag{2.16}$$

The flexibility of $\widehat{f}(\mathbf{X})$ in its ability to fit many different functions, derives from the freedom to choose different values for the weights. Once the basis functions B_m have been determined and any parameters which they might contain are considered fixed, the models are linear in these new variables, and fitting proceeds as in the base linear model. If this is not the case and the basis functions can change during the learning process, then the model is nonlinear.

Examples of some simple and widely used basis functions B_m include:

- $B_j(\mathbf{X}) = \mathbf{X}_j$, $j = 1, \ldots, m = p$ which recovers the original model

  For example, the simple straight line, $f(x) = b + ax$, which is a linear model with the basis functions $B_1(x) = 1$ and $B_2(x) = x$ and whose weights are $w_1 = b$ and $w_2 = a$.

- $B_j(\mathbf{X}) = \mathbf{X}_j^b$ or $B_j(\mathbf{X}) = \mathbf{X}_j\mathbf{X}_k$ or other variations allowing augmentation of the inputs with polynomials to achieve higher-order Taylor expansions.

- $B_j(\mathbf{X}) = \log(\mathbf{X}_j), \sqrt{\mathbf{X}_j}, \ldots$ permitting other nonlinear transformations.

2.1.3 Radial Basis Function Networks

Radial basis function (RBF) networks (Broomhead and Lowe, 1988; Moody and Darken, 1989) as shown in Figure 2.1 is a type of artificial network for applications to problems of supervised learning, e.g., regression, classification and time series prediction. Traditionally, RBF neural networks have a single hidden layer, equation (2.1), were **H** is the **H**idden basis function.

The characteristic feature of RBF networks is the radial nature of the hidden unit transfer function$\{H_j(\mathbf{X})\}_{j=1}^{m}$, which is monotonic for non-negative numbers, and depends only on the distance between the input x and the center c of each hidden unit, scaled by a bandwidth or smoothing parameter h.

The RBF network is typically trained in two stages:

Stage 1: The parameters of the radial basis functions are initialized using unsupervised training. Classically, the location of the centers is usually conducted by some clustering algorithm such as k-means. The widths of the radial basis functions are then determined using a nearest-neighbor heuristic.

Stage 2: The weights in the connections between the hidden units and the output are determined through supervised learning. The sum of squared errors over the set of training input-output vector pairs is minimized typically using a steepest decent algorithm. The individual input-output training pairs are presented to the RBF network repeatedly until the error decreases to an acceptable level.

An RBF network is considered non-linear if the basis functions can move or change size or if there is more than one hidden layer otherwise the RBF network is considered linear.

2.2 Bias and Variance

For the regression function, given the true output, $\mathbf{y}$, the mean-squared-error is

$$MSE = E\left[\mathbf{y} - f(\mathbf{X})\right]^2 \tag{2.17}$$

which can be broken down into two components

$$MSE = \left(y - E\left[f(\mathbf{X})\right]\right)^2 + E\left[f(\mathbf{X}) - E\left[f(\mathbf{X})\right]^2\right]. \tag{2.18}$$

The first part is the bias and the second part is the variance.

If $E\left[f(\mathbf{X})\right] = \mathbf{y}$ for all $\mathbf{X}$ then the model is unbiased (the bias is zero). However, an unbiased model may still have a large mean-squared-error if it has a large variance.

Reconsider, the problem where we are trying to minimize sum of squared residuals

$$\widehat{\mathbf{w}} = \arg\min_{w}(EPE(f)) = \arg\min_{w} E[(\mathbf{y} - \mathbf{X}^T\mathbf{w})^2]. \tag{2.19}$$

Minimizing (2.19) leads to infinitely many standard regression solutions, since any function $\widehat{f}(\mathbf{X})$ passing through the training points is a solution.

The least squares model is smooth relying on the assumption that a linear model/decision boundary is appropriate. It has a low variance and potentially high bias. As a global model; a local regression model such as k-nearest neighbors that depends on a particular position will tend to exhibit high variance with low bias.

However, the least square estimates $\widehat{\mathbf{w}}$ often will have low bias but large variance. This will be the case if $\widehat{f}(\mathbf{X})$ is highly sensitive to the peculiarities (such as noise and the choice of sample points); and it is this sensitivity which causes regression problems to be ill-posed. With highly correlated variables in a linear regression model, the coefficients can become poorly determined and exhibit high variance.

Prediction accuracy can sometimes be improved by shrinking or setting some coefficients, $\widehat{\mathbf{w}}$, to zero. By doing so, the variance of the least squares estimates can be significantly reduced by

deliberately introducing a small amount of bias so that there may be an overall improvement in the model's prediction accuracy.

2.2.1 Regularization

When solving the regression function

$$\widehat{f}(\mathbf{X}) = E[\mathbf{y} \mid \mathbf{X} = \mathbf{x}] \tag{2.20}$$

the least squares solution is given by

$$\widehat{\mathbf{w}} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}. \tag{2.21}$$

The sum of squared residuals is minimized by

$$\widehat{\mathbf{w}} = \arg\min_{w}(EMSE(f)) = E[(\mathbf{y} - \mathbf{X}^T\mathbf{w})^2]. \tag{2.22}$$

When the data matrix, $\mathbf{X}$, is ill-conditioned (due to potential singularity), the least squares solution is unstable. Also, when the sample size is small in comparison to the dimensionality, the model performance may be poor even when the training error is small. The reason is that the regression function will fit the noise in a phenomenon known as over fitting. In order to prevent these two problems, a commonly used practice called regularization (Hoerl and Kennard, 1970) is employed. The regularized version of least square regression is Ridge Regression also known as weight decay.

2.2.2 Ridge Regression

Introducing bias is equivalent to restricting the range of functions for which a model can account. This is typically achieved by removing degrees of freedom. An example would be lowering the order of a polynomial or reducing the number of weights in a neural network.

Ridge regression does not explicitly remove degrees of freedom but instead reduces the effective number of parameters by shrinking the regression coefficients by imposing a penalty on their size. The ridge coefficients minimize a penalized residual sum of squares:

$$\begin{aligned} \widehat{\mathbf{w}}_{Global}^{ridge} &= \arg\min_{w} \left[(\mathbf{y}-\mathbf{X}\mathbf{w})^T(\mathbf{y}-\mathbf{X}\mathbf{w}) + \lambda w^T w\right] \\ \widehat{\mathbf{w}}_{Global}^{ridge} &= \left(\mathbf{X}^T\mathbf{X}+\lambda\mathbf{I}\right)^{-1}\mathbf{X}^T\mathbf{y}, \end{aligned} \tag{2.23}$$

where $\lambda \geq 0$ is a complexity parameter that controls the amount of shrinkage, the balance between fitting the data and avoiding the penalty.

A small value for λ means the data can be fit tightly without causing a large penalty; $\lambda = 0$ is the ordinary least squares solution. The larger the value of λ, the greater the amount of shrinkage favoring solutions involving small weights. This has the effect of smoothing the output function since large weights are usually required to produce a highly variable (rough) output function. For example, a extremely large coefficient can be canceled by a similarly large negative correlated coefficient. By imposing a size constraint, this phenomenon is prevented from occurring.

Of course, since the regularized solution is biased, the expected value of $\widehat{\mathbf{w}}^{ridge}$ is not equal to the "true" value of the regression coefficients. So the regularized solution will have no physical interpretation, but will improve the *prediction accuracy* of the model. The ridge solutions are also not equivariant under scaling of the inputs, which are normally standardized.

Standard or global ridge regression, with just one parameter, λ, to control the bias/variance trade-off, has difficulty with functions which have significantly different smoothness in different parts of the input space. The ridge regression model can be generalized from a global to a local model by associating a separate regularization parameter with each weight coefficient variable. Instead of treating all weights equally with the penalty term $\lambda\mathbf{w}^T\mathbf{w}$ we can treat them all separately and have a regularization parameter associated with each $(\lambda\mathbf{w})^T(\lambda\mathbf{w})$:

$$\begin{aligned} \widehat{w}_{local}^{ridge} &= \arg\min_{w} \left[(\mathbf{y}-\mathbf{X}\mathbf{w})^T(\mathbf{y}-\mathbf{X}\mathbf{w}) + (\lambda\mathbf{w})^T(\lambda\mathbf{w})\right] \\ \widehat{w}_{local}^{ridge} &= \left(\mathbf{X}^T\mathbf{X}+\mathbf{\Lambda}\right)^{-1}\mathbf{X}^T\mathbf{y}, \end{aligned} \tag{2.24}$$

where $\mathbf{\Lambda} = diag\{\lambda_j\}_{j=1}^{m}$ is a diagonal regularization parameter matrix.

In general there is nothing local about this form of weight decay. However, if we confine ourselves to local basis functions such as radial functions which are monotonically decreasing in their response then the smoothness produced by this form of ridge regression is controlled in a local fashion by the individual regularization parameters.

2.2.3 Regularization Parameter Optimization

The regularization parameter is typically optimized through methods such as cross validation. Several authors have proposed analytical procedures for choosing the optimal ridge parameter.

- Hoerl, Kennard, and Baldwin (1975)

$$\widehat{\lambda}_{HKB} = \frac{ms^2}{\widehat{\mathbf{w}}^T\widehat{\mathbf{w}}} \tag{2.25}$$

where $m = k$, the number of predictors not including the intercept term, n is the number of observations, s^2 is the estimated error variance using k predictors so that

$$s^2 = \frac{1}{(n-k+1)}(\mathbf{y} - \mathbf{X}\widehat{\mathbf{w}})^T(\mathbf{y} - \mathbf{X}\widehat{\mathbf{w}}) \tag{2.26}$$

- Lawless and Wang (1976)

$$\widehat{\lambda}_{LW} = \frac{ms^2}{\sum_{j=1}^{k}\widehat{w}_j^2\widehat{\lambda}} \tag{2.27}$$

as an estimator of $\widehat{\sigma}^2/\widehat{\sigma}_w^2$ by Bayesian argument

- Sclove (1973) Empirical Bayes Method

$$\widehat{\lambda}_S = \frac{\widehat{\sigma}^2}{\widehat{\sigma}_w^2} \tag{2.28}$$

where

$$\widehat{\sigma}^2 = \frac{1}{n}\mathbf{y}^T\left[\mathbf{I} - \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\right]\mathbf{y} \tag{2.29}$$

is the estimated residual variance and

$$\widehat{\sigma}_w^2 = \frac{\mathbf{y}^T\mathbf{y} - n\widehat{\sigma}^2}{tr(\mathbf{X}^T\mathbf{X})} \tag{2.30}$$

2.3 Kernelized Radial Basis Functions

The application of the kernel trick as outlined in Chapter 1 will be applied to the RBF network in this dissertation through the use of the kernelized regularized least squares.

By substituting a Mercer kernel for the Gram matrix as outlined in the following section, the RBF model

$$\widehat{f}(\mathbf{X}) = \widehat{\mathbf{y}} = \sum_{j=1}^{m} \widehat{w}_j H_j(\mathbf{X}) \tag{2.31}$$

the model can be stated in the original data space as

$$\widehat{\mathbf{y}}_{new} = K(\mathbf{X}_{new}, \mathbf{X})\widehat{\boldsymbol{\alpha}} \tag{2.32}$$

where $\widehat{\boldsymbol{\alpha}}$ is estimated as

$$\widehat{\boldsymbol{\alpha}} = (\mathbf{K} + \lambda\mathbf{I})^{-1}\mathbf{y} \tag{2.33}$$

2.3.1 Kernelized Regularized Least Squares Regression

In the section on linear models, it was noted that least squares for the additive error model

$$\widehat{f}(\mathbf{X}) = \widehat{\mathbf{y}} = \mathbf{X}^T\mathbf{w} + \boldsymbol{\epsilon} \tag{2.34}$$

with $\epsilon \sim N(0, \sigma^2)$, is equivalent to maximum likelihood using the conditional likelihood

$$\Pr(\mathbf{y} \mid \mathbf{X}, \theta) = \mathbf{N}(f(\theta = \mathbf{w}, \mathbf{X}), \sigma^2) \tag{2.35}$$

giving the maximum likelihood estimate of $\mathbf{w}$

$$(\mathbf{X}^T\mathbf{X})\widehat{\mathbf{w}}= \mathbf{X}^T\mathbf{y} \tag{2.36}$$

where $\mathbf{X}$ is $n \times p$, so that $\mathbf{X}^T\mathbf{X}$ is $p \times p$, and $\mathbf{X}^T\mathbf{y}$ is $p \times 1$, which do not depend on n for dimension.

The ridge coefficients were then defined to minimize a penalized residual sum of squares.

$$\begin{aligned} \widehat{\mathbf{w}}^{ridge}_{Global} &= \arg\min_w \left[(\mathbf{y} - \mathbf{X}\mathbf{w})^T(\mathbf{y} - \mathbf{X}\mathbf{w}) + \lambda w^T w\right] \\ \widehat{\mathbf{w}}^{ridge}_{Global} &= \left(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}\right)^{-1}\mathbf{X}^T\mathbf{y} \end{aligned} \tag{2.37}$$

where $\lambda \geq 0$ is a complexity parameter that controls the amount of shrinkage and $\lambda = 0$ is the ordinary least squares solution.

So that

$$\begin{aligned} \left(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}\right)\widehat{\mathbf{w}}^{ridge}_{Global} &= \mathbf{X}^T\mathbf{y} \\ \mathbf{X}^T\mathbf{X}\widehat{\mathbf{w}}^{\mathbf{ridge}}_{\mathbf{Global}} + \lambda\widehat{\mathbf{w}}^{ridge}_{Global} &= \mathbf{X}^T\mathbf{y} \\ \lambda\widehat{\mathbf{w}}^{ridge}_{Global} &= \mathbf{X}^T\mathbf{y} - \mathbf{X}^T\mathbf{X}\widehat{\mathbf{w}}^{\mathbf{ridge}}_{\mathbf{Global}} \\ \widehat{\mathbf{w}}^{ridge}_{Global} &= \lambda^{-1}\mathbf{X}^T\left(\mathbf{y} - \mathbf{X}\widehat{\mathbf{w}}^{\mathbf{ridge}}_{\mathbf{Global}}\right) \\ \widehat{\mathbf{w}}^{ridge}_{Global} &= \mathbf{X}^T\alpha \end{aligned} \tag{2.38}$$

then

$$\widehat{\mathbf{y}}_{new} = \mathbf{X}_{new}\widehat{\mathbf{w}} = \mathbf{X}_{new}\mathbf{X}^T\alpha \tag{2.39}$$

where

$$\begin{aligned} \alpha &= \lambda^{-1}\left(\mathbf{y} - \mathbf{X}\widehat{\mathbf{w}}_{\mathbf{Global}}^{\mathbf{ridge}}\right) \\ \alpha &= \lambda^{-1}\left(\mathbf{y} - \mathbf{X}\mathbf{X}^{\mathbf{T}}\alpha\right) \\ \mathbf{X}\mathbf{X}^{\mathbf{T}}\alpha + \lambda\alpha &= \mathbf{y} \\ \left(\mathbf{X}\mathbf{X}^{\mathbf{T}} + \lambda\mathbf{I}\right)\alpha &= \mathbf{y} \\ \alpha &= \left(\mathbf{X}\mathbf{X}^{\mathbf{T}} + \lambda\mathbf{I}\right)^{-1}\mathbf{y}. \end{aligned} \tag{2.40}$$

Then using the kernel trick as defined in the Introduction where one can implicitly map input data into a high dimension feature space via a nonlinear function:

$$\begin{aligned} \Phi : \mathbf{X} &\longrightarrow \mathbf{F} \\ \mathbf{x} &\longmapsto \phi(\mathbf{x}). \end{aligned} \tag{2.41}$$

By substituting a Mercer kernel trick for the Gram matrix, $\widehat{\boldsymbol{\alpha}}$ can be estimated as

$$\widehat{\boldsymbol{\alpha}} = (\mathbf{K} + \lambda\mathbf{I})^{-1}\mathbf{y} \tag{2.42}$$

and the evaluation of the regression function as

$$\widehat{\mathbf{y}}_{new} = K(\mathbf{X}_{new}, \mathbf{X})\widehat{\boldsymbol{\alpha}}. \tag{2.43}$$

Besides the issue of optimizing the smoothing parameter of the kernel to control poor generalization, one of the weaknesses of the kernel trick is the size of the Gram matrix leading to size of the $\boldsymbol{\alpha}$ matrix. Because of its nonparametric nature, $\mathbf{K}$ is an $n \times n$ matrix; $\boldsymbol{\alpha}$ is therefore $n \times 1$. When dealing with large databases, this will lead to the problem of storing an information matrix large as or larger than the original data set.

2.3.2 Support Vector Machines

An example of the interest in reducing the size of $\boldsymbol{\alpha}$ is in the method of Support Vector Machines (SVM) (Vapnik, 1997). The SVMs map the training data non linearly into a higher-dimensional feature space via the kernel trick and reduce the size of the $\boldsymbol{\alpha}$ matrix by constructing a separating hyperplane with maximum margin. Notice that in Figure (2.2), the solution of the hyperplane depends only on the four circled points. These training patterns define the support vectors, carrying all relevant information about the classification problem, and are uniquely solved as a constrained quadratic optimization problem.

The RBF model as shown in Figure 2.1 can then be illustrated as Figure 2.3

2.3.3 Statistical Kernel Density Estimation

Statistics as a field has had a long time practice of reducing information into sufficient statistics. An example which is related to this dissertation is that of use of kernel functions in density estimation

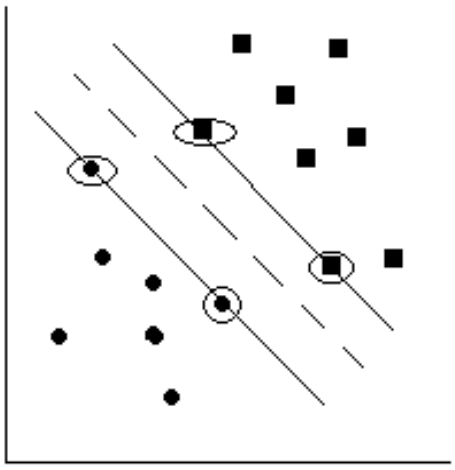

Figure 2.2: SVM Maximum Separating Hyperplane

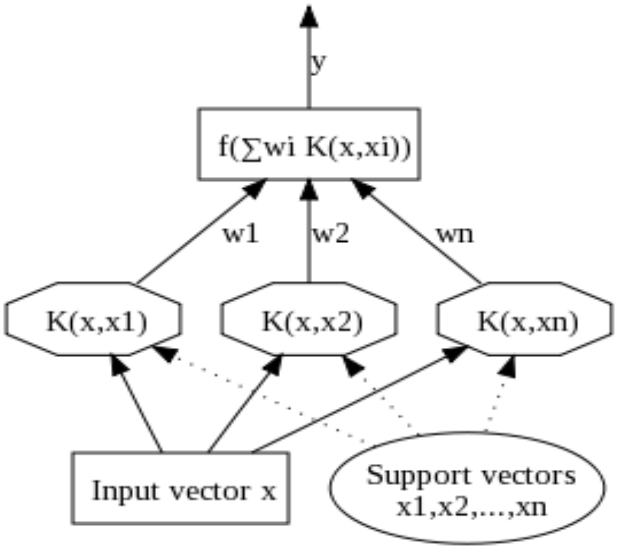

Figure 2.3: Support Vector Machine (SVM)

and their relation to finite mixtures. Note that kernel in this section is positive definite function, but is less specific than a Mercer kernel.

To achieve further flexibility with basis functions in estimating the data density function, a simple model is fitted at each observation point $\mathbf{X}_0$. Observations in a region local at the target point, $\mathbf{X}_0$, are used to fit the model such that the estimated function $\widehat{f}(\mathbf{X})$ is smooth in $\Re^p$. The localization is achieved through a weighing function or kernel $\mathbf{K}_H(\mathbf{X}_0, \mathbf{X}_i)$ which assigns a weight to $\mathbf{X}_i$ based on its distance from $\mathbf{X}_0$ where each observation is a $p-$dimensional vector.

The $1-$dimension or univariate kernel estimator is given by

$$\widehat{f}_{KER}(x) = \widehat{y} = \frac{1}{nh}\sum_{i=1}^{n} K(\frac{x - x_i}{h}) \tag{2.44}$$

where $K(t)$ is called a kernel. In the case of density estimation, $K(t)$ must be positive definite and $\int K(t)dt = 1$

Defining $K_h(t) = K(t/h)/h$, the estimate is sometimes written as

$$\widehat{f}_{KER}(x) = \widehat{y} = \frac{1}{nh}\sum_{i=1}^{n} K_h(x - x_i). \tag{2.45}$$

The kernels are indexed by a smoothing parameter h known as the *window width* which dictates the width of the neighborhood.

Examples of h for a few kernels K_h:

- For the Epanechikov kernel, h is the radius of the support region

$$K_h(t) = \left\{ \begin{array}{cc} \frac{3}{4}(1 - t^2) & -1 \leq t \leq 1 \\ 0 & \text{otherwise} \end{array} \right\} \tag{2.46}$$

- For the Gaussian kernel, h is the standard deviation

$$K_h(t) = \frac{1}{\sqrt{2\pi}} \exp\left\{\frac{-t^2}{2}\right\}, for -\infty \leq t \leq \infty \tag{2.47}$$

- For the $k-$nearest neighborhood, h is the number k of nearest neighbors.

The smoothing parameter is the only parameter that typically needs to be determined and is often determined off-line. (Kernel estimation is a memory-based technique: the model is the training set.). There is a natural bias-variance trade-off in determining the size of window width.

- A large h providing a wide window yields a smooth curve that averages over more observations, implying a lower variance but with a higher bias with the possibility of obscuring structure. The bias is higher because observations $\mathbf{X}_i$ are being used further away from $\mathbf{X}_0$ with no guarantee that $f(\mathbf{X}_i)$ will be close to $f(\mathbf{X}_0)$
- With a small h or a narrow window , the estimated function will tend to fit noise or spurious structures. The variance will larger and the bias smaller

The univariate kernel is easily expanded to the multivariate case. However, the model complexity increases as there may be a different window width in each dimension.

The simplest case for the $p-$ dimensional or multivariate kernel estimator is the product kernel

$$\widehat{f}_{KER}(\mathbf{x}) = \widehat{\mathbf{y}} = \frac{1}{nh_1 \ldots h_p} \sum_{i=1}^{n} \left\{ \prod_{j=1}^{p} K(\frac{\mathbf{x}_j - \mathbf{X}_{ij}}{h_j}) \right\}, \tag{2.48}$$

which is a product of a univariate kernel with a potentially different window width in each dimension. Also, one could also use a multivariate density kernel such as the multivariate Gaussian where $h_1, \ldots, h_p$ is given by the estimated variance covariance matrix.

While easier computationally to assume $h_j = h$, this can have the effect of creating regions in $\Re^p$ where none of the kernels has support due to the local estimating technique.

In order to remove the need to estimate a smoothing parameter, the model can be redefined as a Finite Mixture. As will be shown, this another way of presenting the same model where the smoothing parameter is replaced with a weight or mixing coefficient.

Finite mixtures assume that the density $f(\mathbf{X})$ can be modeled as the sum of c weighted densities, with $c << n$. The densities of the finite mixture can be any probability density function, univariate or multivariate,

$$\widehat{f}_{FM}(x) = \widehat{y} = \sum_{i=1}^{c} p_i g(x; \theta_i), \tag{2.49}$$

where p_i represents the weight or mixing coefficients for the ith group, $g(x; \theta_i)$ denotes a probability density, and $\sum p_i = 1$ for a density estimation, otherwise a linear mixture model.

The relationship with linear basis functions is clear with $m = c$ and $g(x; \theta_i) = B_i(x)$.

Also, the connection between finite mixtures and kernel density estimation is obvious. If the covariance matrices are constrained to be scalar $\Sigma_i = \sigma_i \mathbf{I}$ where $\sigma_i = \sigma > 0$ is fixed and $c \to n$, then the maximum likelihood estimate for 2.49 approaches the kernel density estimate 2.45 where $p_i = 1/n$ and $\widehat{\mu}_i = x_i$ in the case of the Gaussian kernel. A kernel estimate can be considered a special case of finite mixtures where $c = n$.

Since $c << n$, there is a significant computational savings in evaluating with the kernel method. With finite mixtures much of the computational burden is shifted to the estimation part of the problem.

2.4 Kernel Space Reduction

Instead of using support vectors using a maximum margin hyperplane, it is proposed to parameterize the $\boldsymbol{\alpha}$ matrix in much the same way that kernel density estimation is parameterized in finite mixture density estimation. Then $\boldsymbol{\alpha}$ will become an $c \times 1$ reduced matrix centered on the radial basis functions with $\mathbf{K}$ becoming an $n \times c$ matrix where c is the number of clusters. Following is an example simulation of the process using the kernelized least squares. While this dissertation will be developing a decision tree to define the basis functions, here the Expectation Maximization Algorithm will be utilized.

2.4.1 Expectation Maximization Algorithm

The Expectation Maximization (EM) algorithm is a standard iterative method in statistics for dealing with missing values in maximum likelihood parameter estimation (Dempster et al, 1977). See also (Meng and van Dyke,1977). Probability distribution parameters are estimated from observed (incomplete) data by iteratively maximizing the available data likelihood as a function of the parameters assuming that missing values are missing at random. Each iteration consists of an expectation E-step which finds the distribution for the unobserved variables and a maximization M-step which re-estimates the parameters of the model to be those with the maximum likelihood for the observed and missing data combined.

As shown, the RBF Network or mixture model will have the form

$$p(\mathbf{x} \mid \mathbf{\Psi}) = \sum_{j=1}^{m} \pi_j f(\mathbf{x} \mid \theta_j) \tag{2.50}$$

where $f(\cdot \mid \theta_j)$ denotes a RBF, θ_j the parameters occurring in $f_j(\cdot)$, and Ψ the complete collection of parameters occurring in the mixture model. Then the likelihood function is given by

$$L(\mathbf{\Psi}) = \sum_{i=1}^{n} \left[\sum_{j=1}^{m} \pi_j f(\mathbf{x} \mid \theta_j) \right] \tag{2.51}$$

Maximization of $L(\mathbf{\Psi})$ with respect to Ψ, for given data $\mathbf{X}$, yields the maximum likelihood estimate of Ψ. Equivalently, the usual quantity maximized is the log-likelihood.

$$l(\mathbf{\Psi}) = \ln L(\mathbf{\Psi}) \tag{2.52}$$

Let $\mathbf{Z}$ denote the complete version of the incomplete observed data set $\mathbf{X}$, and let the likelihood from $\mathbf{Z}$ be

$$g(\mathbf{y} \mid \mathbf{\Psi}) \tag{2.53}$$

The EM algorithm generates from some initial approximation, $\mathbf{\Psi}^{(t=0)}$, a sequence of estimates $\mathbf{\Psi}^{(t)}$. Each iteration of estimates consists of the double step

- E step: Evaluate $E\left[\log g(\mathbf{y} \mid \mathbf{\Psi}) \mid \mathbf{x}, \mathbf{\Psi}^{(t)}\right] = Q(\mathbf{\Psi}, \mathbf{\Psi}^{(t)})$
- M step: Find $\mathbf{\Psi} = \mathbf{\Psi}^{(t+1)}$ to maximize $Q(\mathbf{\Psi}, \mathbf{\Psi}^{(t)})$

Example 2.4.1 Radial Basis Function Network Hybrid EM Training

As mentioned in the previous section the RBF network is typically trained in two stages. In the first stage, the radial functions' parameters are determined using unsupervised training. Then the second stage, the weights are determined using a supervised technique. this is known as hybrid training

The first stage can be performed with the EM algorithm with the parameters initialized by the regression tree. The second stage is equivalent to solving a system of linear equations.

For the EM algorithm

1. Determine the number of terms or component densities m in the mixture model
2. Determine an initial guess at the component parameters: the mixing coefficients and (from the regression tree) the parameters for each radial basis function
3. For each data point $\mathbf{x}_i$, calculate the posterior probability for $i = 1, \ldots, n$ and $j = 1, \ldots, m$ using

$$\widehat{\tau}_{ij} = \frac{\widehat{p}_j H\left(\mathbf{x}_i : \widehat{\theta}_j\right)}{f(\mathbf{x}_i)} \tag{2.54}$$

where $\widehat{\tau}_{ij}$ represents the estimated posterior probability that point $\mathbf{x}_i$ belongs to the jth mixture, $H\left(\mathbf{x}_i : \widehat{\theta}_j\right)$ is radial basis function for the jth mixture evaluated at $\mathbf{x}_i$, and

$$\widehat{f}(\mathbf{x}_i) = \sum_{j=1}^{m} \widehat{p}_j H\left(\mathbf{x}_i : \widehat{\theta}_j\right) \tag{2.55}$$

is the RBF network estimate at point $\mathbf{x}_i$.

1. Update the component parameters (mean, covariance, mixing coefficients) for the individual components (using the Gaussian parameters as an example)

$$\widehat{p}_j = \frac{1}{n}\sum_{i=1}^{n}\widehat{\tau}_{ij} \tag{2.56}$$

$$\widehat{\mu}_j = \frac{1}{n}\sum_{i=1}^{n}\frac{\widehat{\tau}_{ij}\mathbf{x}_i}{\widehat{p}_j} \tag{2.57}$$

$$\widehat{\Sigma}_j = \frac{1}{n}\sum_{i=1}^{n}\frac{\widehat{\tau}_{ij}\left(\mathbf{x}_i - \widehat{\mu}_j\right)\left(\mathbf{x}_i - \widehat{\mu}_j\right)^T}{\widehat{p}_j} \tag{2.58}$$

2. Repeat steps 3 and 4 until the estimates converge

In the above hybrid training, the RBF network was transformed into a linear model. If the radial basis functions are allowed to change while determining the weights, then the model is nonlinear. For more information the reader is referred to (Desarbo and Cron 1988.) and (Desarbo and Weidel, 1995)

2.4.2 Kernel Trick Simulation

For this discussion, the following mixture model will be used

$$y = \sum_{i=1}^{3} w_i g(x; \mu_i, \sigma^2) + \epsilon \tag{2.59}$$

where $g(x; \mu_i, \sigma^2)$ denotes the normal probability function and $\epsilon = N(0, 0.05)$ where.

$$\mu_1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

$$\mu_2 = \begin{bmatrix} 3 \\ 5 \end{bmatrix}$$

$$\mu_3 = \begin{bmatrix} -1 \\ 0 \end{bmatrix}$$

$$\mathbf{\Sigma}_1 = \begin{bmatrix} 0.5 & 0.2 \\ 0.2 & 0.5 \end{bmatrix}$$

$$\mathbf{\Sigma}_2 = \begin{bmatrix} 1 & 0.4 \\ 0.4 & 1 \end{bmatrix}$$

$$\mathbf{\Sigma}_3 = \begin{bmatrix} 1 & 0.2 \\ 0.2 & 1 \end{bmatrix}$$

$$w_1 = 3$$

$$w_2 = 4$$

$$w_3 = 2$$

For each generation of the test data set, a sample size of $n = 120$ of bivariate data for the data space as shown in the Figure 2.4.

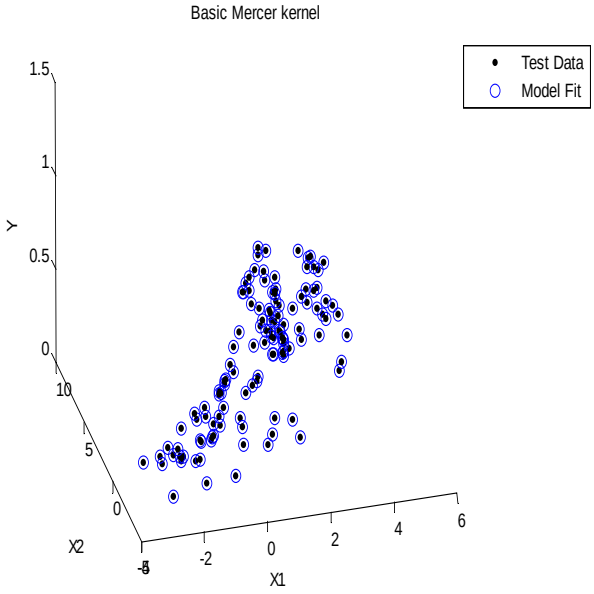

Figure 2.4: Kernel Trick Test Data

Using ICOMP, the numbers of clusters are determined and their information from the EM algorithm will be applied using the kernelized least squares. In Figure 2.5 are the histograms for the AIC and ICOMP scores for 50 generations.

From the EM algorithm, the estimated parameters are

$$\widehat{\mu}_1 = \begin{bmatrix} 0.91 \\ 1.99 \end{bmatrix}$$
$$\widehat{\mu}_2 = \begin{bmatrix} 3.08 \\ 4.96 \end{bmatrix}$$
$$\widehat{\mu}_3 = \begin{bmatrix} -1.2 \\ -.20 \end{bmatrix}$$

$$\widehat{\mathbf{\Sigma}}_1 = \begin{bmatrix} 0.76 & 0.35 \\ 0.35 & 0.74 \end{bmatrix}$$
$$\widehat{\mathbf{\Sigma}}_2 = \begin{bmatrix} 1.09 & 0.43 \\ 0.43 & 0.71 \end{bmatrix}$$
$$\widehat{\mathbf{\Sigma}}_3 = \begin{bmatrix} 0.86 & 0.2 \\ 0.2 & 1.16 \end{bmatrix}$$

Applying the transformation and solving for the linear weights, results in the following weights

$$\widehat{w}_1 = 4.41$$
$$\widehat{w}_2 = 3.87$$
$$\widehat{w}_3 = 1.86$$

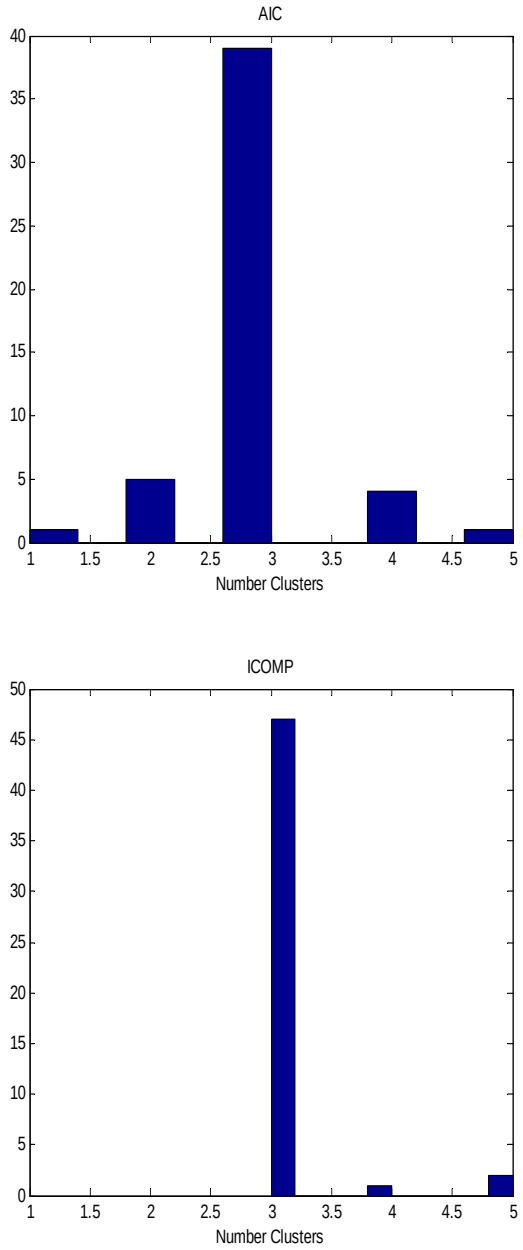

Figure 2.5: Kernel Trick Demonstration Fit Scores

Figure 2.6 shows the predicted and generalized fits of the kernelized RBF model using the values determined from the EM algorithm for each cluster parameters.

In this case, where we are working in the data space and projecting into the feature space in n dimensions parametrically, the model is fitting better from a mean square error than by working in the feature space. When comparing with the underlying true model to the fitted model, the reduced alpha model using the kernel trick has an average mean square error that is on average lower than the linear model using the transformed data. Refer to Figure 2.7.

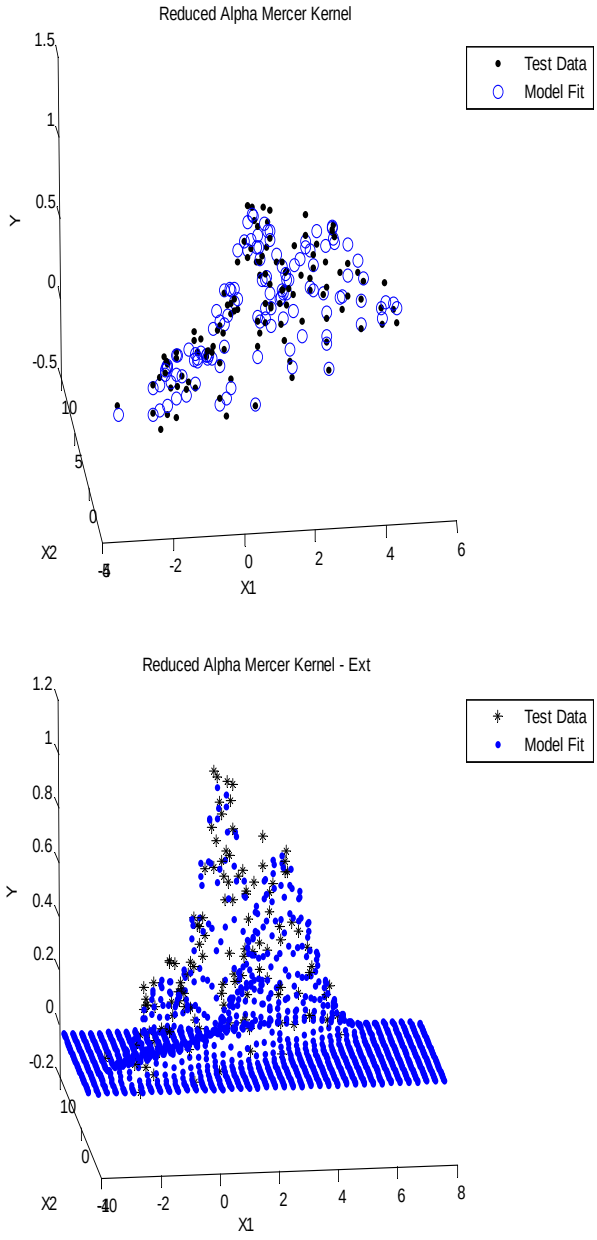

Figure 2.6: Kernel Trick Demonstration Model Fit

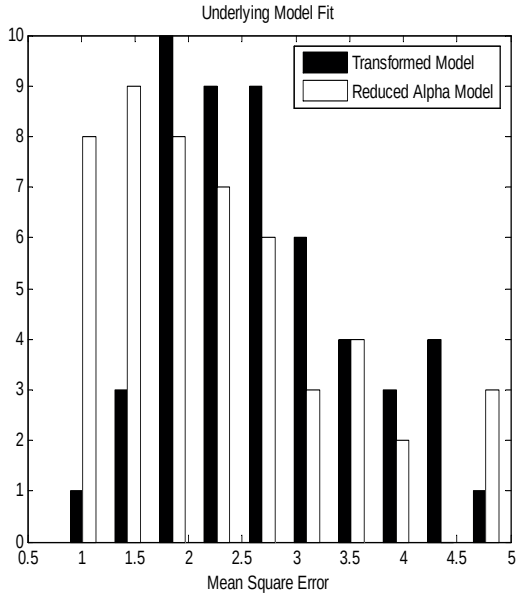

Figure 2.7: Kernel Trick Test Mean Square Error Values

Chapter 3

Decision Tree

Tree-structured classification and regression are nonparametric approaches to classification and regression that are not based on assumptions of normality and user-specified model statements. Yet, unlike the case for some other nonparametric methods, the resulting tree-structured predictors can be easy to use and relatively simple functions of the input variables.

Regression trees originated in the 1960s with the development of AID (Automatic Interaction Detection) by Morgan and Sonquist (1963). Morgan and Messenger (1973) then created THAID (Theta AID) to produce classification trees at the Institute for Social Research at the University of Michigan.

In the 1980s, statisticians Breiman et al. (1984) developed CART (Classification And Regression Trees). Since the original version, CART has been improved and given new features, and it is now produced, sold, and documented by Salford Systems. Statisticians have also developed other tree-based methods, and classification and regression trees can now be produced using many different software packages, some of which are relatively expensive and are marketed as being commercial data mining tools. Some software, such as S-Plus, use algorithms that are very similar to those underlying the CART program. CART will be the basis for the decision tree used in this dissertation.

This section describes the method involving decision trees to initialize the centers and radius for RBF networks. The combination of trees and RBF networks was first suggested by (M. Kubat and I. Ivanova,1995) in the context of classification rather than regression with further elaboration

of the idea appearing in (Kubat, 1998). The method has been reviewed and built upon by (Mark Orr, 1999)

3.1 Decision Tree Construction

The decision tree recursively partitions the input space into a two with each division parallel to one or more of the axes, dividing the input space into hyper-rectangles, also known as leaf nodes. A simple model or function is determined for each leaf such as a constant approximated by the sample average. The input space is ultimately organized into a binary tree with each branch expressed by an inequality involving one or more of the input components (e.g. $x_k \geq b$), where each dimension (k) and boundary (b) is selected so that the model error is minimized between model and data (Breiman, 1984). When using the decision tree as a starting point for a RBF network, each leaf node sets the initial parameters for each hidden unit of the RBF network, the center and radius being determined by the corresponding hyper-rectangle.

A flow chart illustrating a decision tree's growth can be found in Figure 3.1. Starting with the full data set, the algorithm first decides if a node needs to be split based on purity requirements. If a split is needed, the leaf node is now a branch node and a greedy search is employed to find a best split. Branching is based on a splitting rule, and standard splitting rules for regression and classification trees are given in the following subsections. Upon splitting two new leaf nodes are generated, and the process is repeated until a terminal node is reached. Once the tree is fully grown, a pruning step is employed to prevent over fitting.

The RBF network modeling developed in this thesis will build on Orr's variation on Kubat's idea with the following alterations.

- An external genetic algorithm, which sends potential data subsets to the decision tree, will allow variable subset selection. If the user wishes, the genetic algorithm will also allow a search over several radial basis functions and/or covariance smoothers.

- Pruning will not be done through a forward/backward selection method but through the use of information selection criteria.

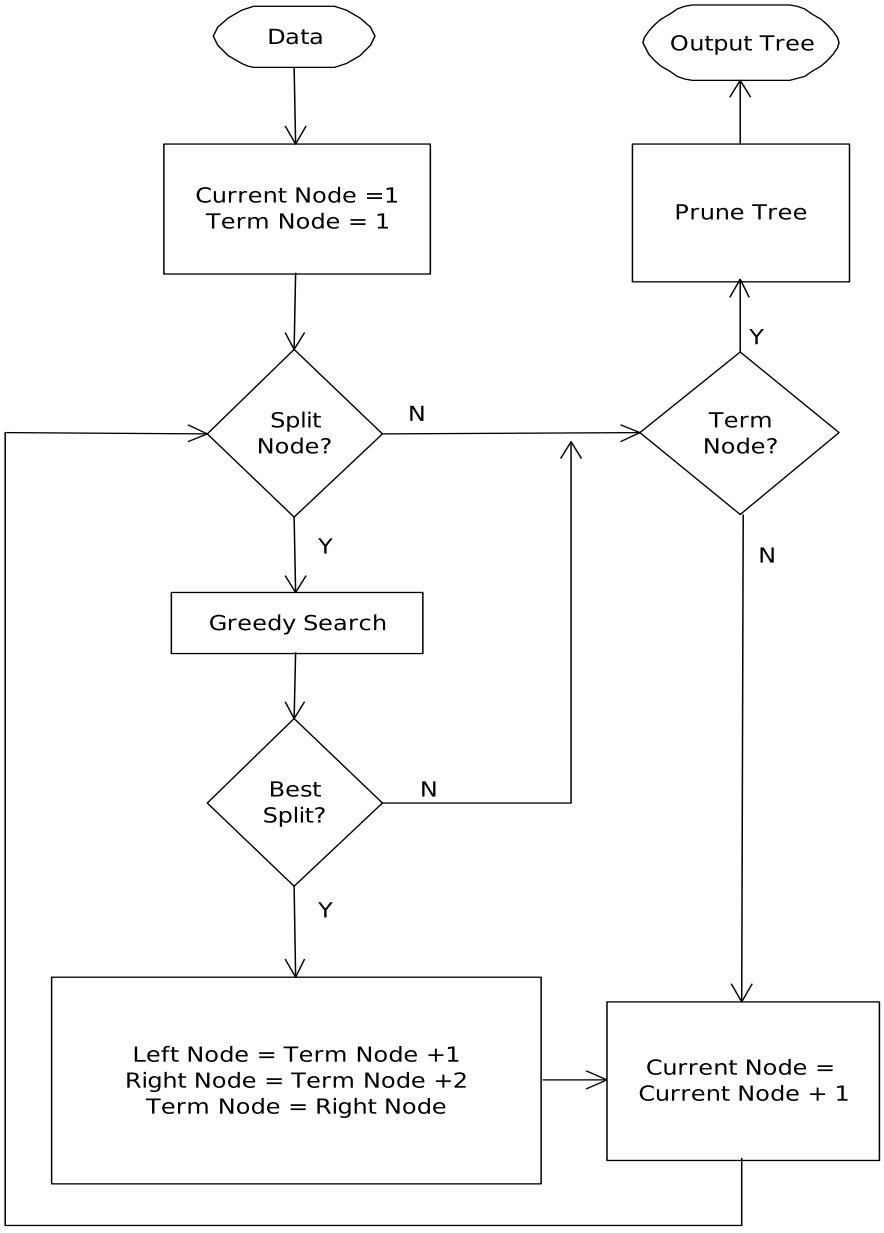

Figure 3.1: Decision Tree Flow Chart

- The decision tree allows for multiple variable splits with different separation operators (e.g. $x_k >= b$, $x_j <= c$). Due to the increased space of possible split locations, the greedy search will be replaced by a nested internal GA search engine. This will have the effect of eliminating the hyper-rectangles but will allow partitioning more consistent with the data. Possible over-fitting is controlled by an information complexity fitness function.

- The kernel trick using Mercer's condition is applied through the splitting rule as scored by information criteria. This allows the algorithm step that allows the use of projected feature space information while remaining in the current data space, never explicitly computing the data transformations.

3.1.1 Classification Tree Growth

In classification where the target outcome belongs to one of G unordered classes $G \in \{1, \ldots, C\}$. the observations in node d are classified so as to minimize the expected cost:

$$y_d = \arg\max_k \sum_g \Pr(g \mid d)\mathbf{C}(k, g) \tag{3.1}$$

where $\mathbf{C}(k, g)$is the cost of predicting that d belongs to class k when it belongs to class g.

The probability that a an observation is in class g given that it is node d is calculated by

$$\Pr(g \mid d) = \frac{\Pr(g, d)}{\Pr(d)} \tag{3.2}$$

where $\Pr(g, d)$ is the joint probability that an observation will be node d and class g. It is calculated as

$$\Pr(g, d) = \pi_g \frac{n_g(d)}{n_g} \tag{3.3}$$

where n_g is the number of observations that belong to class g, $n_g(d)$ is the number of observations at node d that belong to class g, and π_g is the prior probability that an observation belongs to class

g. When unknown the prior can be estimated from the data as

$$\pi_g = \frac{n_g}{n}. \tag{3.4}$$

$\Pr(d)$ is the probability that a an observation is in node d and is calculated by

$$\Pr(d) = \sum_{g=1}^{C} \Pr(g, d). \tag{3.5}$$

In determining splitting rule, different measures of node impurity Q_g can be implemented:

- Misclassification error:

$$Q_d = \frac{1}{n} \sum_{i \in D} (y_i \neq d) = 1 - \Pr(g \mid d) \tag{3.6}$$

- Gini Index:

$$Q_d = \sum_{g=1}^{C} \Pr(g \mid d)(1 - \Pr(g \mid d)) \tag{3.7}$$

- Cross-entropy:

$$Q_d = \sum_{g=1}^{C} \Pr(g \mid d) \log(\Pr(g \mid d)) \tag{3.8}$$

- or deviance:

$$Q_d = -2 \sum_{g=1}^{C} n_g \log(\Pr(g \mid d)). \tag{3.9}$$

The split chosen is the one that yields the largest decrease in impurity

$$\Delta Q = Q_d - p_R Q_R - p_L Q_L, \tag{3.10}$$

which is given by the split that maximizes $(p_R Q_R + p_L Q_L)$ with p_R and p_L equal to the proportion of data that are sent to the left and right child nodes from the split.

In this thesis, the deviance will be used as the measure for node impurity since it is related to the multinomial likelihood. The root nodes' children are split recursively by the same process until

a split will create a child containing less samples than a given maximum or when the terminal node is pure. An example of a classification tree using the Iris data is given in Figure 3.2.

3.1.2 Regression Tree Growth

In the construction of a regression tree, a division of the root node splits the training samples into left and right subsets, S_L and S_R. The mean output value on either side of the split is

$$\overline{y}_L = \frac{1}{n_L} \sum_{i \in S_L} y_i \tag{3.11}$$

$$\overline{y}_R = \frac{1}{n_R} \sum_{i \in S_R} y_i \tag{3.12}$$

where n_L and n_R are the number of samples in each subset. The mean square error between model and data is then

$$E(split) = \frac{1}{n}\{\sum_{i \in S_L} (y_i - \overline{y}_L)^2 + \sum_{i \in S_R} (y_i - \overline{y}_R)^2\}, \tag{3.13}$$

which corresponds to a node impurity of

$$Q_d = \frac{1}{n} \sum_{i \in D} (y_i - \overline{y})^2. \tag{3.14}$$

The division which minimizes $E(split)$ over all possible split choices is used to create the children of the root node and is typically found by discrete greedy search over p dimensions and n observations. The root nodes' children are split recursively by the same process until a split will create a child containing less samples than a given minimum, $n_{\min}$. Refer to Figure 3.3 for an example of a regression tree.

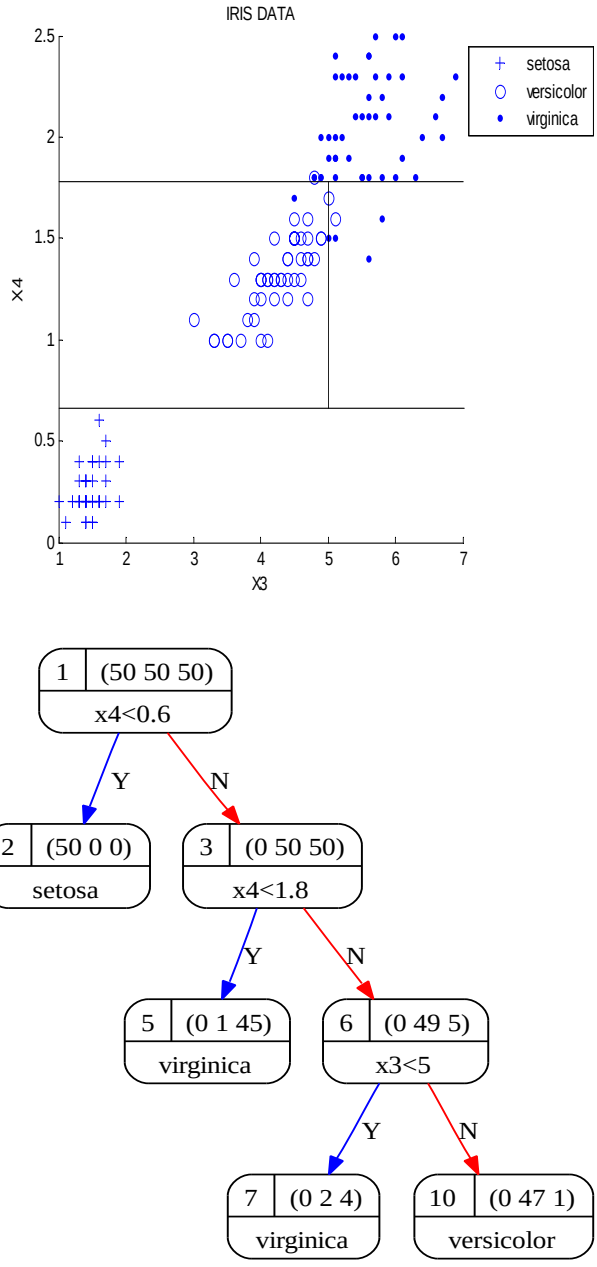

Figure 3.2: Nominal classification Tree using Iris Data

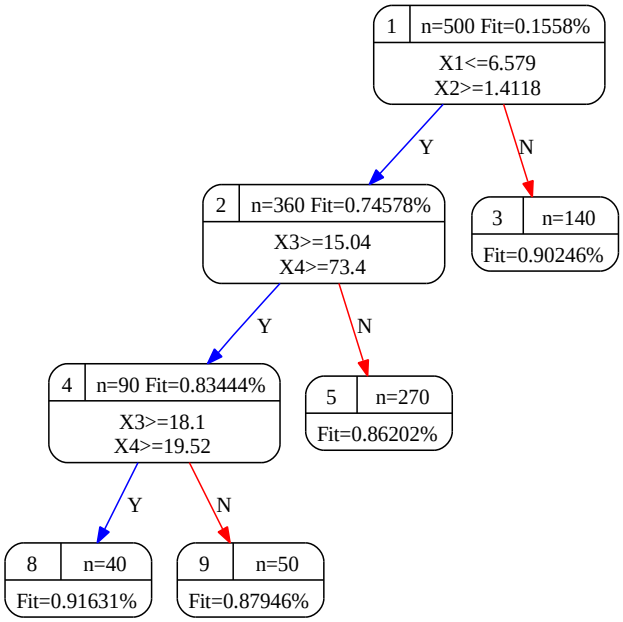

Figure 3.3: Regression Tree

3.1.3 Decision Tree Pruning

The tree is typically grown until some predetermined purity or minimum node size is reached. Pruning of the tree is through cost-complexity pruning with the cost complexity given by

$$C_\alpha(T) = \sum_{d=1}^{D} N_d Q_d + \alpha D \tag{3.15}$$

where D equals the number of terminal nodes in tree T.

Through tenfold cross validation, the value of $\widehat{\alpha} \geq 0$ is chosen to find the subtree $T_\alpha \subseteq T_o$ that will minimize $C_\alpha(T)$ The tuning parameter α governs the tradeoff between tree size and its goodness of fit to the data. See Breiman et al

Here it is proposed that the technique of Bozdogan's (1990) new information of complexity (ICOMP) criterion be implemented to grow and prune the decision tree.

3.2 Kernelization of the Decision Tree

One of the advantages of using the decision tree, is its ability to divide the data space into potential hyper areas for radial basis functions. The tree contains a root node, some nonterminal nodes (having children) and some terminal nodes (having no children). Each node is associated with a hyper-rectangle of input space having a center and size as described above. The node corresponding to the largest hyper-rectangle is the root node and the node sizes decrease down the tree as they are divided into smaller and smaller pieces.

While the hyperrectangle defined by the subset of samples in a terminal node can help to define the radial basis parameters, there are problems:

1. The data's placement within the tree node may truly not be centered and dispersed as depicted by the hyperrectangle.

2. Since the decision tree in this thesis will allow for multi-variable splits, the node regions are no longer a true hyperrectangle. This increases the difficulty of using the node center, radius, and/or borders as parameters for the radial basis functions.

3. The program uses parameters estimated from the data within a tree node. Yet, using the data's covariance to estimate the true smoothing parameter is often difficult due to an ill-conditioned singular matrix which will often degenerate as the data dimension p increases with decreasing node sample size as the tree grows.

4. With decreasing node sample size, the data position tends towards the boundary of the hyperrectangle.

However, the decision boundary modeling often gives more accurate results than the probability approach of the radial basis functions. Often this is the case, since the decision tree is non parametric with no assumptions about the form of the probability distribution and is free to dynamically adapt to the complexity of the data. Through combining the hard membership of the decision boundary with the soft membership of the probability distribution, a significant issue of the decision tree is addressed. Each node's corresponding area assigns the same probability estimates to all points in the region or the same expected value for the classification and regression trees respectively.

Determining the best available splits and corresponding node radial basis functions will be performed through using the kernelized least squares prediction at each leaf node, a model tree. Details of using the kernelized ridge least squares in the context of regression and classification are given in Chapters 4 through 6. Following is a discussion on the modifications needed in building and pruning the decision tree.

3.2.1 Genetic Search Algorithm

While the most comprehensive and ideal way to find the model with the lowest model selection criterion value would be to compute and evaluate all possible model combinations, this method would require intensive computation. Because of its adaptive nature, GA has been widely used in many different areas to solve complex problems through handful of simple constructs. A form a generate-and-test paradigm, the GA "breeds" a solution using techniques that simulate the processes of natural evolution. The genetic algorithm starts with a large population of potential solutions and through the application of crossover and mutation evolves a solution over time that is more

optimal than previous solutions. The field of genetic and evolutionary algorithms was introduced by John H. Holland (1975). Goldberg (1989 - 2002) and many others have popularized the genetic algorithm.

In the model selection framework, the genetic algorithm has the following advantages and disadvantages.

Advantages:

- The ability to solve nonlinear, noisy, and discontinuous problems.
- The ability to solve complex optimization problems
- It can handle data sets of virtually any size.
- There are no specific requirements on the fitness function. The function does not need to be monotone, continuous or differentiable.
- It can return several good competing models.

Disadvantages:

- Complete dependence on the fitness function and even then the GA is not guaranteed to find the optimal solution, but only a good solution.
- Sensitivity to genetic algorithm parameters: *choosing the optimal combination of parameters still remains a question to be solved.*
- Sensitivity to chromosome genome encoding

There are two genetic algorithms (GA) used in the program developed in this thesis. First, an external GA controls kernel function, covariance smoother, and variable subset selection. Second, because of the added complexity of the search space, an internal nested GA replaces the traditional greedy search algorithm of the decision tree. This will allow the use of an informational complexity criterion to prevent over fitting and penalize non-parsimonious behavior while partitioning the

data space into potential radial basis functions based on a subset selection of candidate variables. The genetic code used in the kernelized decision tree is a modified version of Genetic Algorithm Optimization Toolbox known as GAOT version 2 written by C.R. Houck, J.A. Joines, and M.G. Kay (Houck, et al, 1998).

Following is the final program algorithm structure:

Proposition 3.2.1 Program Algorithm

1. Determine if Y is continuous, nominal, or ordinal
2. Determine if RBF is to be solved for specific kernel function and covariance estimator or should the program search over all kernel functions and estimators
3. Send to Outer Genetic Algorithm for X variable subset selection.
4. Subset is sent to Decision Tree to divide data space into potential RBF areas via the kernel trick
    - The program iterates then through three levels of an inner nested genetic algorithm
    - The first genetic algorithm chooses the variables on which to divide the tree nodes.
    - The second genetic algorithm determines the direction of the operator (<= or >=) for each variable. (Categorical variables are divided on equal or not equal.)
    - The third genetic algorithm determines the cutoff value of the separation. So one will get something like (X1 <=48 and X2 >=400)
    - To prevent over fitting from using too many variables, information complexity is used to penalize non-parsimonious behavior.
5. Based on Y's data type, the decision tree will find an optimal regression, categorical, or ordinal splits.
6. The model fit is determined through informational complexity and the value is sent back to Step 4

7. Pruning of the Tree and the resulting RBF structure will be by information criteria scoring. Generalized cross validation is not used due to the labor and time intensive method of the kernel trick. Pruning is required since the decision tree's growth is inherently instable by its top-down induction This is true either in a greedy search's tendency to over fit or the by the non guarantee of the genetic algorithm to find a true optimal, but a local optimal value.

External Genetic Algorithm

The external GA flow chart is shown in Figure 3.4. This diagram also illustrates the basic structure of a genetic algorithm. The GA works by first creating in Step 2 an initial population of P possible solutions in the form of candidate chromosomes representing individual solutions. Each chromosome is an encoding of the individual solution's input parameters that are used to initialize model's analysis and measure the outcome against a fitness function which measures the individual's goodness of fit. Here, each chromosome is sent to the decision tree for modeling and given a goodness of fit score after pruning, Step 5. Based on predicted variable's data type, the decision tree will find optimal regression, categorical, or ordinal splits. The better the goodness of fit the closer the individual is to an optimal solution.

After all individuals in a population have been evaluated, a terminating decision is performed in Step 6 . Several common termination conditions which may be use in parallel are maximum number of generations, a maximum elapse time, the average/best fitness function value reaches steady state over successive generations, the average/best fitness function value oscillates over generations, and the average fitness reaches a very early steady state or begins to decay The GAs in this dissertation terminate based on a maximum number of generations or when the difference in the best and worst fitness function in a generation reaches a value of less than one.

If the terminating condition is not met, a new population is created, Step 7, by saving a percentage of the top K chromosomes . The remaining $P-K$ chromosomes are replaced with chromosomes created by merging the parameters of the top K chromosomes. Of the fittest K chromosomes, a

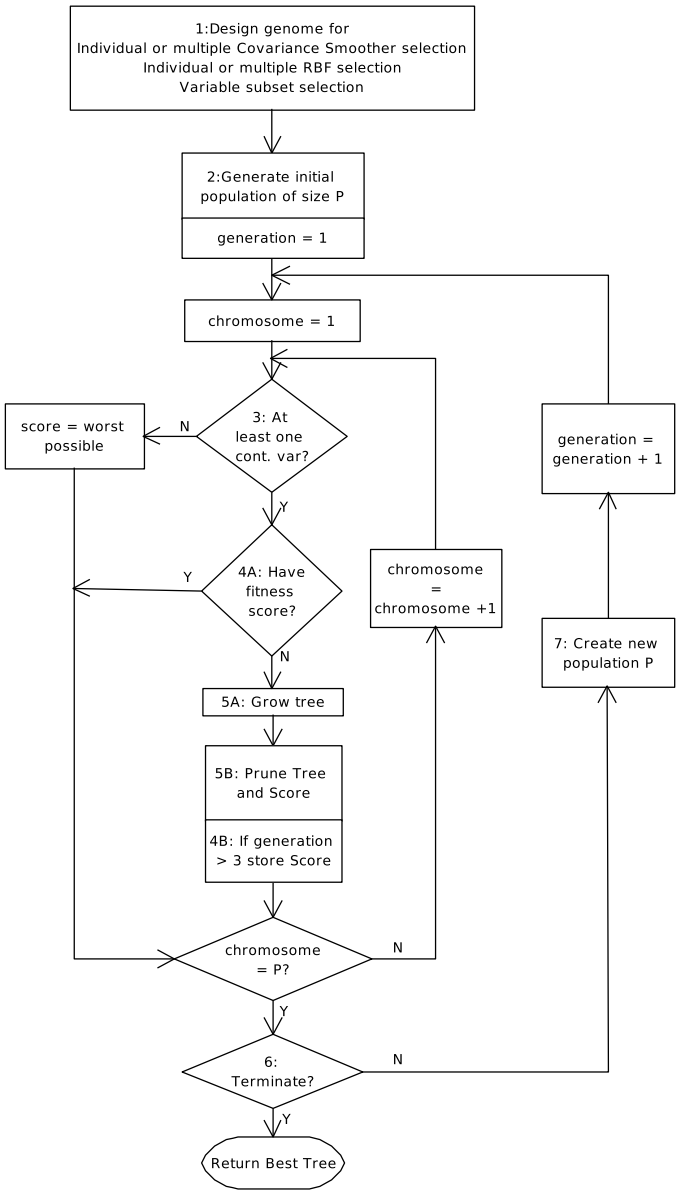

Figure 3.4: External GA Algorithm

percentage of these will become parents and generate one or more children in the new population. To encourage genetic diversity, new chromosomes are also created by randomly mutating the parameters in a few of the chromosomes in the new population.

Determining the size of the initial population in Step 2 is an important processing parameter. With initial population that is too small, a GA can take a long time to find an optimal solution or trap itself in a local minimum or maximum. With a relatively large population, the GA can require a large number of generations to filter out high-performing chromosomes from the large number of lower-performing chromosomes. Currently, the population size is set so that each chromosome bit has an 80% chance of being in the initial population.

A different selection process may and is often used for selecting the set of fittest individuals, selecting parents for crossover, or selecting chromosome mutation. Strategies for selection include elitist, proportional fitness, ranking, random, and tournament. For this dissertation, ranking based on the geometric distribution is the selected method.

The process of mating and reproduction in a GA to create future potential solutions is called crossover. This name describes the way in which future chromosomes are generated using the genetic material from two parents which are extracted and appended. A low mutation uniform crossover, based on a mixing rate where individual gene positions are picked from the parent chromosome and exchanged, is the method employed in the external and following internal genetic algorithms.

If the GA maintains a steady population size, the two new children will tend to replace poorer-performing genomes. However, not every crossover will nor is intended to improve the fitness of the population. Crossover or breeding also has the goal of increasing or maintaining genetic diversity in the population. It is through this diversity that a GA economically and effectively explores the solution space.

In some cases, crossover breeding is insufficient to explore the underlying solution space, and the population becomes confined to a small region of the solution space. Mutation randomly changes the value of a genome locus to produce individuals that move outside this region. However, with too much mutation the genome loses its ability to retain any pattern and prevents the GA from converging to a solution.

In order to speed the GA search, Step 4 reduces the amount of redundant tree builds while allowing a few random builds in the beginning. Since the model is a RBF network, Step 3 is a filter that maintains at least one variable in the subset selection must be continuous variable.

Inner Genetic Algorithm

Each tree node specifies a condition of some attribute(s) of the data. When the data is parsed through the tree, the left child of a node is chosen if the condition(s) in the node is true for that instance, else the right child is chosen. To allow partitioning more consistent with the data structure, the nested GA presented in Figure 3.5 is used in place of the decision tree's traditional greedy search as shown in Figure 3.1.

The Split Variable GA selects which variables from the will be used to split the node. Here it is important to note an important difference in variable subset selection between the external and internal genetic algorithms. One external variable subset with (X1, X2, and X3) and another with (X1 and X2) both may only need X1 and X2 as significant variables to divide the data space X in relation to Y. The value of X3 (here a continuous variable) in relation to the radial basis function fit cannot be determined until the final RBF model is scored.

Each chromosome combination from the Split Variable population S is then sent to the Operator Selection GA. The Operator Selection GA then generates a population O of possible operator conditions for the selected split variables. This is repeated as each split variable and operator selection chromosome is sent to the Cut Value GA which generates a population of possible (if any) cut values.

The chromosomes have been encoded as bit strings where the genome has a value of 0 or 1. This representation is chosen for its significant effect on ease of programming, processing speed, convergence, and amenability to crossover and mutation genetic operators. Each variable selection is encoded as a string, where 0 or 1the absence or presence of a given predictor variable. For example, 1101 contains the predictor variables 1, 2, and 4. If a variable has been selected, a second chromosome would indicate the type of split to be performed. A 0 or 1 would indicate a <= or >=split respectively for a continuous variable. For a categorical variable, a 0 or 1would indicate

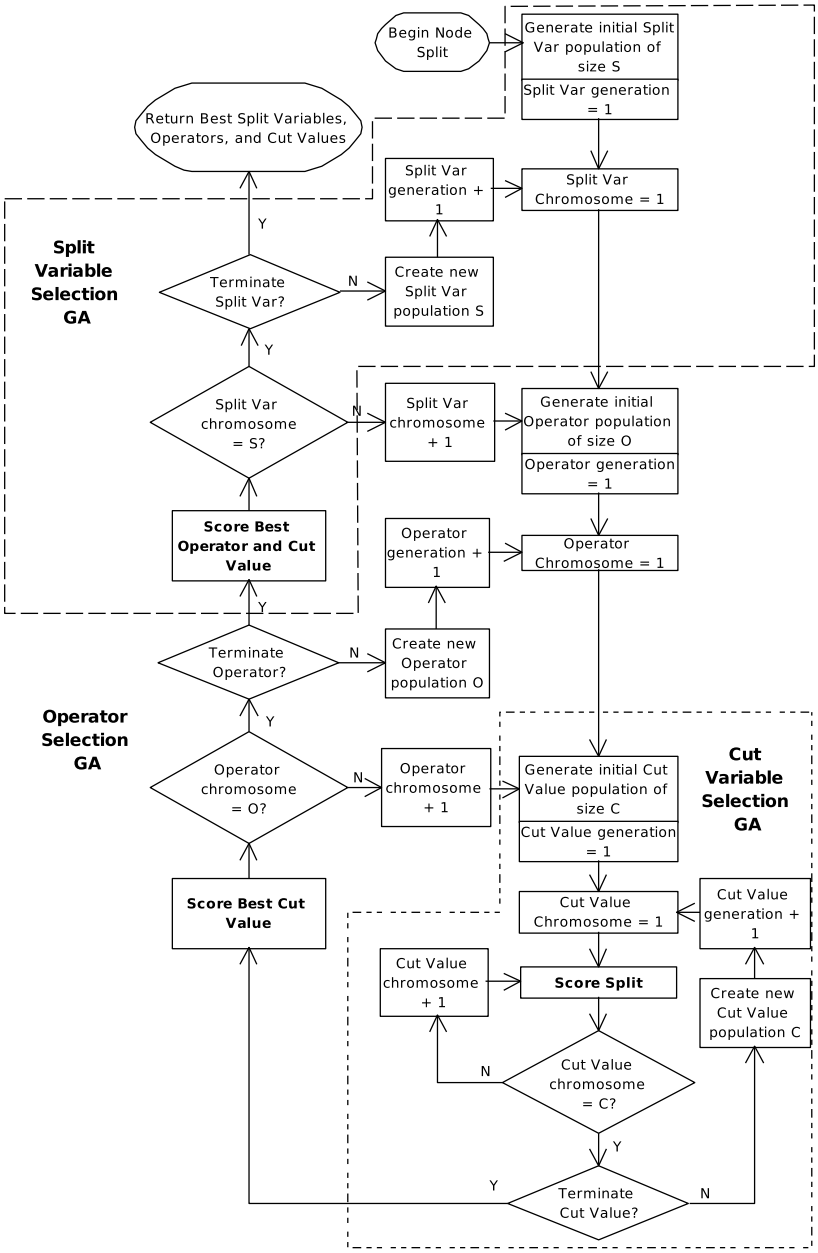

Figure 3.5: Internal GA Algorithm

equal or not equal to a certain value. The final or third chromosome, is a set of integers which reference a indicator matrix that references a value for an existing split location.

Each split variable, operator selection, and cut value combination is then scored with an informational complexity goodness of fit using the appropriate kernel trick method. The health of the combination is cycled back up through the nested GA as each stage reaches an optimal solution.

3.2.2 Information Complexity

Evaluating a chromosome associates a measure of goodness-of-fit or performance to each chromosome in the population. The chromosomes are processed by the GA algorithm which generates an actual outcome or model whose degree of performance is measured and ranked by a fitness function. The fitness function that determines the goodness of fit or health of the individual chromosome will be determined through information criteria for the best split of the mapping function in the decision tree and subsequent tree pruning. The information complexity criterion used will be that developed by Bozdogan (1988, 1990, 1994, 2000, 2004).

Bozdogan's (1990) new information of complexity (ICOMP) criterion is defined as

$$ICOMP = -2\log L(\widehat{\theta}) + 2C_1(\widehat{\mathcal{F}}^{-1}(\widehat{\theta})) \tag{3.16}$$

where $L(\widehat{\theta})$ is the likelihood of the data and $C_1(\widehat{\mathcal{F}}^{-1})$ is a maximal information theoretic measure of complexity of $\widehat{\mathcal{F}}^{-1}$. defined by

$$C_1(\widehat{\mathcal{F}}^{-1}) = \frac{q}{2}\log\left[\frac{tr(\widehat{\mathcal{F}}^{-1}(\widehat{\theta}))}{q}\right] - \frac{1}{2}\log|\widehat{\mathcal{F}}^{-1}(\widehat{\theta})|. \tag{3.17}$$

ICOMP can be divided into two components: Lack of Fit given by $-2\log L(\widehat{\theta})$and Lack of Parsimony given by$_1(\widehat{\mathcal{F}}^{-1})$.

Lack of Fit (Regression Tree)

In Chapter 2 it was shown that least squares for the additive error model , with $\epsilon \sim N(0, \sigma^2)$, is equivalent to maximum likelihood using the conditional likelihood

$$\Pr(\mathbf{y} \mid \mathbf{X}) = \mathbf{N}(f(\mathbf{X}), \sigma^2). \tag{3.18}$$

The log-likelihood of the data is then

$$\begin{aligned} l(\theta) &= \sum_{i=1}^{n} \log \Pr_{\theta}(y_i) \\ l(\theta) &= -\frac{n}{2}\log(2\pi) - \frac{n}{2}\log\sigma^2 - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(\mathbf{y} - f(\mathbf{X}))^2. \end{aligned} \tag{3.19}$$

With the structure of the decision tree, there will an individual kernel model fit at each node. This results in a mixture model so that the conditional likelihood is

$$\Pr(\mathbf{y} \mid \mathbf{X}) = \Pr(\mathbf{d}) * \mathbf{N}(f_d(\mathbf{X_d}), \sigma_d^2). \tag{3.20}$$

The probability of the node $\Pr(d)$can be estimated by

$$\Pr(d) = \frac{n_d}{n}, \tag{3.21}$$

and the log-likelihood of the data is

$$l(\theta) = -\frac{n}{2}\log(2\pi) - \frac{1}{2}\sum_{d=1}^{D} n_d \log\sigma_d^2 - \sum_{d=1}^{D}\frac{1}{2\sigma_d^2}\sum_{i=1}^{n_d}(\mathbf{y}_{di} - f(\mathbf{X}_{di}))^2. \tag{3.22}$$

During the pruning or fitting of the radial basis model, the MLE of $\sigma_d^2$is defined as

$$\widehat{\sigma}_d^2 = \frac{1}{n}\sum_{i=1}^{n_d}(\mathbf{y}_{di} - f(\mathbf{X}_{di}))^2, \tag{3.23}$$

and the simplified log-likelihood becomes

$$l(\theta) = -\frac{n}{2}\log(2\pi) - \frac{1}{2}\sum_{d=1}^{D} n_d \log \sigma_d^2 - \frac{n}{2}. \tag{3.24}$$

However, as shown in Chapter 4 when splitting the nodes, the model $f(\mathbf{X}_{di})$ is an additive function such that

$$\sigma_d^2 = \sigma_1^2 + \sigma_2^2 \tag{3.25}$$

preventing the above simplification of the log-likelihood.

Lack of Fit (Classification Tree)

When applying the kernelized least squares, the probability an observation will be assigned to specific class will be determined by the softmax function

$$\Pr(g \mid f(\mathbf{X})) = \frac{\exp(f(\mathbf{X}))}{\sum \exp(f(\mathbf{X}))}, \tag{3.26}$$

which is the multinomial response model. The deviance of a node is then given by

$$\begin{aligned} Q_d = \quad & -2 * \textstyle\sum_{g=1}^{C} \left(n_g \log(\Pr(g \mid f(\mathbf{X}))) + (n - n_g)\log(1 - \Pr(g \mid f(\mathbf{X})))\right) \\ & -2 * \sum_{g=1}^{C} n_g \log(\Pr(g \mid d)). \end{aligned} \tag{3.27}$$

Lack of Parsimony

In the context of the regression tree, it is assumed that for each node

$$\Pr(\mathbf{y} \mid \mathbf{X}) = \Pr(\mathbf{d}) * \mathbf{N}(f_d(\mathbf{X_d}), \sigma_d^2). \tag{3.28}$$

Similarly, with the classification tree, for each class cluster in the feature space

$$\Phi(\mathbf{x})_{gi} \sim N(\mu_g, \sigma_g^2) \text{ for } g = 1, 2, \ldots, C;\ i = 1, \ldots, n_k. \tag{3.29}$$

Using the fact that the parameters for a particular cluster and/or node are considered independent from each other, the inverse-Fisher information matrix $\mathcal{F}$ is defined (Bozdogan, 1990) as

$$\mathcal{F}^{-1} = \begin{bmatrix} \mathcal{F}_1^{-1} & 0 & \cdots & 0 \\ 0 & \mathcal{F}_2^{-1} & & \\ \vdots & & \ddots & \\ 0 & & & \mathcal{F}_K^{-1} \end{bmatrix} \tag{3.30}$$

so that

$$\mathcal{F}^{-1} = Diag\left(\begin{bmatrix} \frac{\sigma_1^2}{n_1} & 0 \\ 0 & \frac{2\sigma_1^4}{n_1} \end{bmatrix} \quad \begin{bmatrix} \frac{\sigma_K^2}{n_K} & 0 \\ 0 & \frac{2\sigma_K^4}{n_K} \end{bmatrix} \right). \tag{3.31}$$

Therefore

$$C_1(\widehat{\mathcal{F}}^{-1}) = \frac{\dim \widehat{\mathcal{F}}^{-1}}{2} \log \left[\frac{tr(\widehat{\mathcal{F}}^{-1}(\widehat{\theta}))}{q} \right] - \frac{1}{2} \log |\widehat{\mathcal{F}}^{-1}(\widehat{\theta})|. \tag{3.32}$$

$$C_1(\widehat{\mathcal{F}}^{-1}) = \frac{2K}{2} \log \left[\frac{\sum_{k=1}^{K} \left(\frac{\widehat{\sigma}_k^2 + 2\widehat{\sigma}_k^4}{n_k} \right)}{2K} \right] - \frac{1}{2} \log \prod_{k=1}^{K} \left(\frac{\widehat{\sigma}_k^2}{n_k} \cdot \frac{2\widehat{\sigma}_k^4}{n_k} \right)$$

$$C_1(\widehat{\mathcal{F}}^{-1}) = K \log \left[\frac{\sum_{k=1}^{K} \left(\frac{\widehat{\sigma}_k^2 + 2\widehat{\sigma}_k^4}{n_k} \right)}{2K} \right] - \frac{1}{2} \log \prod_{k=1}^{K} \left(\frac{2\widehat{\sigma}_k^6}{n_k} \right)$$

where $\widehat{\sigma}_k^2$ is given by $\widehat{\sigma}_d^2$ in the regression tree and $\widehat{\sigma}_g^2$ in the classification tree.

Note that what we would like to estimate $\widehat{\sigma}_g^2$ in the feature space (Shawe-Taylor, 2005) where we have

$$\widehat{\mathbf{s}}_g^2 = \frac{1}{n_g}\sum_{i=1}^{n_k}\left(\Phi\left(\mathbf{x}_{gi}\right) - \Phi\left(\overline{\mathbf{x}}_g\right)\right)^2 \tag{3.33}$$
$$\widehat{\mathbf{s}}_g^2 = \frac{1}{n_g}\sum_{i=1}^{n_k}\left(\Phi\left(\mathbf{x}_{gi}\right) - \Phi\left(\overline{\mathbf{x}}_g\right)\right)^T\left(\Phi\left(\mathbf{x}_{gi}\right) - \Phi\left(\overline{\mathbf{x}}_g\right)\right),$$

and $\Phi\left(\overline{\mathbf{x}}_g\right)$ is the cluster center in the feature space.

Then with the kernel trick

$$\left\|\Phi(\mathbf{x}_i) - \Phi\left(\overline{\mathbf{x}}_g\right)\right\|^2 = \left(\Phi(\mathbf{x}_i) - \Phi\left(\overline{\mathbf{x}}_g\right)\right)^T\left(\Phi(\mathbf{x}_i) - \Phi\left(\overline{\mathbf{x}}_g\right)\right) \tag{3.34}$$
$$\begin{aligned}\left\|\Phi(\mathbf{x}_i) - \Phi\left(\overline{\mathbf{x}}_k\right)\right\|^2 &= \Phi(\mathbf{x}_i)^T\Phi(\mathbf{x}_i) - \Phi\left(\overline{\mathbf{x}}_g\right)^T\Phi\left(\overline{\mathbf{x}}_g\right) \\ &\quad - \Phi(\mathbf{x}_i)^T\Phi\left(\overline{\mathbf{x}}_g\right) + \Phi\left(\overline{\mathbf{x}}_g\right)^T\Phi(\mathbf{x}_i)\end{aligned} \tag{3.35}$$
$$\left\|\Phi(\mathbf{x}_i) - \Phi\left(\overline{\mathbf{x}}_g\right)\right\|^2 = K(\mathbf{x}_i, \mathbf{x}_i) + K(\mathbf{x}_g, \mathbf{x}_g) - 2K(\mathbf{x}_i, \overline{\mathbf{x}}_g)$$

and noting that for a Gaussian RBF, $K(\mathbf{x}_i, \mathbf{x}_i) = 1$, we have

$$\widehat{\mathbf{s}}_g^2 = \frac{2}{n_g}\sum_{i=1}^{n_g}(1 - K(\mathbf{x}_i, \overline{\mathbf{x}}_g)). \tag{3.36}$$

3.2.3 Covariance Parameter Estimation

Parameter optimization will then be dependent on the selection of a suitable kernel function and the kernel parameters in relation to the decision tree structure. Examples of kernel functions are

Gaussian	$\exp[-\frac{\|\|\mathbf{x}_i-\mathbf{x}_j\|\|^2}{h}]$
Power Exponential (PE)	$\exp[-(\frac{\|\|\mathbf{x}_i-\mathbf{x}_j\|\|^2}{h^2})^{\beta}]$
Polynomial	$((x_i.x_j)+h)^d$
Sigmoidal	$\tanh[a(\mathbf{x}_i.\mathbf{x}_j)+b]$
Cauchy	$\frac{1}{1+\frac{\|\|\mathbf{x}_i-\mathbf{x}_j\|\|^2}{h}}$.

Methods to estimate the covariance (smoothing) matrix Σ_{node} that address the issue of ill conditioning both due to the reduction in data points as the tree grows and the selection of a small set for a possible split include smoothed, robust, or stoyki covariance estimators. Included in the dissertation program are the following:

- The *maximum likelihood* (MLE) covariance matrix:

$$\hat{\Sigma}_{MLE} = \frac{1}{n}\mathbf{X^T}\left[\mathbf{I} - \frac{1}{n}11^T\right]\mathbf{X} \tag{3.37}$$

  where 1 is a column vector of ones.

- The covariance matrix divided by n instead of $n-1$ (None)

- The *maximum entropy* (ME) covariance matrix:

$$\hat{\Sigma}_{ME} = C + D \tag{3.38}$$

  where C is the covariance matrix of the secondary midpoints and D is a diagonal matrix with positive elements. When $n \leq p$, $\hat{\Sigma}_{ME}$ has the advantage of not degenerating. For information, refer to (Fiebig ,1982) and (Thiel, 1984).

- The *maximum likelihood empirical Bayes* estimator (MLEEB) covariance matrix:

$$\hat{\Sigma}_{MLEEB} = \hat{\Sigma}_{MLE} + \frac{p-1}{n * trace(\hat{\Sigma}_{MLE})}\mathbf{I}_{p\times p} \tag{3.39}$$

- The *maximum entropy empirical Bayes* estimator (MEEB) covariance matrix:

$$\hat{\Sigma}_{MEEB} = \hat{\Sigma}_{ME} + \frac{p-1}{n * trace(\hat{\Sigma}_{ME})}\mathbf{I}_{p\times p} \tag{3.40}$$

- The *stipulated diagonal covariance estimator (SDCE)*:

$$\hat{\Sigma}_{SDE} = (1-\pi)\widehat{\Sigma}_{node} + \pi Diag(\widehat{\Sigma}_{node}), \tag{3.41}$$

where $\pi = p(p-1)\left[2n\left(trR^{-1} - p\right)\right]^{-1}$ and where

$$R = Diag^{-1/2}(\widehat{\Sigma}_{node})\widehat{\Sigma}_{node}Diag^{-1/2}(\widehat{\Sigma}_{node}) \tag{3.42}$$

is the correlation matrix.

The SRE and SDE estimators are due to Shurygin (1983). SDE avoids scale dependence of the units of measurement of the variables.

- The *convex sum covariance estimator (CSCE)*:

$$\hat{\Sigma}_{CSE} = \frac{n}{n+m}\widehat{\Sigma}_{node} + (1 - \frac{n}{n+m})\hat{D}_W, \tag{3.43}$$

Based on the quadratic loss function used by Press (1975), Chen (1976) proposed a *convex sum covariance matrix estimator (CSE)* given by where $\hat{D}_W = (\frac{1}{p}tr\widehat{\Sigma}_{node})\mathbf{I}_p$.

For $p \geq 2$, m is chosen to be

$$0 < m < \frac{2[p(1+\beta)-2]}{p-\beta}, \tag{3.44}$$

where

$$\beta = \frac{(tr\widehat{\Sigma}_{node})^2}{tr(\widehat{\Sigma}^2_{node})}. \tag{3.45}$$

This estimator improves upon the $\widehat{\Sigma}_{node}$ by shrinking all the estimated eigenvalues of $\widehat{\Sigma}_{node}$ toward their common mean. Note that there are other smoothed covariance estimators. For space considerations, we will not discuss them in this paper. For more on these, see Bozdogan (2005).

Chapter 4

Genetic Kernelized Regression Tree

The aim of this section is to develop the regression tree method that provides a linking structure between the data and feature space. Through the use of the nonparametric division of the data space by the decision tree and the selection controlled by information criteria, a method will be developed to determine an optimal radial basis function model. Presented is the idea of using a Gaussian process model as a framework for the regression tree.

A Gaussian process is a method of putting a prior over a function with inference taking place in the "function space", not to be confused with the earlier mentioned feature-space. Gaussian processes are also known in spatial statistics as kriging (Cressie, 1993). For a more detailed description than what is presented in the following sections, the reader is referred to (Williams, 1997) , (Tipping, 2001), and (Rasmussen et al, 2006).

4.1 Gaussian Processes

A Gaussian Process is a collection of random variables, any finite of which have a joint Gaussian distribution that can be completely defined by its mean function $\mu = \mathbf{m}(\mathbf{x})$ and its covariance function $\mathbf{K}(\mathbf{X}, \mathbf{X})$.

Definition 4.1.1 Given an index set X and a collection of random variables $\mathbf{F}(\mathbf{x})$ with $x \in X$, if for every finite set $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, $\mathbf{F}(\mathbf{x}_i)$ has a multivariate Gaussian distribution with mean $\mu \in \Re^n$

and covariance $\mathbf{K} \in \Re^{n \times n}$, $\mathbf{F}(\mathbf{x})$ is a Gaussian process (GP).

$$\mathbf{y} = \mathbf{F}(\mathbf{x}) \sim \boldsymbol{GP}(\mu, \mathbf{K}) \tag{4.1}$$

Consider the simple case when the observations are noise free,

Let $\mathbf{y}$ be the known function values of the training cases and $\mathbf{y}_*$ correspond to the test inputs. The joint distribution for a zero mean Gaussian process is

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{y}_* \end{bmatrix} \sim N\left(0, \begin{bmatrix} \mathbf{K} & \mathbf{K}_* \\ \mathbf{K}_*^{\boldsymbol{T}} & \mathbf{K}_{**} \end{bmatrix}\right) \tag{4.2}$$

where $\mathbf{K} = \mathbf{K}(\mathbf{X}, \mathbf{X})$ is the training set covariance, $\mathbf{K}_{**} = \mathbf{K}(\mathbf{X}_*, \mathbf{X}_*)$ is the test set covariance, and $\mathbf{K}_* = \mathbf{K}(\mathbf{X}_*, \mathbf{X})$ is the training-test set covariance with $\mathbf{X}$ and $\mathbf{X}_*$ the training and test input data matrix respectively.

The conditional distribution of $\mathbf{y}_*$ given $\mathbf{y}$ is normal with the expected prediction

$$\mathbf{E}\left[\mathbf{y}_* \mid \mathbf{y}\right] = \mathbf{K}_*^{\boldsymbol{T}} \mathbf{K}^{-1} \mathbf{y} \tag{4.3}$$

and variance

$$Var\left(\mathbf{y}_* \mid \mathbf{y}\right) = \mathbf{K}_{**} - \mathbf{K}_*^{\boldsymbol{T}} \mathbf{K}^{-1} \mathbf{K}_*, \tag{4.4}$$

which is the kernelized least squares solution.

In the case of where the model assumes additive independent and identically distributed noise as Gaussian

$$\epsilon \sim \boldsymbol{N}(0, \sigma_n^2) \tag{4.5}$$

with zero mean and variance σ_n^2 such that the covariance prior

$$cov\left(\mathbf{y}\right) = \mathbf{K} + \sigma_n^2 \mathbf{I}, \tag{4.6}$$

the joint distribution becomes

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{y}_* \end{bmatrix} \sim N\left(0, \begin{bmatrix} \mathbf{K}+\sigma_n^2\mathbf{I} & \mathbf{K}_* \\ \mathbf{K}_*^{\boldsymbol{T}} & \mathbf{K}_{**} \end{bmatrix}\right). \tag{4.7}$$

This gives the regularized version of the kernelized least squares with the expected prediction

$$\mathbf{E}\left[\mathbf{y}_* \mid \mathbf{y}\right] = \mathbf{K}_*^{\boldsymbol{T}}\left(\mathbf{K}+\sigma_n^2\mathbf{I}\right)^{-1}\mathbf{y} \tag{4.8}$$

and variance

$$Var\left(\mathbf{y}_* \mid \mathbf{y}\right) = \mathbf{K}_{**} - \mathbf{K}_*^{\boldsymbol{T}}\left(\mathbf{K}+\sigma_n^2\mathbf{I}\right)^{-1}\mathbf{K}_* \tag{4.9}$$

with $\sigma_n^2 \geq 0$ as the global regularization complexity parameter.

Defining the prior as

$$p(\mathbf{y} \mid \mathbf{X}, \theta) \sim \boldsymbol{N}(0, \mathbf{K}) \tag{4.10}$$

or

$$p(\mathbf{y} \mid \mathbf{X}, \theta) \sim \boldsymbol{N}(0, \mathbf{K}+\sigma_n^2\mathbf{I}) \tag{4.11}$$

with zero mean is a common choice, since the Gaussian random variable is completely characterized by the covariance function.

There are two equivalent views (Williams, 1997) of the Gaussian Process, the weight space and the function space. From the weight-space perspective, a Gaussian Process is a Bayesian linear regression with kernels. In the function-space view, a Gaussian process is a distribution over functions. Both views are equivalent.

Given a set of n observations, $\mathbf{X}$ is the $n \times p$ data matrix of observations, $\mathbf{y}$ are the observed real responses, and $\boldsymbol{B} : \Re^p \rightarrow \Re^q$ is the design matrix with $B_j(\mathbf{X})$ as it j-th row basis function response to input $\mathbf{X}$.

The weight space view, a Gaussian process as a Bayesian linear regression with kernels, is

$$\mathbf{y} = f(\mathbf{X}) + \epsilon \tag{4.12}$$

with

$$f(\mathbf{X}) = \mathbf{B}^{\boldsymbol{T}}\boldsymbol{w} \tag{4.13}$$

and

$$\begin{array}{rl} \epsilon \sim & \boldsymbol{N}(0, \sigma_n^2) \\ \boldsymbol{w} \sim & \boldsymbol{N}(0, \boldsymbol{\Sigma}_p) \end{array}. \tag{4.14}$$

The function space view, a Gaussian process as a distribution over functions f, is

$$\mathbf{y} = f(\mathbf{X}) + \epsilon \tag{4.15}$$

with

$$f(\mathbf{X}) \sim \boldsymbol{GP}(\mu, \mathbf{K}) \tag{4.16}$$

where

$$\mu \equiv \mathbf{E}\left[f(\mathbf{X})\right] \tag{4.17}$$

$$\mathbf{K}(\mathbf{X}, \mathbf{X}) \equiv \mathbf{E}\left[\left(f(\mathbf{X}) - \mu\right)\left(f(\mathbf{X}) - \mu\right)^{\mathbf{T}}\right] \tag{4.18}$$

and

$$\epsilon \sim \boldsymbol{N}(0, \sigma_n^2). \tag{4.19}$$

4.1.1 Bayesian Linear Regression, Weight Space View

Returning to the Radial Basis Function model,

$$f(\mathbf{X}) = \mathbf{y} = \sum_{j=1}^{m} w_j B_j(\mathbf{X}) + \epsilon = \mathbf{B}^{\boldsymbol{T}}\boldsymbol{w} + \epsilon, \tag{4.20}$$

which is linear in the weights, $\boldsymbol{w}$. $\boldsymbol{B}$ is an arbitrary transformation of $\mathbf{X}$ and the error is assumed to be independent and identically normally distributed with zero mean and variance σ_n^2.

$$\epsilon \sim \boldsymbol{N}(0, \sigma_n^2) \tag{4.21}$$

The probability density of the observed values, likelihood, given the parameters is then

$$Pr(\mathbf{y} \mid \mathbf{B}, \boldsymbol{w}) = \boldsymbol{N}(\mathbf{B}^{\boldsymbol{T}}\boldsymbol{w}, \sigma_n^2\mathbf{I}) \tag{4.22}$$

Following the Bayesian method, let the weights have a prior distribution which is zero mean Gaussian with covariance matrix Σ_p.

$$\boldsymbol{w} \sim \boldsymbol{N}(0, \boldsymbol{\Sigma}_p) \tag{4.23}$$

By Bayes rule, the posterior of the weights is given by

$$p(\boldsymbol{w}|\ \mathbf{B}, \mathbf{y}) = \frac{p(\mathbf{y} \mid \mathbf{B}, \boldsymbol{w})p(\boldsymbol{w})}{p(\mathbf{y} \mid \mathbf{B})} \tag{4.24}$$

where the marginal likelihood is

$$p(\mathbf{y} \mid \mathbf{B}) = \int p(\mathbf{y} \mid \mathbf{B}, \boldsymbol{w})p(\boldsymbol{w})d\boldsymbol{w}. \tag{4.25}$$

Using only the terms from the likelihood and prior which depend on the weights, the posterior distribution of the weights $\boldsymbol{w}$ can be shown to be Normal.

$$p(\boldsymbol{w}|\boldsymbol{B}, \mathbf{y}) = \boldsymbol{N}(\beta\mathbf{A}^{-1}\mathbf{B}^{\mathbf{T}}\mathbf{y}, \mathbf{A}^{-1}) \tag{4.26}$$

where

$$\mathbf{A} = \beta\mathbf{B}^{\mathbf{T}}\mathbf{B}+\boldsymbol{\Sigma}_p^{-1} \tag{4.27}$$

and

$$\beta = \frac{1}{\sigma_n^2}. \tag{4.28}$$

The mean prediction for a new input $\mathbf{X}_*$ is

$$\widehat{\mathbf{y}} = \widehat{f}(\mathbf{X}_*) = \mathbf{B}_*^{\mathbf{T}}\beta\mathbf{A}^{-1}\mathbf{B}^{\mathbf{T}}\mathbf{y} \tag{4.29}$$

with variance

$$\mathbf{B}_*^{\mathbf{T}}\mathbf{A}^{-1}\mathbf{B}_*. \tag{4.30}$$

4.1.2 Bayesian Linear Regression, Function Space View

The feature space considers the set of Basis Function $\mathbf{B}$ to be fixed with random weights $\mathbf{w}$ where

$$\mathbf{w} \sim \boldsymbol{N}(0, \boldsymbol{\Sigma}_p). \tag{4.31}$$

Let $\mathbf{y}$ be a random variable defined as

$$\mathbf{y} = \sum_{j=1}^{m} w_j B_j(\mathbf{X}) = \mathbf{B}^{\boldsymbol{T}}\boldsymbol{w}. \tag{4.32}$$

Since $\boldsymbol{w}$ is Gaussian, $\mathbf{y}$ is also Gaussian with mean

$$\mathbf{E}[\mathbf{y}] = \mathbf{E}\left[\mathbf{B}^{\boldsymbol{T}}\boldsymbol{w}\right] = \mathbf{B}^{\boldsymbol{T}}\mathbf{E}[\boldsymbol{w}] = 0 \tag{4.33}$$

and variance

$$\mathbf{E}\left[\mathbf{y}\mathbf{y}^{\boldsymbol{T}}\right] = \mathbf{E}\left[\left(\mathbf{B}^{\boldsymbol{T}}\boldsymbol{w}\right)\left(\mathbf{B}^{\boldsymbol{T}}\boldsymbol{w}\right)^{\boldsymbol{T}}\right] = \mathbf{B}^{\boldsymbol{T}}\mathbf{E}\left[\boldsymbol{w}\boldsymbol{w}^{\boldsymbol{T}}\right]\mathbf{B} = \mathbf{B}^{\mathbf{T}}\boldsymbol{\Sigma}_p\mathbf{B}, \tag{4.34}$$

to find the prediction for a new input $\mathbf{X}_*$, one uses the definition of a multivariate Gaussian distribution with

$$\mathbf{y}_* \sim N(0, \mathbf{B}_*\boldsymbol{\Sigma}_p\mathbf{B}_*^{\mathbf{T}}). \tag{4.35}$$

Adding noise to get the predicted mean

$$\mathbf{E}[\mathbf{y}_* \mid \mathbf{y}] = \mathbf{B}_*^{\mathbf{T}}\boldsymbol{\Sigma}_{\mathbf{p}}\mathbf{B}^{\mathbf{T}}\mathbf{Z}^{-1}\mathbf{y} \tag{4.36}$$

and variance

$$Var(\mathbf{y}_* \mid \mathbf{y}) = \mathbf{B}_*^{\mathbf{T}}\boldsymbol{\Sigma}_p\mathbf{B}_* - \left(\mathbf{B}_*^{\mathbf{T}}\boldsymbol{\Sigma}_{\mathbf{p}}\mathbf{B}^{\mathbf{T}}\right)\mathbf{Z}^{-1}(\mathbf{B}_*\boldsymbol{\Sigma}_{\mathbf{p}}\mathbf{B}) \tag{4.37}$$

where

$$\mathbf{Z} = \mathbf{B}\boldsymbol{\Sigma}_p\mathbf{B}^\mathbf{T} + \sigma_n^2\mathbf{I}. \tag{4.38}$$

4.1.3 Space Equivalence

To show that both views are equivalent (Rasmussen, 2002), involves showing that the predicted means and variances are equal. For the means, first the equations are simplified my multiplying through

$$\begin{aligned}
\mathbf{B}_*^\mathbf{T}\boldsymbol{\Sigma}_\mathbf{p}\mathbf{B}^\mathbf{T}\mathbf{Z}^{-1}\mathbf{y} &= \mathbf{B}_*^\mathbf{T}\beta\mathbf{A}^{-1}\mathbf{B}^\mathbf{T}\mathbf{y} && (4.39)\\
\boldsymbol{\Sigma}_\mathbf{p}\mathbf{B}^\mathbf{T}\mathbf{Z}^{-1} &= \beta\mathbf{A}^{-1}\mathbf{B}^\mathbf{T} && (4.40)\\
\mathbf{A}\boldsymbol{\Sigma}_\mathbf{p}\mathbf{B}^\mathbf{T} &= \beta\mathbf{B}^\mathbf{T}\mathbf{Z}. && (4.41)
\end{aligned}$$

Then it can be shown that

$$\begin{aligned}
\beta\mathbf{B}^\mathbf{T}\mathbf{Z} &= \beta\mathbf{B}^\mathbf{T}\left(\mathbf{B}\boldsymbol{\Sigma}_p\mathbf{B}^\mathbf{T} + \sigma_n^2\mathbf{I}\right) && (4.42)\\
\beta\mathbf{B}^\mathbf{T}\mathbf{Z} &= \beta\mathbf{B}^\mathbf{T}\mathbf{B}\boldsymbol{\Sigma}_p\mathbf{B}^\mathbf{T} + \mathbf{B}^\mathbf{T} && (4.43)\\
\beta\mathbf{B}^\mathbf{T}\mathbf{Z} &= \left(\beta\mathbf{B}^\mathbf{T}\mathbf{B} + \boldsymbol{\Sigma}_p^{-1}\right)\boldsymbol{\Sigma}_p\mathbf{B}^\mathbf{T} && (4.44)\\
\beta\mathbf{B}^\mathbf{T}\mathbf{Z} &= \mathbf{A}\boldsymbol{\Sigma}_\mathbf{p}\mathbf{B}^\mathbf{T}. && (4.45)
\end{aligned}$$

For the variances, using the matrix inversion lemma, $\mathbf{A}^{-1}$ can be rewritten

$$\begin{aligned}
\mathbf{A}^{-1} &= \left(\beta\mathbf{B}^\mathbf{T}\mathbf{B} + \boldsymbol{\Sigma}_p^{-1}\right)^{-1} && (4.46)\\
\mathbf{A}^{-1} &= \boldsymbol{\Sigma}_p + \boldsymbol{\Sigma}_p\mathbf{B}^\mathbf{T}\left(\mathbf{B}\boldsymbol{\Sigma}_p\mathbf{B}^\mathbf{T} + \sigma_n^2\mathbf{I}\right)^{-1}\mathbf{B}\boldsymbol{\Sigma}_p, && (4.47)
\end{aligned}$$

which then can be substituted back into equation (4.30).

So then, if both are equivalent, which is the preferred or more efficient method? At this point neither, and it depends. The two views are a reverse derivation from the application of the matrix

inverse lemma, the Woodbury Formula (Woodbury, 1950), in reducing the computational load of inverting a matrix. Depending on which is greater, the number of data points n or the number of Basis functions m (or non-transformed data p), determines which method requires inverting a larger matrix. The weight space view requires inverting $\mathbf{A}$ which is $m \times m$ ($p \times p$), and the Function space requires inverting $\mathbf{Z}$ an $n \times n$ matrix. The weight space view is of course preferred when performing simple linear regression, but there are times when the number of basis functions is large in regards to the number of observations.

4.2 Gaussian Process and the Feature Space

Note that the above Gaussian Process while modeling a nonlinear transformed space through the use of basis functions is still a linear model; it is linear in the weights $\boldsymbol{w}$. The kernel trick can be applied by noting that

$\mathbf{B}_*^{\mathbf{T}}\boldsymbol{\Sigma}_p\mathbf{B}_*$, $\mathbf{B}_*^{\mathbf{T}}\boldsymbol{\Sigma}_{\mathbf{p}}\mathbf{B}^{\mathbf{T}}$, $\mathbf{B}\boldsymbol{\Sigma}_p\mathbf{B}^{\mathbf{T}}$, and $\mathbf{B}_*\boldsymbol{\Sigma}_{\mathbf{p}}\mathbf{B}$ are inner products. $\boldsymbol{\Sigma}_{\mathbf{p}}$ is a covariance function and can be defined as

$$\boldsymbol{\Sigma}_{\mathbf{p}} = \boldsymbol{\Sigma}_{\mathbf{p}}^{\frac{\mathbf{1}}{\mathbf{2}}}\boldsymbol{\Sigma}_{\mathbf{p}}^{\frac{\mathbf{1}}{\mathbf{2}}} \tag{4.48}$$

so that

$$\mathbf{B}\boldsymbol{\Sigma}_p\mathbf{B}^{\mathbf{T}} = \left(\mathbf{B}\boldsymbol{\Sigma}_{\mathbf{p}}^{\frac{\mathbf{1}}{\mathbf{2}}}\right)\left(\boldsymbol{\Sigma}_{\mathbf{p}}^{\frac{\mathbf{1}}{\mathbf{2}}}\mathbf{B}^{\mathbf{T}}\right) = \mathbf{Z}(\mathbf{X})\mathbf{Z}(\mathbf{X})^{\mathbf{T}}. \tag{4.49}$$

Then using the Reproducing Hilbert Space as defined in Equation 1.14,

$$\left\langle \mathbf{Z}(\mathbf{x}), \mathbf{Z}(\mathbf{x}^{\mathbf{T}}) \right\rangle = \left\langle \mathbf{K}(\cdot, \mathbf{x}), \mathbf{K}(\cdot, \mathbf{x}^{\mathbf{T}}) \right\rangle = \mathbf{K}(\mathbf{x}, \mathbf{x}^{\mathbf{T}}), \tag{4.50}$$

the non-linear feature space can be represented as a kernel function of the original data space's inner products.

As noted in the section on Space Equivalence, the choice between using the Weight or Function Space was primarily determined on the size of the inverting matrix, $\mathbf{A}$ which is $m \times m$ or $\mathbf{Z}$ an $n \times n$ matrix. When working in the feature space through the use of an estimating covariance/kernel

function $\mathbf{K}$ and not using explicit basis functions, the resulting equations require the inversion of an $n \times n$ matrix.

4.3 Regression Tree Algorithm

With the use of the decision tree, the data space regions can be represented by explicit basis functions.

$$f(\mathbf{X}) \sim \boldsymbol{GP}(\mu, \mathbf{K}) \tag{4.51}$$

where

$$\mu = \overline{y} = \mathbf{E}\left[B(\mathbf{X})\right]. \tag{4.52}$$

Applying the zero mean kernelized Gaussian Process on the difference between the observed values and the estimated radial basis function, the conditional joint distribution of given is expressed as

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{y}_* \end{bmatrix} \sim N\left(\begin{bmatrix} \mu \\ \mu_* \end{bmatrix}, \begin{bmatrix} \mathbf{K}+\sigma_n^2\mathbf{I} & \mathbf{K}_* \\ \mathbf{K}_*^{\boldsymbol{T}} & \mathbf{K}_{**} \end{bmatrix} \right) \tag{4.53}$$

which gives the regularized version of the kernelized least squares with the expected prediction

$$\mathbf{E}\left[\mathbf{y}_* \mid \mathbf{y}\right] = \mu_* + \mathbf{K}_*^{\boldsymbol{T}} \left(\mathbf{K}+\sigma_n^2\mathbf{I}\right)^{-1} (\mathbf{y} - \mu) \tag{4.54}$$

and variance

$$Var\left(\mathbf{y}_* \mid \mathbf{y}\right) = \mathbf{K}_{**} - \mathbf{K}_*^{\boldsymbol{T}} \left(\mathbf{K}+\sigma_n^2\mathbf{I}\right)^{-1} \mathbf{K}_* \tag{4.55}$$

with the noise σ_n^2 being estimated from the error of the RBF model and the residuals being modeled by the Gaussian process.

4.4 Application Examples

The application of the kernelized genetic decision tree with respect to regression will be illustrated with two data sets, the Boston and Auto-mpg data. The Boston housing and auto data sets were

chosen to present a data set that was primarily continuous variables and one with several nominal attributes .

4.4.1 Boston Housing

The Boston Housing Data set (Harrison, 1978) contains 506 census observations concerning housing values in the suburbs of Boston. This data set was taken from the StatLib library which is maintained at Carnegie Mellon University. In order to compare the different model fits, the Boston Housing Data set was divided into a training set of 455 observations and a testing set of 51 observations. The predictive input values contain 12 continuous and 1 binary-valued attribute.

1. X1 (CRIM): per capita crime rate by town
2. X2 (ZN): Proportion of residential land zoned for lots over $25,000$ sq.ft.
3. X3 (INDUS): Proportion of non-retail business acres per town.
4. X4 (CHAS): Charles River dummy variable (1 if tract bounds river; 0 otherwise).
5. X5 (NOX): nitric oxides concentration (parts per 10 million).
6. X6 (RM): average number of rooms per dwelling.
7. X7 (AGE): proportion of owner-occupied units built prior to 1940.
8. X8 (DIS): weighted distances to five Boston employment centers.
9. X9 (RAD): index of accessibility to radial highways.
10. X10 (TAX): full-value property-tax rate per ten thousand.
11. X11 (PTRATIO): pupil-teacher ratio by town.
12. X12 (B): 1000*(Proportion of Blacks - 0.63)^2
13. X13 (LSTAT): percent lower status of the population.

The output real values y are the median values (in $\$1000's$) of owner-occupied homes in that area.

The next few steps will explore the benefits of the kernelized decision tree by introducing the search algorithm in stages. The structure of these stages can be found in Table 4.1. The first stage begins with a classically grown regression tree which is pruned to find the best fitting Gaussian RBF network. Stage two adds variable subset selection via the genetic algorithm. Stage three adds kernel and covariance smoother selection. Stage four extends the program algorithm by using the kernelization splitting rule in place of the classical squared error method.

For each stage, the training and test set model fit values are summarized in Table 4.2. The RBF R^2 is the fit using a radial basis function network. The Kernel R^2 is the Gaussian Process fit; and the Tree R^2 is the model fit using the leaf node's average response. Pruning is performed to find the best RBF R^2 for all stages with the same ICOMP method as the genetic kernelized regression tree.

In the first stage, the model decision tree is constructed allowing only one variable splits using all 12 predictor variables. The regression splitting rule is the standard regression tree method by reduction in squared error. The tree is pruned to find the best fitting Gaussian RBF network using

Table 4.1: Boston Housing Algorithm Stage Structure

Stage	Split	Subset	Kernel/Cov Search	Kernel	Covariance
1	Single/Squared Error	No	No	Gaussian	MLE
2	Single/Squared Error	Yes	No	Gaussian	MLE
3	Single/Squared Error	Yes	Yes	Cauchy	SDCE
4	Multi/ICOMP	Yes	Yes	Cauchy	MLEEB

Table 4.2: Boston Housing Algorithm Stage Results

Stage	Data	ICOMP	Nodes	Variables	RBF R^2	Kernel R^2	Tree R^2
1	Train	3000	8	1 - 12	0.10	0.14	0.78
	Test				0.12	0.15	0.75
2	Train	2865	4	1 6	0.47	0.54	0.57
	Test				0.40	0.43	0.46
3	Train	2404	24	1 3 6 7 9 11 12 13	0.85	0.82	0.84
	Test				0.84	0.89	0.75
4	Train	2346	26	1 3 6 7 11 12 13	0.91	0.87	0.97
	Test				0.89	0.86	0.78

the maximum likelihood covariance estimator. With the training data set, this regression tree has an ICOMP score of 3000, 8 leaf nodes, an RBF R^2 of 0.10, a Kernel R^2 of 0.14, and Tree R^2 of 0.78. Fitting the model to the test data set, the RBF R^2 score is 0.12, the Kernel R^2 score is 0.15, and Tree R^2 score is 0.75. While a decent fitting regression tree, the resulting fit of the RBF network is weak. This provides a nice illustration that a well fitting decision tree does not always lead to a well fitting RBF network.

In the second stage the search algorithm is extended to allow variable subset selection through the genetic algorithm. All other constraints mentioned above remain the same. Splits are still being performed with only one variable. The kernel function is still Gaussian with the maximum likelihood covariance estimator. Building with the training data set, the regression tree has an ICOMP score of 2865, 4 leaf nodes, an RBF R^2 equal to 0.47, a Kernel R^2 of 0.54, and Tree R^2 of 0.57. The subset of variables is reduced to two: X1 , per capita crime rate, and X6, average number of rooms. Fitting the model to the test data set, the RBF R^2 score is 0.40, a Kernel R^2 score is 0.43, and Tree R^2 score is 0.46. The increased performance and predictive ability is typical for a simpler and more parsimonious model.

Next, stage three is allowing a search over different kernels and covariance estimators. The Cauchy kernel is selected with the stipulated diagonal covariance estimator. The training regression tree has an ICOMP score of 2404, 24 leaf nodes, an RBF R^2 of 0.85, a Kernel R^2 of 0.82, and Tree R^2 of 0.85. The subset of variables is reduced from twelve to eight. Fitting the model to the test data set, the results are a RBF R^2 score of 0.84, a Kernel R^2 score of 0.89, and Tree R^2 score of 0.75. This illustrates the importance (in some data sets) of correctly choosing the RBF function.

In the last stage, the regression tree is constructed with the full genetic algorithm. The splitting rule is the regularized version of the kernelized least squares. Table 4.3 shows the selected models as the GA progresses in its search. The results illustrate that while ICOMP prefers a higher fit, it is not at the expense of extra variables and tree structure. Fitting the best model listed to the test data set, the results are a RBF R^2 score of 0.89, a Kernel R^2 score of 0.86, and TreeR^2 score of 0.78. A slight improved result over stage three, stage four differs in selecting the maximum likelihood empirical Bayes covariance estimator while dropping variable X9. The RBF R^2 score of 0.89 is an

improvement over the classically train Tree R^2 score of 0.75. An illustration of this regression tree is given in Figure 4.1.

4.4.2 Auto MPG

The Auto MPG data was downloaded from the UCI Machine Learning Data Repository and concerns fuel consumption in miles per gallon (Quinlan, 1993). This data set is divided into a training set of 352 observations and a testing set of 40 observations. The predictive input values contain 4 continuous and 3 multivalued discrete attributes.

1. X1 cylinders: multi-valued discrete
2. X2 displacement: continuous
3. X3 horsepower: continuous
4. X4 weight: continuous
5. X5 acceleration: continuous
6. X6 model year: multi-valued discrete
7. X7 origin: multi-valued discrete

The output real values y is the fuel consumption in miles per gallon (mpg).

Table 4.3: Boston Housing Kernelized Search Results

ICOMP	Kernel	CoVar	Variables	Nodes	RBF R^2	Ker R^2	Tree R^2
2345	Cauchy	MLEEB	1,3,6,7,11-13	26	0.91	0.87	0.97
2348	InvMultiQuad	MLEEB	1-4,6,7,9,11-13	29	0.91	0.90	0.91
2376	InvMultiQuad	MLEEB	1,6,8,11,13	20	0.89	0.63	0.90
2412	Exp	SDCE	1,6,8,11,13	19	0.88	0.88	0.89
2414	Cauchy	MLEEB	1,2,4,6,8,10,13	19	0.88	0.80	0.88
2475	Cauchy	MLE	1,3,6,7,10-13	19	0.86	0.84	0.86
2607	InvMultiQuad	None	8,11,13	16	0.79	0.73	0.79
2919	Power Exp	None	1,2,5,7-9,11,13	23	0.53	0.58	0.78

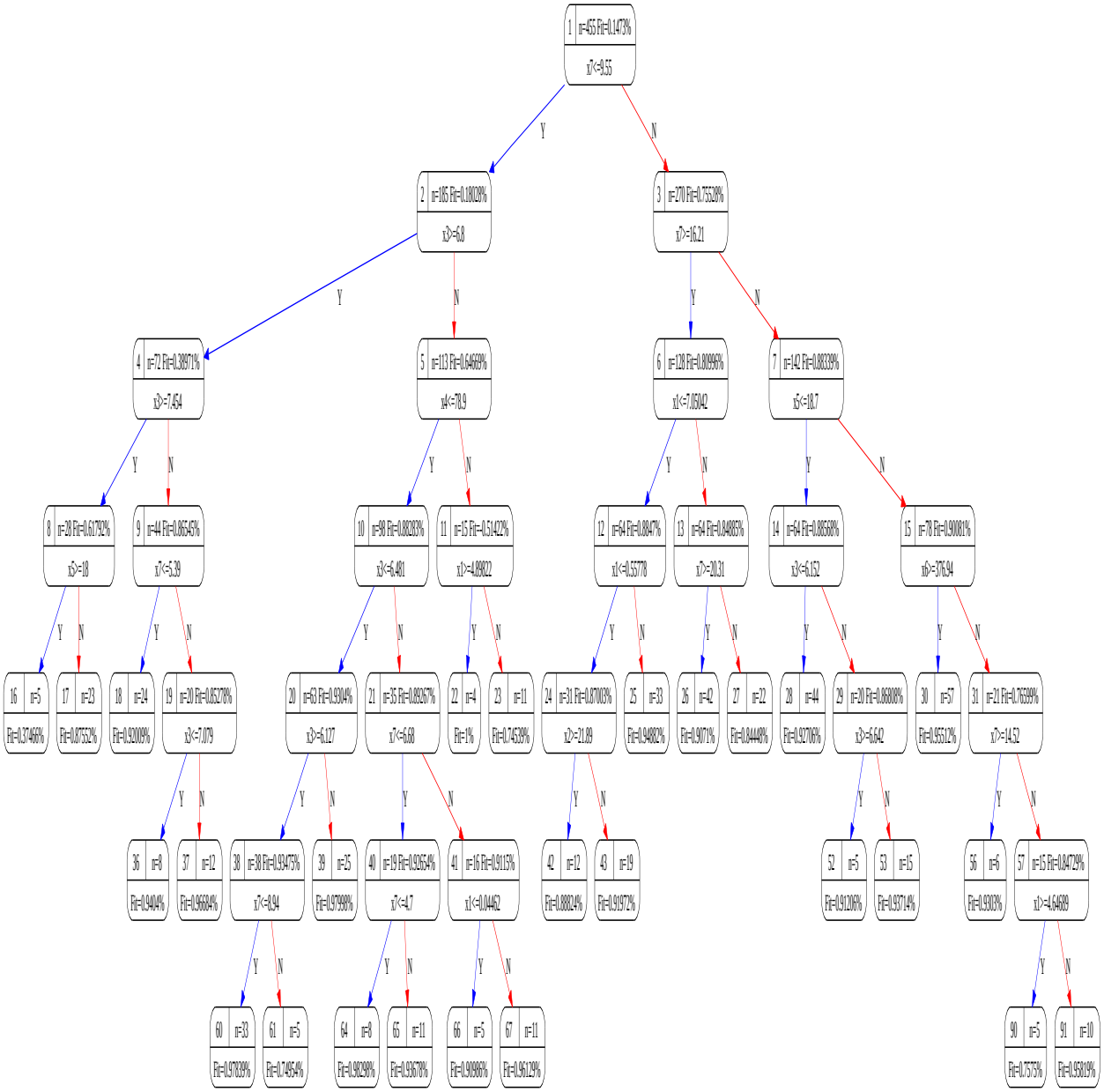

Figure 4.1: Boston Regression Tree

As for the Boston Housing data set, the next few steps will explore the benefits of the kernelized decision tree by introducing the search algorithm in stages. The structure of these stages can be found in Table 4.4. The first stage begins with a classically grown regression tree which is pruned to find the best fitting Gaussian RBF network. Stage two adds variable subset selection via the genetic algorithm. Stage three adds kernel and covariance smoother selection. Stage four extends the program algorithm by using the kernelization splitting rule.

For each stage, the training and test set model fit values are summarized in Table 4.5. The RBF R^2 is the fit using a radial basis function network. The Kernel R^2 is the Gaussian process fit; and the Tree R^2 is the model fit using the leaf node's average response. Pruning to find the best RBF R^2 is performed for all stages with the same ICOMP method as the genetic kernelized regression tree.

In the first stage, model decision tree is constructed allowing only one variable splits using all 7 predictor variables. The regression splitting rule is the standard reduction in squared error. The tree is pruned to find the best fitting Gaussian RBF network using the maximum likelihood covariance estimator. With the training data set, the regression tree has an ICOMP score of 2284, 5 leaf nodes.

Table 4.4: Auto MPG Algorithm Stage Structure

Stage	Split	Subset	Kernel/Cov Search	Kernel	Covariance
1	Single/Squared Error	No	No	Gaussian	MLE
2	Single/Squared Error	Yes	No	Gaussian	MLE
3	Single/Squared Error	Yes	Yes	Multiquad	Ledoit
4	Multi/ICOMP	Yes	Yes	Multiquad	Ledoit

Table 4.5: Auto MPG Algorithm Stage Results

Stage	Data	ICOMP	Nodes	Variables	RBF R^2	Kernel R^2	Tree R^2
1	Train	2284	5	1 - 7	0.31	0.42	0.75
	Test				0.39	0.43	0.48
2	Train	2105	9	2 6 7	0.61	0.65	0.75
	Test				0.71	0.72	0.51
3	Train	1849	15	1 2 3 6	0.82	0.82	0.81
	Test				0.72	0.73	0.59
4	Train	1575	38	1 2 3 4 5 6	0.91	0.91	0.92
	Test				0.89	0.90	0.62

an RBF R^2 score is 0.31, a Kernel R^2 is 0.42, and Tree R^2 is 0.75. Fitting the model to the test data set, the RBF R^2 score is 0.39, a Kernel R^2 score is 0.43, and Tree R^2 score is equal to 0.48. While a decent fitting regression tree, the resulting fit of the RBF network is weak. Again, this provides a nice illustration that a well fitting decision tree does not always lead to a well fitting RBF network.

In stage two, the base model search algorithm is extended to allow variable subset selection through the genetic algorithm. All other constraints mentioned above remain the same. Splits are still being performed with only one variable. The kernel function is still Gaussian with the maximum likelihood covariance estimator. Building with the training data set, the regression tree has an ICOMP score of 2105, 9 leaf nodes, an RBF R^2 of 0.61, a Kernel R^2 of 0.65, and Tree R^2 of 0.75. The subset of variables selected is X2 , displacement, X6, model year, and X7 or origin. Fitting the model to the test data set, the RBF R^2 score is 0.71, the Kernel R^2 score is 0.72, and Tree R^2 score is 0.51. The increased performance and predictive ability is typical for a simpler and more parsimonious model.

Next, stage three is allowing a search over different kernels and covariance estimators. The kernel selected is the Multiquadratic with Ledoit covariance estimator. The training regression tree has an ICOMP score of 1849, 15 leaf nodes, an RBF R^2 equal to 0.82, a Kernel R^2 of 0.82, and Tree R^2 of 0.81. The subset of variables selected is X1 , X2 , X3, and X6. Fitting the model to the test data set, the RBF R^2 score is 0.72, the Kernel R^2 score is 0.73, and Tree R^2 score is equal to 0.59. This increase in the training data set is due to the choose of the RBF function, but the generalization to the test data set is the same as stage 2.

In the last stage, the regression tree is constructed with the full genetic algorithm. The splitting rule is the regularized version of the kernelized least squares. Table 4.6 shows the selected models as the GA progresses in its search. The results illustrate that while ICOMP prefers a higher fit, it is not at the expense of extra variables and tree structure. Fitting the best model listed to the test data set, the results are a RBF R^2 score of 0.89, a Kernel R^2 score of 0.90, and TreeR^2 score of 0.62. An improved result over stage three (and stage two), stage four agrees with the choice of the kernel and covariance estimator, but has built a larger tree while adding a couple of variables. The

RBF R^2 score of 0.89 is a significant improvement over the classically train Tree R^2 score of 0.48. An illustration of this regression tree is given in Figure 4.2.

Table 4.6: Auto MPG Kernelized Results

ICOMP	Kernel	Covariance	Variables	Nodes	RBF R^2	KER R^2	Tree R^2
1575	Multiquad	Ledoit	1-6	38	0.91	0.92	0.92
1677	Cauchy	MLE	1-3,5-7	43	0.87	0.84	0.89
1791	Multiquad	None	1,3,6,7	24	0.87	0.87	0.87
1833	Power Exp	CSCE	4-6	26	0.84	0.85	0.88
1953	InvMultiQuad	ME	1,2,4,7	13	0.76	0.70	0.76

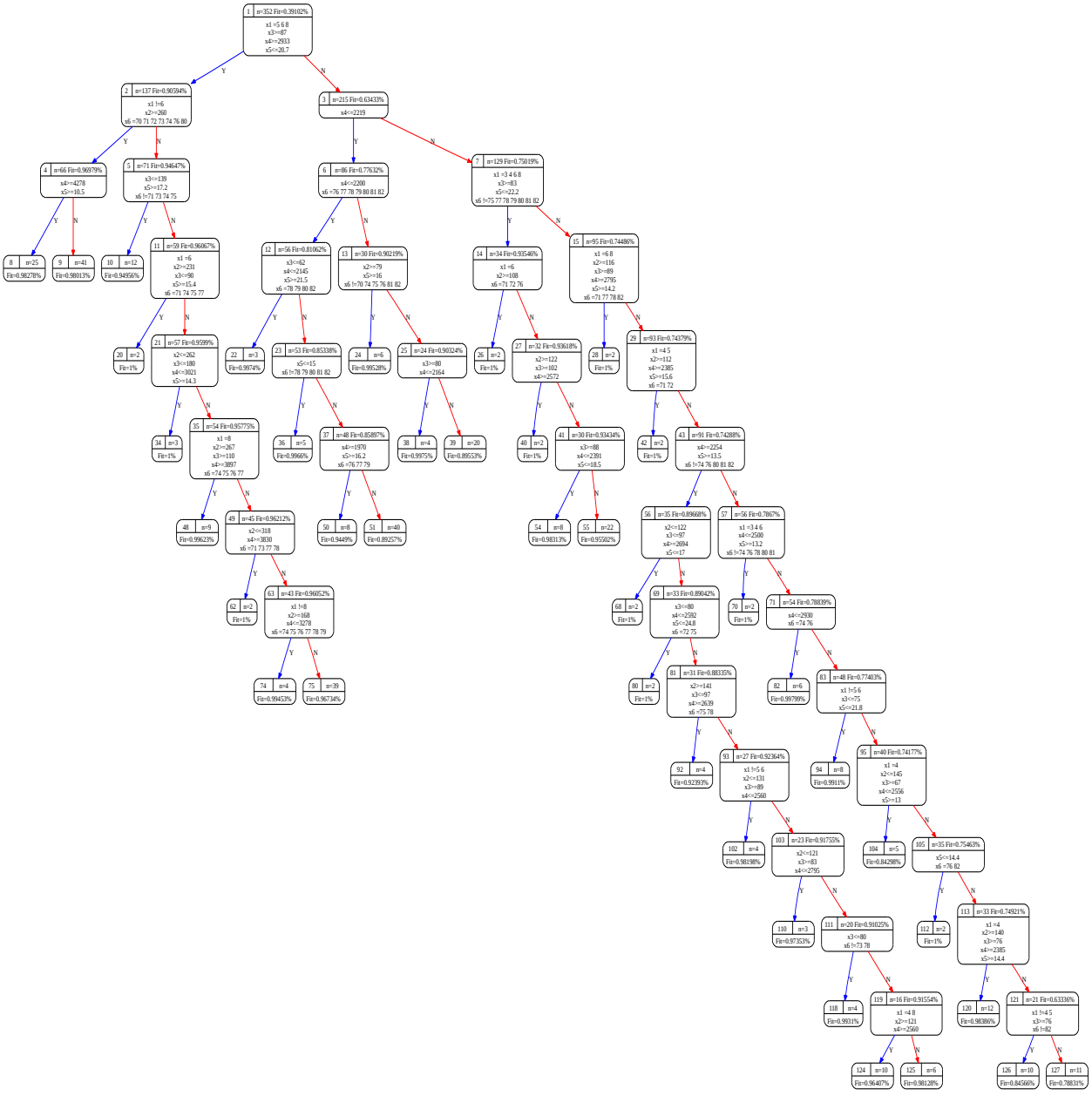

Figure 4.2: Auto Regression Tree

Chapter 5

Genetic Classification Tree

An important problem in many field applications is multi-group classification or discrimination for predicting a class membership based on N measurements of predictors $\mathbf{X} \in \Re^p$ where the outcome belongs to one of G unordered class $G \in \{1, \ldots, C\}$. The topic of ordered class is the subject of the following chapter.

The next step presented in this chapter is extending the kernelized regression tree in order to determine optimal radial basis functions. These radial basis functions transform the data space into function space allowing the linear separation of the classes.

5.1 Multiple Classification Algorithms

As a starting point, an $N \times C$ indicator membership matrix $\mathbf{Y}$ can be constructed from the response g_i such that $Y_{ij} = 1$ if $g_i = j$ and $Y_{ij} = 0$ if $g_i \neq j$.

$$\mathbf{Y} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} g_1 = 1 \\ g_2 = 4 \\ \vdots \\ g_N = 3 \end{bmatrix} \tag{5.1}$$

This membership forms naturally in the programming of the decision tree. It use will also come into play during the classification of ordinal responses. It would be tempting to proceed with a multivariate linear regression and assign the class of a new test point to the class with the largest fitted value. This is especially true since (through the kernel trick) the model is projecting to a dot product space where the classes are linearly separate. However, a known problem of using multivariate regression for classification is that of masking or partial masking of a class by one or more other classes. Using the Fisher Iris data as an example in Figure 5.1, Figure 5.1(a) shows a miss-classification rate of 22.6% were linear discriminant analysis in Figure 5.1(b) has a miss-classification rate of 4%. The misclassified points are noted with a circle in the figures.

Traditional statistical methods for this problem include multinomial logistic regression and linear discriminant analysis.

5.1.1 Multinomial Logistic Regression

Gaussian Process classification constructs a two-step model for the conditional probability $\Pr(G \mid \mathbf{x})$ through a latent variable $\mathbf{u} \in \Re$. In the GP model, $\mathbf{x} \mapsto \mathbf{u}$ is given a Gaussian process prior with zero mean and covariance function $\mathbf{K}$. The conditional probability of $\Pr(C \mid \mathbf{u})$ is then modeled by the softmax function

$$\Pr(C \mid \mathbf{u}) = \frac{\exp(\mathbf{u})}{\sum \exp(\mathbf{u})} = \frac{\exp(\mathbf{X}^{\boldsymbol{T}}\boldsymbol{w})}{\sum \exp(\mathbf{X}^{\boldsymbol{T}}\boldsymbol{w})} \tag{5.2}$$

which is the multinomial response model. This allows one to model the posterior probabilities of the C classes with linear functions (in the data or kernel space) and ensuring that the probabilities remain in $[0, 1]$ and sum to one.

The decision boundary between class a and b is determined by the equation

$$\Pr(G = b \mid X = x) = \Pr(G = a \mid X = x) \tag{5.3}$$

By enforcing a linear boundary, the model is specified by $C - 1$ logit transformations

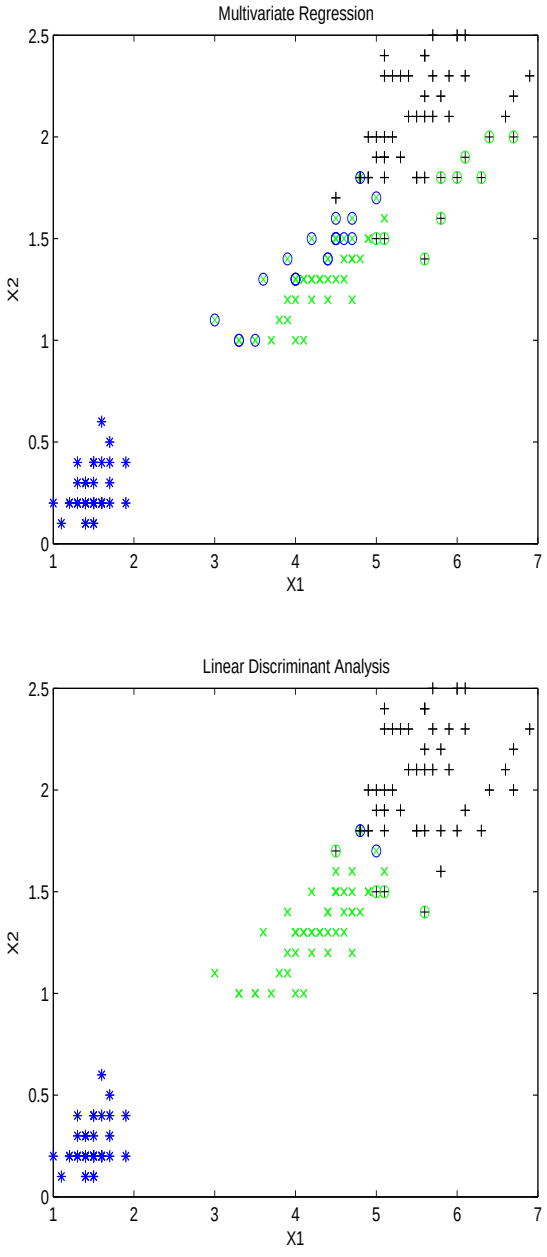

Figure 5.1: Linear Discriminant Analysis versus Multivariate Regression Classification

$$\log \frac{\Pr(G = g_i \mid X = x)}{\Pr(G = C \mid X = x)} = \mathbf{X}^T w_{g_i} \tag{5.4}$$

with $(C-1)(p+1)$ parameters.

The multinomial logistic regression model is the natural extension to the logistic binomial distribution and can handle both nominal and ordinal responses. Estimates of the weights, $\mathbf{w}$, is by maximum likelihood estimation of the log likelihood function

$$\begin{aligned} l(\mathbf{w}) &= \sum_{i=1}^{N} \log \Pr_{g_i}(x_i; w) && (5.5)\\ l(\mathbf{w}) &= \sum_{i=1}^{N} \log\left(\frac{\exp(x_i^T w_{g_i})}{1 + \sum_{j=1}^{C-1} \exp(x_i^T w_j)}\right) && (5.6)\\ l(\mathbf{w}) &= \sum_{i=1}^{N} \left[x_i^T w_{g_i} - \log(\mathbf{1} + \sum_{\mathbf{j=1}}^{\mathbf{C-1}} \exp(x_i^T w_j)) \right]. && (5.7) \end{aligned}$$

This is typically accomplished through iteratively reweighted least squares, Newton's method. As with binary classification

$$\frac{\partial l(\mathbf{w})}{\partial \mathbf{w}} = \widetilde{\mathbf{X}}^T(\mathbf{y} - \mathbf{p}) \tag{5.8}$$

$$\frac{\partial l(\mathbf{w})}{\partial \mathbf{w} \partial \mathbf{w}^T} = -\widetilde{\mathbf{X}}^T \mathbf{Q} \widetilde{\mathbf{X}}, \tag{5.9}$$

the formula to update $\mathbf{w}$ is given by

$$\mathbf{w}^{new} = \mathbf{w}^{old} + (\widetilde{\mathbf{X}}^T \mathbf{Q} \widetilde{\mathbf{X}})^{-1} \widetilde{\mathbf{X}}^T (\mathbf{y} - \mathbf{p}) \tag{5.10}$$

where $\mathbf{y}$ and $\mathbf{p}$ are $N \times (C-1)$ matrices of indicator values and fitted probabilities. $\mathbf{Q}$ is no longer (as in binary classification) a $N \times (C-1)$ diagonal matrix with $\Pr_{g_i}(x_i; w^{old}) \left(1 - \Pr_{g_i}(x_i; w^{old})\right)$ as its ith diagonal element, but a $N(C-1) \times N(C-1)$ matrix

$$\mathbf{Q} = \begin{bmatrix} \mathbf{Q}_{11} & \mathbf{Q}_{12} & \cdots & \mathbf{Q}_{1,C-1} \\ \mathbf{Q}_{21} & \ddots & & \\ \vdots & & & \\ \mathbf{Q}_{C-1,1} & & & \mathbf{Q}_{C-1,C-1} \end{bmatrix} \tag{5.11}$$

with each sub matrix $\mathbf{Q}_{ij}$ being a diagonal matrix. When $i = j$, the diagonal is given by the equation $\Pr_i(x_i; w^{old})\left(1 - \Pr_i(x_i; w^{old})\right)$ otherwise it is $-\Pr_i(x_i; w^{old})$. $\widetilde{\mathbf{X}}$ is an $N(C-1) \times (p+1)(C-1)$ diagonal matrix of $\mathbf{X}$.

$$\widetilde{\mathbf{X}} = \begin{bmatrix} \mathbf{X} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \ddots & & \\ \vdots & & \mathbf{X} & \\ \mathbf{0} & & & \mathbf{X} \end{bmatrix}. \tag{5.12}$$

Computational issues include choice of parameter initialization, manipulation of $\mathbf{Q}$, a $N(C-1) \times N(C-1)$ matrix which when kernelized would be $N(N-1) \times N(N-1)$, and a non-guaranteed convergence.

5.1.2 Flexible Discriminant Analysis

Linear discriminant analysis (LDA) satisfies the assumptions of the linear logistic model given in equation 5.4 . Yet where the linear logistic model only specifies the conditional distribution $\Pr(G = C \mid X = x)$ with no assumptions about $\Pr(X)$, LDA assumes the joint distribution between G and X. $\Pr(G = C \mid X = x)$ is a mixture of Gaussian with a common covariance $\mathbf{\Sigma}$ and $\sum_{j=1}^{C} \pi_j = 1$,

$$\Pr(X) = \sum_{j=1}^{C} \pi_j \phi\left(X; \mu_j, \mathbf{\Sigma}\right), \tag{5.13}$$

so that

$$\log \frac{\Pr(G=j \mid X=x)}{\Pr(G=C \mid X=x)} = X^T w \tag{5.14}$$

$$\log \frac{\Pr(G=j \mid X=x)}{\Pr(G=C \mid X=x)} = \begin{array}{c} \log \frac{\pi_j}{\pi_C} - \frac{1}{2}\left(\mu_j + \mu_C\right)^T \Sigma^{-1}\left(\mu_j - \mu_C\right) \\ +x^T \Sigma^{-1}\left(\mu_j - \mu_C\right) \end{array} \tag{5.15}$$

Linear logistic maximizes the conditional distribution $\Pr(G = C \mid X = x)$ and LDA maximizes the joint distribution $\Pr(G, X)$ While in practice, both give similar results, LDA is less robust against outliers due to the added assumptions.

LDA can be performed as a multi-response linear regression using optimal scores to represent the classes. A method for combining the nonparametric regression techniques and optimal scores is known as Flexible Discriminant Analysis (Hastie, 1994). Ripley (1994b) has also taken these ideas up independently.

First define a new function

$$\Phi : \{1, \ldots, C\} \mapsto \Re^1 \tag{5.16}$$

which assigns a score to each class that are optimally predicted by a linear regression on $\mathbf{X}$. This produces a one-dimensional separation between the classes. More generally, there can be found $L \leq C - 1$ independent scoring labels.$\Phi_1, \Phi_2, \ldots, \Phi_L$ and linear maps $\eta_l(x) = \mathbf{X}^T w_l$. These are chosen as to minimize the average squared residual

$$ASR = \frac{1}{n} \sum_{l=1}^{L} \left[\sum_{i=1}^{n} \left(\Phi\left(g_i\right) - x_i^T w_l \right)^2 \right]. \tag{5.17}$$

The scores are assumed to be mutually orthogonal and normalized with respect to an inner product to prevent trivial solutions.

Hastie (1994) shows that the regression fits η_l and the optimal scores Φ_l can be found in separate steps. Starting with the indicator response matrix $\mathbf{Y}$, equation 5.1, a linear multivariate regression is fitted to give a matrix of fitted estimated coefficients, $\widehat{\mathbf{W}}$, yielding fitted values

$$\widehat{\mathbf{Y}} = \mathbf{X}\widehat{\mathbf{W}} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y} = \mathbf{P}\mathbf{Y} \tag{5.18}$$

Optimal scores $\mathbf{\Phi}$ are found by computing the eigenvector matrix $\mathbf{\Theta}$ of $\mathbf{Y}^T\widehat{\mathbf{Y}}$ normalized so that $\mathbf{\Theta}\mathbf{D}\mathbf{\Theta} = \mathbf{I}$ where $\mathbf{D} = \mathbf{Y}^T\widehat{\mathbf{Y}}/n$, a diagonal matrix of class proportions. The parameter matrix $\widehat{\mathbf{W}}$ is updated to reflect that the regression is with response $\mathbf{Y}\mathbf{\Theta}$ rather than $\mathbf{Y}$, $\widehat{\mathbf{W}} = \mathbf{P}\mathbf{Y}\mathbf{\Theta}$. Since the regression is linear in $\mathbf{Y}$, the update can be performed without refitting the regression,

$$\widehat{\mathbf{W}}_{updated} = \widehat{\mathbf{W}}\Theta. \tag{5.19}$$

Flexible discriminant analysis replaces the linear projection operator $\mathbf{P}$ by a nonparametric regression procedure. In this dissertation kernelized least squares, or more specifically, the Gaussian process outlined in the previous chapter where the expected value of the node is given by the probability that a an observation is in class g given that it is node d

$$\Pr(g \mid d) = \frac{\Pr(g, d)}{\Pr(d)} \tag{5.20}$$

where $\Pr(g, d)$ is the joint probability that an observation will be node d and class g. It is calculated as

$$\Pr(g, d) = \pi_g \frac{n_g(d)}{n_g} \tag{5.21}$$

where n_g is the number of observations that belong to class g, $n_g(d)$ is the number of observations at node d that belong to class g, and π_g is the prior probability that an observation belongs to class g. When unknown the prior can be estimated from the data as

$$\pi_g = \frac{n_g}{n} \tag{5.22}$$

and $\Pr(d)$ is the probability that a an observation is in node d and is calculated by

$$\Pr(d) = \sum_{g=1}^{C} \Pr(g, d). \tag{5.23}$$

5.2 Application Examples

The application of the kernelized genetic decision tree with respect to nominal classification will be illustrated with two data sets, the Vowel and Wine data.

5.2.1 Wine Recognition Data Set

These data are the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivators. The analysis determined the quantities of 13 continuous constituents found in each of three types of wines with 178 observations. The data were donated to the UCI ML repository by Stefan Aeberhard.

In a classification context, this is a well posed problem with behaved class structures and a good data set for first testing of a new classifier, not very challenging. The classes are separable, though only regularized discriminant analysis has achieved 100% correct classification (Aeberhard, 1992). This data set was chosen to illustrate that the kernelized genetic decision tree is performing as expected. Results can be found in Table 5.1 and the Classification Tree in Figure 5.2. One can observe the more parsimonious model being chosen as the fitness function , ICOMP, decreases. Both the number of end nodes in the decision tree and the number of variables are decreasing as the fit of the model increases.

5.2.2 Vowel Recognition Data Set

The vowel data set is a speaker independent recognition of the eleven steady state vowels of British English using a specified training set of derived log area ratios also known as the Deterding data

Table 5.1: Wine Recognition Kernelized Search Results

ICOMP	Kernel	CoVar	Variables	Nodes	RBF Fit	KER Fit	Tree Fit
1.80	RBF	MLEEB	2,7,10	6	0.97	0.97	0.97
1.89	CAUCHY	MLEEB	1,2,7,10	6	0.97	0.97	0.97
2.10	CAUCHY	ME	1,2,7,10,13	5	0.98	0.98	0.98
2.25	CAUCHY	MLEEB	1,2,7,10	7	0.97	0.97	0.97
2.34	CAUCHY	MLEEB	1-3,6-8,12	4	0.94	0.94	0.94
2.65	MULTIQUAD	MLEEB	1,4,8,10,12	6	0.93	0.93	0.93

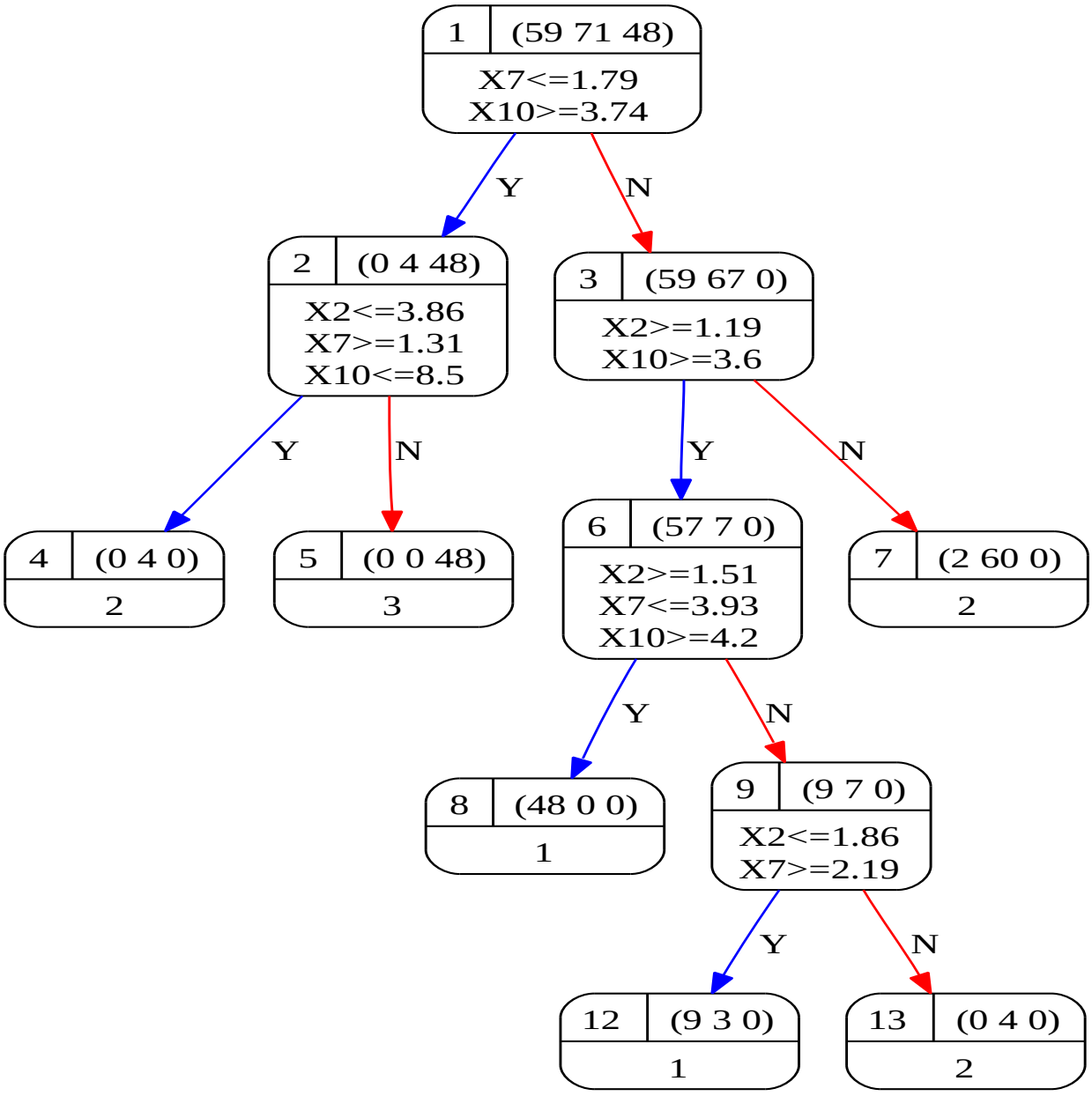

Figure 5.2: Wine Nominal Classification Tree

set (Deterding, 1989). There are $G = 11$ classes in 10 dimensions. The training set has 528 observations and the test set has a total of 462 observations. Unlike the wine recognition data set, this is a difficult data set. Since the covariance structure of the test set is significantly different than the training data set, most methods tend to over fit with a test accuracy of 40%. Linear regression has a test accuracy of 33% and linear discriminant analysis has a test accuracy of 44%. Methods such a k-means or nearest neighbor that use the full training set for mapping tend to peak at a 55% test accuracy.

The next few steps will explore the benefits of the kernelized classification tree. The structure of these stages can be found in Table 5.2. The first stage begins with a classically grown decision tree which is pruned to find the best fitting Gaussian RBF network. Stage two adds variable subset selection via the genetic algorithm. Stage three adds kernel and covariance smoother selection. Stage four extends the program algorithm by using the kernelization splitting rule.

For each stage, the training and test set model fit values are summarized in Table 5.3. The RBF fit is the percent correctly classified using a radial basis function classification network; and the Tree fit is the classification accuracy using the predicted class response at each decision tree leaf node. Of main interest is the RBF fit upon which the tree is pruned. Pruning is performed for all stages with the same ICOMP method as the genetic kernelized classification tree. The Kernel fit is a kernelized linear discriminate analysis model updated with the information provided by the nominal classification tree. The kernel model is still considered experimental by the author, but is provided for completeness of information.

In the first stage, model decision tree is constructed allowing only one variable splits using all 10 predictor variables. The classification splitting rule is deviance. For the training data set, this tree finished with an ICOMP score of 6.26, 17 leaf nodes. an RBF fit of 0.64, a Kernel fit of 0.98,

Table 5.2: Vowel Algorithm Stage Structure

Stage	Split	Subset	Kernel/Cov Search	Kernel	Covariance
1	Single/Deviance	No	No	Gaussian	MLE
2	Single/Deviance	Yes	No	Gaussian	MLE
3	Single/Deviance	Yes	Yes	Power Exp	SRCE
4	Multi/ICOMP	Yes	Yes	Gaussian	MLEEB

and Tree fit of 0.66. Fitting the model to the test data set, the RBF fit is equal to 0.09, the Kernel fit is 0.98, and Tree fit is 0.39. It is known that classification trees will tend to over fit the training set with poor generalization to the testing set of data. This seems particular so with the vowel data set.

The kernelized linear discriminant fit for the training data set is reasonable. Without using the tree or a tree with no splits, a straight kernelized linear discriminant analysis will give the same high fit for both the training and test data sets. Again, the vowel data set is known for its poor generalization with the best results from models that use the full training set for mapping. Because the mercer kernel trick uses the $n \times n$ Gram matrix, there is a complete observational mapping of the data. The purpose of the method discussed in this thesis is to use the information of the kernel method to determine a smaller dimension RBF network that performs well. As this illustration moves through the presented stages, it can be seen that the performance of the RBF network and the kernel linear discriminant analysis updated with the information provided by the nominal classification tree tend to converge.

In stage two, the genetic algorithm is extended to allow variable subset selection. Splits are still being performed with only one variable. The kernel function is still Gaussian with the maximum likelihood covariance estimator. For the training data set, the selected tree has an ICOMP score of 5.03, 40 leaf nodes, an RBF fit of 0.85, a Kernel fit of 0.72, and Tree fit of 0.84. The subset of variables selected is X1, X2, and X8. Fitting the model to the test data set, the RBF fit is 0.09, a

Table 5.3: Vowel Algorithm Stage Results

Stage	Data	ICOMP	Nodes	Variables	RBF Fit	Kernel Fit	Tree Fit
1	Train	6.26	17	1 - 10	0.64	0.98	0.66
	Test				0.09	0.98	0.39
2	Train	5.03	40	1 2 8	0.85	0.72	0.84
	Test				0.09	0.37	0.45
3	Train	4.88	45	1 2 5 6 7 8 9 10	0.86	0.95	0.86
	Test				0.41	0.30	0.37
4	Train	4.61	49	1 2 5 8 10	0.88	0.91	0.88
	Test				0.42	0.46	0.42

Kernel fit is 0.37, and Tree fit 0.45. Again, there is poor generalization for the RBF network even with the variable subset.

Stage three allows a search over different kernels and covariance estimators. The kernel selected is the power exponential with the stipulated regularized covariance estimator. The training classification tree has an ICOMP score of 4.88, 45 leaf nodes, an RBF fit equal to 0.88, a Kernel fit of 0.95, and Tree fit of 0.85. Eight of the ten variables are selected for the subset. Fitting the model to the test data set, the RBF fit is 0.41, the Kernel fit is 0.30, and Tree fit score is equal to 0.37. This increase in the training data set is mostly to the choice of the RBF function.

Using the full search genetic algorithm using the splitting rule based on the kernelized linear discriminant analysis, Table 5.4 shows the fitness function is favoring a smaller set of variables and terminal nodes. This is while generating good generalization results without using techniques such as cross validation. The accuracy of the radial basis function fit in stage four is on par with past literature results with a substantial reduction in variables from stage three. The RBF Fit score of 0.42 is a slight improvement over the classically train Tree Fit score of 0.39. An illustration of the kernelized classification tree can be found in Figure 5.3.

Table 5.4: Vowel Recognition Kernelized Search Results

Data Set	ICOMP	Kernel	CoVar	Variables	Nodes	RBF	Ker	Tree
Train	4.61	Gaussian	MLEEB	1,2,5,8,10	49	0.88	0.91	0.88
Test						0.42	0.46	0.42
Train	4.83	Gaussian	MLEEB	1,5,6,8,10	56	0.89	0.91	0.89
Test						0.31	0.31	0.31
Train	4.90	Gaussian	MLEEB	2-6,8	63	0.82	0.88	0.82
Test						0.29	0.31	0.29
Train	5.00	Gaussian	MLEEB	1-6,10	46	0.86	0.97	0.86
Test						0.35	0.36	0.35
Train	5.26	Cauchy	ME	2,6,7,9	61	0.80	0.79	0.80
Test						0.30	0.30	0.30

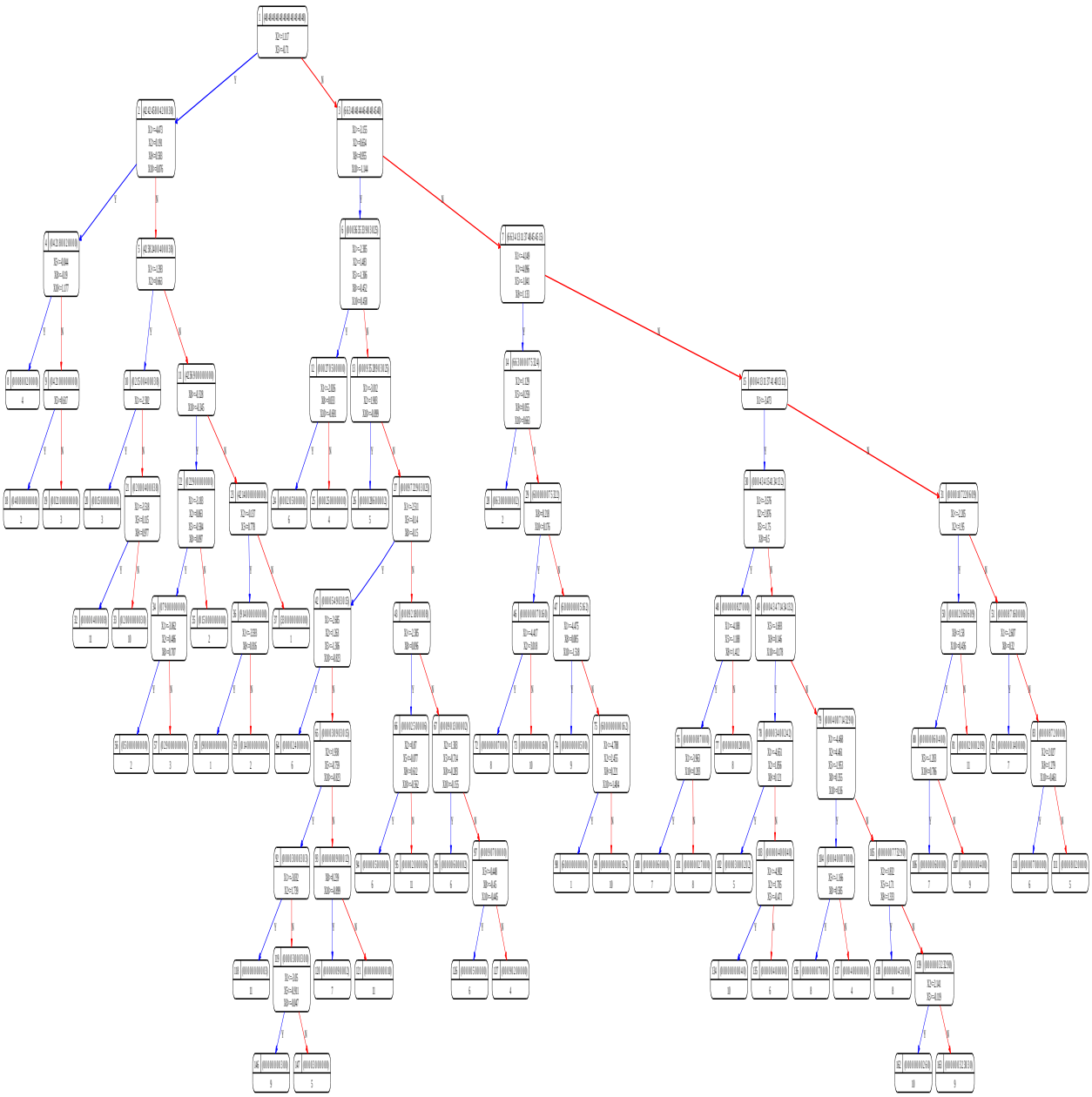

Figure 5.3: Vowel Nominal Classification Tree

Chapter 6

Genetic Ordinal Tree

Modification the classical classification tree to handle ordinal attribute data is through the use of nested dichotomies,

$$\begin{aligned} [1,2,...,C] &\Longrightarrow [1 \mid 2] \\ &\Longrightarrow [1,2 \mid ...] \\ &\Longrightarrow [1,2,... \mid C]\,. \end{aligned} \tag{6.1}$$

The ordinal classification tree is then prepared for the data for the cumulative log-odds model. In the special case of the 'backward' continuation-ratio model, each response, $Y = i$, is compared to all lower responses, $Y < i$

$$\Pr_g(x) = \ln\left[\frac{\Pr(Y=g|x)}{\Pr(Y<g|x)}\right]. \tag{6.2}$$

If there are only two cases, everything reduces to the Logistic RBF model as a special case. To change the classification algorithm in Chapter 5, the expected class count is reassigned according to the cumulative nested grouping

$$\Pr(g \mid d) = \frac{1}{n}\sum_{i \in D} I(y_i \leq i). \tag{6.3}$$

With the standard classification algorithm , the $N \times C$ indicator membership matrix $\mathbf{Y}$ is constructed from the response g_i such that $Y_{ij} = 1$ if $g_i = j$ and $Y_{ij} = 0$ if $g_i \neq j$

$$\mathbf{Y} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} g_1 = 1 \\ g_2 = 4 \\ \vdots \\ g_N = 3 \end{bmatrix}. \tag{6.4}$$

Summing along the vertical dimension, allows one to calculate the expected probabilities for the node d

$$\Pr(g \mid d) = \frac{\Pr(g, d)}{\Pr(d)} \tag{6.5}$$

where $\Pr(g, d)$ is the joint probability that an observation will be node d and class g.

$$\Pr(g, d) = \pi_g \frac{n_g(d)}{n_g} \tag{6.6}$$

and n_g is the number of observations that belong to class g. $n_g(d)$ is the number of observations at node d that belong to class g and is calculated by the vertical summation of the indicator matrix. π_g is the prior probability that an observation belongs to class g and $\Pr(d)$ is the probability that an observation is in node d and is calculated by

$$\Pr(d) = \sum_{g=1}^{C} \Pr(g, d). \tag{6.7}$$

In order to utilize the cumulative value of the nested probabilities, the $N \times C$ indicator membership matrix $\mathbf{Y}$ is constructed from the response g_i such that $Y_{ij} = 1$ if $g_i <= j$ and $Y_{ij} = 0$ if $g_i > j$.

$$\mathbf{Y} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} g_1 = 1 \\ g_2 = 4 \\ \vdots \\ g_N = 3 \end{bmatrix} \tag{6.8}$$

Figure (6.1) shows the classification tree applying ordinal splits to the Iris data with a just the standard classification tree, no information criteria or kernelization. The classes were assigned such that Virginica = 1, Versicolor = 2, and Setosa = 3.

6.1 Application Examples

The application of the kernelized genetic decision tree with respect to ordinal classification will be illustrated with two data sets, the Fisher Iris and the Abalone data sets.

6.1.1 Fisher Iris

The Fisher Iris data contains 150 observations with 50 in each of three classes. As shown in Figure 6.1(a), one class is linearly separable from the other two. Versicolor and Virginica are not linearly separable from each other. There are four numeric attributes

1. X1: sepal length in cm
2. X2: sepal width in cm.
3. X3 : petal length in cm.
4. X4 : petal width in cm.

While a non ordinal problem, this data set was chosen to illustrate that the kernelized genetic decision tree performs as should be expected. The data structure as shown in Figure 3.2(a) shows a problem that can be extended to the ordinal prediction. The Iris classes are assigned such that Virginica = 1, Versicolor = 2, and Setosa = 3. Results can be found in Table 6.1 and the Classification Tree in Figure 6.2. A well fitting proportional model can be attained with just variable X4; but the addition of X3 adds enough information to be selected by the ICOMP fitness value.

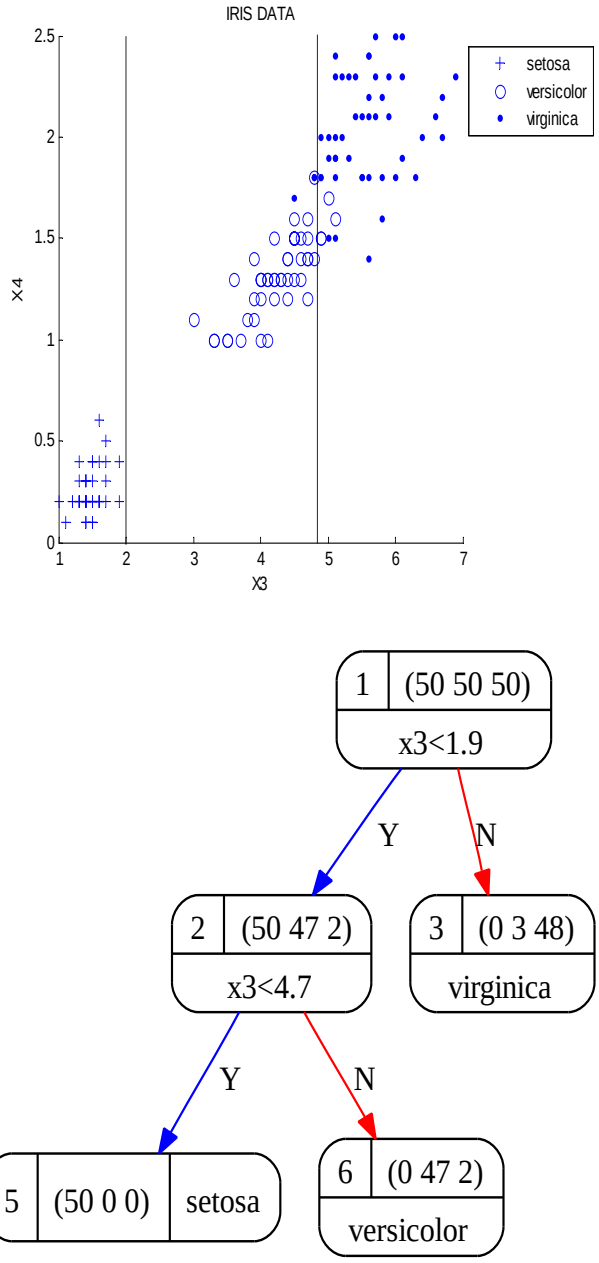

Figure 6.1: Ordinal classification Tree using Iris Data

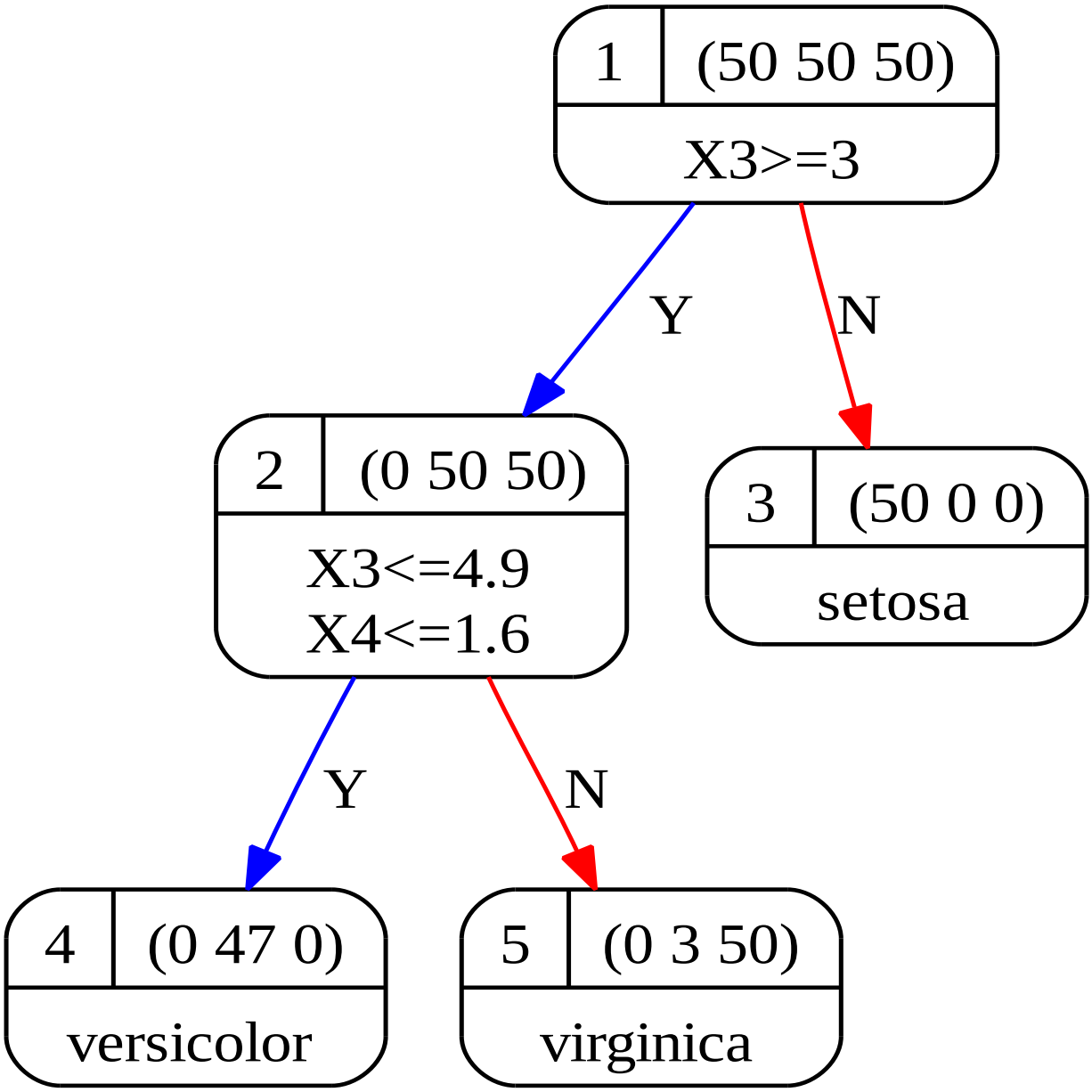

Figure 6.2: Iris Ordinal Kernel FDA Tree

6.1.2 Abalone

The abalone data is from the USI machine learning data repository donated by Sam Waugh. This information is a from the UCI repository notes.

To predict the age of abalone from physical measurements, the age of abalone is determined by cutting the shell through the cone, staining it, and counting the number of rings through a microscope – a boring and time-consuming task. Other measurements, which are easier to obtain, are used to predict the age. Further information, such as weather patterns and location (hence food availability) may be required to solve the problem.

There are 4177 observations, of which the first 3133 are used for training the decision tree and the remaining 1044 for testing. There eight predictive input values which contain 1 nominal attribute, sex.

1. X1: Sex nominal M, F, and I (infant)
2. X2: Length continuous mm Longest shell measurement
3. X3: Diameter continuous mm perpendicular to length
4. X4: Height continuous mm with meat in shell
5. X5: Whole weight continuous grams whole abalone
6. X6: Shucked weight continuous grams weight of meat

Table 6.1: Fisher Iris Kernelized Search Results

ICOMP	Kernel	CoVar	Variables	Nodes	RBF Fit	KER Fit	Tree Fit
1.46	RBF	ME	3,4	3	0.98	0.98	0.98
1.52	RBF	MLE	4	3	0.96	0.67	0.96
1.52	RBF	ME	4	3	0.96	0.67	0.96
1.57	RBF	ME	2,4	3	0.96	0.73	0.96
1.60	CAUCHY	ME	2,3,4	6	0.99	0.67	0.99
1.66	RBF	MLEEB	2,4	3	0.96	0.72	0.96
1.84	RBF	MLEEB	1,4	3	0.95	0.82	0.95
1.98	CAUCHY	NONE	1,2,4	4	0.97	0.33	0.97

7. X7: Viscera weight continuous grams gut weight (after bleeding)

8. X8: Shell weight continuous grams after being dried

The value to predict is the number of rings which gives the age in years. For this exercise the number of rings has been grouped into a 3-category classification problem: ring classes 1-8, 9 and 10, and 11 and greater. Reported test set classification performance (Waugh, 1995) using the same setup includes 59.2% for the C4.5 tree, 32.57% by Linear Discriminate Analysis, and 62.46% using k=5 Nearest Neighbor.

The next few steps will explore the benefits of the kernelized classification tree by introducing the search algorithm in stages. The structure of these stages can be found in Table 6.2. The structure of these stages can be found in Table 5.2. The first stage begins with a classically grown decision tree which is pruned to find the best fitting Gaussian RBF network. Stage two adds variable subset selection via the genetic algorithm. Stage three adds kernel and covariance smoother selection. Stage four extends the program algorithm by using the kernelization splitting rule.

For each stage, the training and test set model fit values are summarized in Table 6.3. The RBF fit is the percent correctly classified using a radial basis function classification network; and the Tree fit is the ordinal classification accuracy using the predicted class response at each decision tree leaf node. Of main interest is the RBF fit upon which the tree is pruned. Pruning is performed for all stages with the same ICOMP method as the genetic kernelized classification tree. The Kernel fit is a kernelized linear discriminate analysis model updated with the information provided by the ordinal classification tree. The kernel model is still considered experimental, but is provided for completeness of information.

Table 6.2: Abalone Algorithm Stage Structure

Stage	Split	Subset	Kernel/Cov Search	Kernel	Covariance
1	Single/Deviance	No	No	Gaussian	MLE
2	Single/Deviance	Yes	No	Gaussian	MLE
3	Single/Deviance	Yes	Yes	Power Exp	CSCE
4	Multi/ICOMP	Yes	Yes	Cauchy	MLE

In the first stage, the ordinal classification decision tree is constructed allowing only one variable splits using all 8 predictor variables. The classification splitting rule uses deviance which has been ordinalized as described above. For the training data set, this tree finished with an ICOMP score of 3.50, 20 leaf nodes. an RBF fit of 0.63, a Kernel fit of 0.43, and Tree fit of 0.63. Fitting the model to the test data set, the RBF fit is 0.59, the Kernel fit is 0.49, and Tree fit is equal to 0.59. These results correspond to the C4.5 Tree.

The model in stage two allows variable subset selection through the genetic algorithm. Splits are still being performed with only one variable. The kernel function is still Gaussian with a maximum likelihood covariance estimator. For the training data set, the selected tree has an ICOMP score of 3.54, 6 leaf nodes, an RBF fit of 0.57, a Kernel fit of 0.36, and Tree fit of 0.58. The subset of variables selected is X2, X3, and X8. Fitting the model to the test data set, the RBF fit is 0.57, a Kernel fit is 0.35, and Tree fit is 0.57. A slight improvement in the terms of developing a more parsimonious model with fewer variables with fewer leaf nodes, but not is accuracy of training fit.

The genetic algorithm is allowed to search over different kernels and covariance estimators in stage three. The kernel selected is the power exponential with the convex sum covariance estimator. The training classification tree has an ICOMP score of 3.50, 7 leaf nodes, an RBF fit equal to 0.58, a Kernel fit of 0.36, and Tree fit of 0.59. Five of the eight variables are selected for the subset. Fitting the model to the test data set, the RBF fit is 0.56, the Kernel fit is 0.39, and Tree fit score is equal to 0.56. This increase in the training data set is mostly to the choice of the RBF function.

Table 6.3: Abalone Algorithm Stage Results

Stage	Data	ICOMP	Nodes	Variables	RBF Fit	Kernel Fit	Tree Fit
1	Train	3.50	20	1-8	0.63	0.43	0.63
	Test				0.59	0.49	0.59
2	Train	3.54	6	2 3 8	0.57	0.36	0.58
	Test				0.57	0.35	0.57
3	Train	3.50	7	1 4 5 8	0.58	0.36	0.59
	Test				0.56	0.39	0.56
4	Train	3.47	11	2 4 6 8	0.58	0.34	0.63
	Test				0.63	0.45	0.61

Using the full search genetic algorithm using the splitting rule based on the ordinal kernelized linear discriminant analysis, Table 6.4 shows the more parsimonious model being chosen as the fitness function, ICOMP, decreases. With a fit of 61% for both the RBF network and the Classification tree itself, the tree is performing better than previous methods listed above. The RBF Fit score of 0.61 is a slight improvement over the classically train Tree Fit score of 0.59 but provides a simpler more parsimonious model. The selected kernel of the final model is Cauchy with the maximum likelihood covariance estimator. An illustration of the best fitting ordinal classification tree can be found in Figure 6.3.

Table 6.4: Abalone Kernelized Search Results

Data Set	ICOMP	Kernel	CoVar	Variables	Nodes	RBF	Ker	Tree
Train	3.47	Cauchy	MLE	2,4,6,8	11	0.58	0.34	0.63
Test						0.63	0.45	0.61
Train	3.48	Mulitquad	ME	5,6,7	8	0.56	0.34	0.60
Test						0.57	0.55	0.57
Train	3.70	Cauchy	MLE	1,2,6	9	0.54	0.34	0.54
Test						0.55	0.35	0.52
Train	3.73	Cauchy	MLEEB	1,2,4,7,8	7	0.57	0.34	0.59
Test						0.57	0.40	0.57
Train	3.80	Multiquad	MLE	1-4,7	4	0.56	0.34	0.56
Test						0.56	0.40	0.54

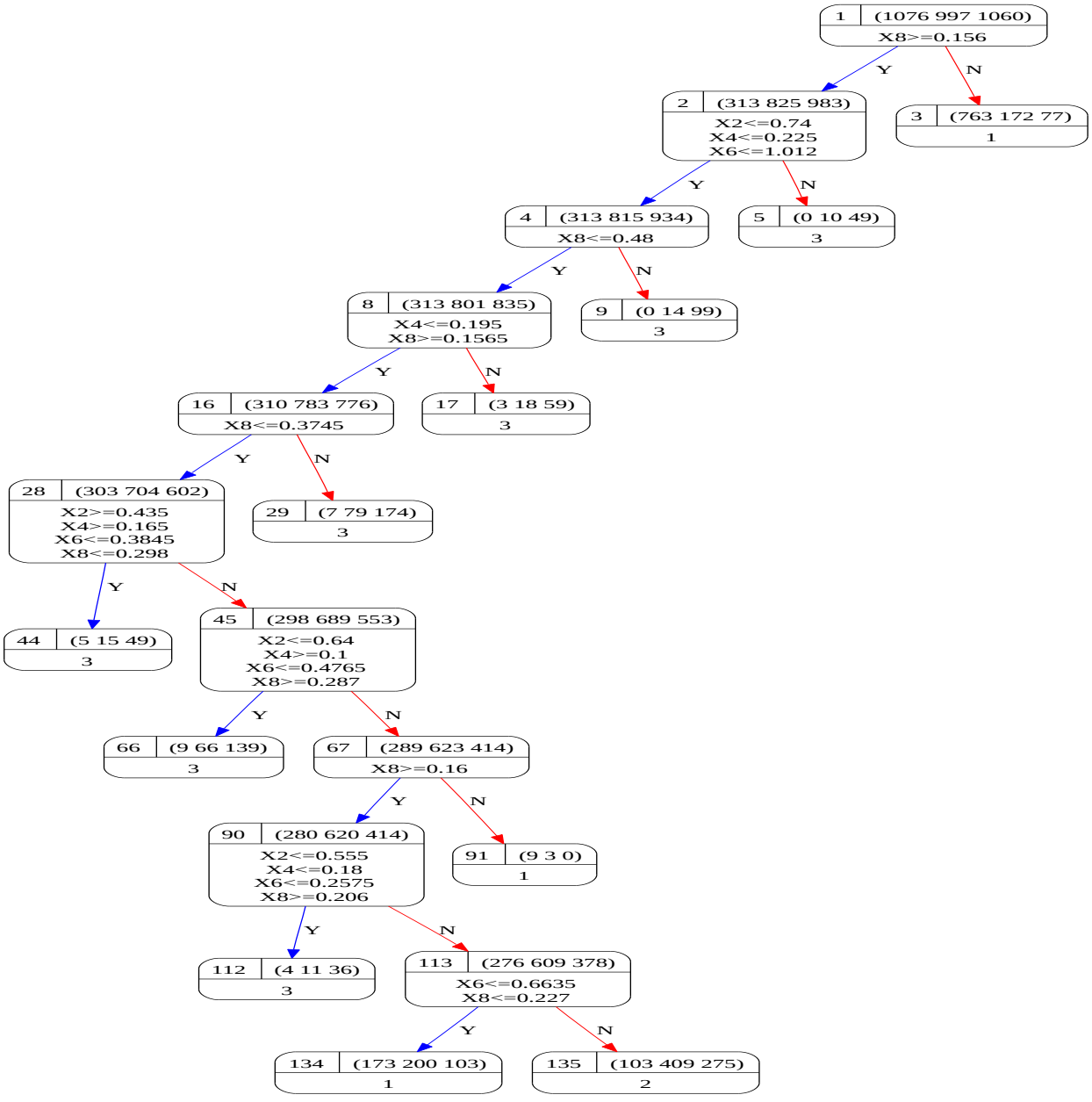

Figure 6.3: Abalone Ordinal Classification Tree

Chapter 7

Discussion and Conclusions

The principal focus of this thesis is the development of an advanced non-linear model that is both interpretable for the analyst and customer. In building a predictive model, understanding a data set's structure through the ability to visualize and interpret its complex nonlinear relationship is an important tool for knowledge extraction. This is especially true with many a customers need to understand possible underlying structures in their data along with a typical trepidation and suspicion of any analytical or statistical method or model. This dissertation explored the use of kernel based data mining techniques in combination with decision trees as a structure in the visualization of resulting radial basis function models.

Combining the Mercer Kernel techniques with decision tree allows the user to maintain a model of understanding an interpretation. Besides just the complexity of the Mercer Kernel Model, the mapping between the kernel and data space is not 1 to 1, or translation invariant. There is no insight into the data structure. The decision tree functions as a bridge between the data and kernel feature space providing a framework of understanding. The decision tree also allows a classification model to be easily extended to ordinal models as shown in chapter seven. One can use the decision tree to provide a illustration/diagram of the underlying structure of the RBF network.

The modification of the decision tree algorithm included replacing its typical greedy search with a genetic algorithm. ICOMP works well with the genetic algorithm as a fitness function providing a powerful optimization tool. Through the use of the kernel trick and the application of Dr.

Bozdogan's information criteria as a fitness function for goodness-of-fit, the genetic search algorithm handles the variable, kernel, covariance smoother selection, and best model subset selection.

Presented were regression, nominal classification, and ordinal classification methods of the kernelized ICOMP search algorithm as the decision tree is applied to radial basis function networks. In each method, the genetic algorithm was expanded in four stages to illustrate the importance of each added step to building the RBF function network.

In stage one a decision tree was grown using the standard classical splitting rule with no variable sub seelction. Shown was by first trying to find clusters the predictive variable and then fitting a radial basis function does not guarantee a fit, even with a well predicting decision tree.

Stage two introduced variable sub setting with the gain that develops with a more parsimonious model with fewer variables. Real world data sets frequently contain hundred and thousands of variables. The use of a genetic algorithm allows the user to examine only a small possible portion of all possible models to determine an optimal solution. This allows user to search of a vast and complex data space of variable combinations effectively.

The genetic algorithm in stage three is allowed to search for the best radial basis function and covariance smoother. Radial Basis function selection is shown to be important to gaining a well predicting model. The choice of the best radial function and subset selection is neither simple nor automatic. In this program, ICOMP as fitness function measures the goodness-of-fit, both in the decision tree pruning for all stages and for the splitting rule in stage four.

Stage four replaces the standard splitting rule with the kernelized ICOMP fitness function. Through the use of the kernel space, the improvements shown in stage three are often enhanced. Combined with the use of ICOMP, as in the case of the abalone tree, the ordinal classification tree is performing better than some traditional models.

The final modeling method shows itself as a viable method of feature extraction to reduce the dimensionality of the kernel methods. Using the genetic algorithm, kernel trick, and decision tree a well fitting predictive clusters are evolved. The method would provide an appropriate preprocessing step to reduce the dimension of the data to be handled by subsequent modeling. Because of its time

consuming nature even with the reduction in search time due to the genetic algorithm, it perhaps better suited to more complex problems such as the vowel data set presented in chapter six.

Future work planned will be to increase the genetic algorithm intelligence in breeding with ICOMP as a fitness function with speed being a criteria for live applications of this technique to operations and information management.

Bibliography

[1] Aeberhard, S., Coomans D., and O. de Vel. (1992). "The Classification Performance of RDA" Technical Report No. 92-01, Dept. of Computer Science and Dept. of Mathematics and Statistics, James Cook University of North Queensland.

[2] Akaike, H. (1973). Information Theory and an Extension of the Maximum Likelihood Principle, in Second International Symposium on Information theory, B.N. Petrov and F. Csaki (eds.), Budapest: Academiai Kiado, 277-281.

[3] Bozdogan, H. (1987). "Model Selection And Akaike Information Criterion (AIC) - The General-Theory And Its Analytical Extensions." Psychometrika 52(3): 345-370.

[4] Bozdogan, H. (1988). ICOMP: A new Model-Selection Criterion. Classification and Related Methods of Data Analysis, Hans H. Bock (ed.), Amsterdam: Elsevier Science Publishers B.V. (North Holland) 599-608.

[5] Bozdogan, H. (1990). "On the Information-Based Measure of Covariance Complexity and its Application to the Evaluation of Multivariate Linear Models." Communication in Statistics Theory and Methods 19(1): 221-278.

[6] Bozdogan, H. (1994). "Mixture-model cluster analysis using model selection criteria and a new informational measure of complexity." Multivariate Statistical Modeling, H. Bozdogan (Ed.), Vol. 2, pp. 69-113. Proceedings of the first US/Japan conference on the frontiers of statistical modeling: An informational approach. Dordrecht, the Netherlands: Kluwer Academic Publishers.

[7] Bozdogan, H. (1996) "Informational complexity criteria for regression models Information Theory and Statistics Section on Bayesian Stat. Science." ASA Ann. Meeting, Chicago, IL, Aug. 1996

[8] Bozdogan, H. (2000). "Akaike's information criterion and recent developments in information complexity." Journal Of Mathematical Psychology 44(1): 62-91.

[9] Bozdogan, H. (2004). Statistical data mining and knowledge discovery. Boca Raton, Fla., Chapman & Hall/CRC.

[10] Bozdogan, H. and P. Bearse (2003). "Information complexity criteria for detecting influential observations in dynamic multivariate linear models using the genetic algorithm." Journal Of Statistical Planning And Inference 114(1-2): 31-44.

[11] Bozdogan, H. and A. K. Gupta (1987). Multivariate statistical modeling and data analysis: proceedings of the Advanced Symposium on Multivariate Modeling and Data Analysis, May 15-16, 1986. Dordrecht; Boston Norwell, MA, U.S.A., D. Reidel; Sold and distributed in the U.S.A. and Canada by Kluwer Academic.

[12] Bozdogan, H. and D. M. A. Haughton (1998). "Informational complexity criteria for regression models." Computational Statistics & Data Analysis 28(1): 51-76.

[13] Bozdogan, H. and S. L. Sclove (1984). "Multi-Sample Cluster-Analysis Using Akaike's Information Criterion." Annals Of The Institute Of Statistical Mathematics 36(1): 163-180.

[14] Bozdogan, H. (2005). Information Complexity and Multivariate Learning Theory with Data Mining Applications. Forthcoming book.

[15] Breiman, L. (1984). Classification and regression trees. Belmont, Calif., Wadsworth International Group.

[16] Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J. (1984). Classification and Regression Trees. Wadsworth, Pacific Grove, CA.

[17] Broomhead, D.S. and D. lowe (1988) "Multivariate functional interpolation and adaptive networks." Complex Systems, 2:321-355

[18] Chen, M. C-F. (1976): Estimation of covariance matrices under a quadratic loss function. Research Report S-46, Department of Mathematics, SUNY at Albany, Albany, N.Y., 1-33.

[19] Cristianini, N. and J. Shawe-Taylor. (2000). An introduction to support vector machines and other kernel-based learning methods. Cambridge University Press.

[20] Dempster, A. P., N. M. Laird, et al. (1977). "Maximum Likelihood From Incomplete Data Via EM Algorithm." Journal Of The Royal Statistical Society Series B-Methodological 39(1): 1-38.

[21] Desarbo, W. S. and W. L. Cron (1988). "A Maximum-Likelihood Methodology For Clusterwise Linear-Regression." Journal Of Classification 5(2): 249-282.

[22] Desarbo, W.S. and M. Wedel (1995). "A Mixture Likelihood Approach For Generalized Linear-Models." Journal Of Classification 12(1): 21-55.

[23] Desoete, G. and W. S. Desarbo (1991). "A Latent Class Probit Model For Analyzing Pick Any/N Data." Journal Of Classification 8(1): 45-63.

[24] Deterding, D. H. (1989). "Speaker Normalization for Automatic Speech Recognition", University of Cambridge.

[25] Fukumizu, K. Bach, F. R., and M. I. Jordan. (2004). "Dimensionality Reduction for Supervised Learning with Reproducing Kernel Hilbert Spaces." Journal of Machine Learning 5:73-99.

[26] Goldberg, D. E. (1989). Genetic algorithms in search, optimization, and machine learning. Addison-Wesley, Reading, Massachusetts.

[27] Goldberg, D. E., Deb, K., and Clark, J. H. (1992). "Genetic algorithms, noise, and the sizing of populations." Complex Systems, 6:333–362.

[28] Goldberg, D. E., Deb, K., and Thierens, D. (1993). "Toward a better understanding of mixing in genetic algorithms." Journal of the Society of Instrument and Control Engineers, 32(1):10–16.

[29] Goldberg, D. E., Routlauf, Heinzl, A. (2002) . "Network Random Keys – A Tree Representation Scheme for Genetic and Evolutionary Algorithms." Evolutionary Computation 10(1): 75-97.

[30] Geurts, P., d'Alche-Buc, F., & Wehenkel, L. (2006). Kernelizing the Output of Tree-Based Methods. Unpublished.

[31] Haistie T. , R. Tibshirani, and J. Friedman. (2001) The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer. New York: 59-63.

[32] Hoerl, A. E., R. W. Kennard and K. F. Baldwin (1975). Ridge regression: Some simulations. Communications in Statistics, Part A - Theory and Methods 4, 105- 123.

[33] Hoerl, A. E. and R. W. Kennard (2000). "Ridge regression: Biased estimation for nonorthogonal problems." Technometrics 42(1): 80-86.

[34] Holland, J. H. (1975). Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence.

[35] Kubat, M. (1988) "Decision trees can initialize radial-basis function networks." IEEE Transactions on Neural Networks, 9(5):813–821.

[36] Kubat, M. and I. Ivanova. (1995) "Initialization of RBF networks with decision trees." In Proc. of the 5th Belgian-Dutch Conf. Machine Learning, BENELEARN '95: 61–70.

[37] Lawless, J.F. and Wang P. (1976) A Simulation Study of Ridge and other Regression Estimators. Commun Statist. Theory Meth., A(5):307-323.

[38] Liu, Z. and H. Bozdogan (2004) "Improving the performance of radial basis function (RBF) classification using information criteria." Statistical Data Mining and Knowledge Discovery, Chapman and Hall/CRC: 193-216.

[39] Meng, X. L. and D. vanDyk (1997). "The EM algorithm - An old folk-song sung to a fast new tune." Journal Of The Royal Statistical Society Series B-Methodological 59(3): 511-540.

[40] Morgan, J.N., Sonquist, J.A. (1963). Problems in the analysis of survey data, and a proposal. J. Amer. Statist. Assoc. 58, 415–434.

[41] Morgan, J.N., Messenger, R.C. (1973). THAID: a sequential search program for the analysis of nominal scale dependent variables. Institute for Social Research, University of Michigan, Ann Arbor, MI.

[42] Moody J. and C.J. Darken (1989) "Fast Learning in Networks of Locally Tuned Processing Units." Neural Computation, 1(2): 281-294

[43] Orr, M. J. L. (1995). "Regularization In The Selection Of Radial Basis Function Centers." Neural Computation 7(3): 606-623.

[44] Orr, M. J. L. (1995) "Local smoothing of radial basis function networks." In International Symposium on Artificial Neural Networks, Hsinchu, Taiwan, 1995.

[45] Orr, M. J. L. (1996) "Introduction to radial basis function networks." Technical report, Institute for Adaptive and Neural Computation, Division of Informatics, Edinburgh University.

[46] Orr, M. J. L. (1988) "An EM algorithm for regularized radial basis function networks." In International Conference on Neural Networks and Brain, Beijing, China.

[47] Press, S.J. (1975): Estimation of a normal covariance matrix. University of British Columbia.

[48] Rasmussen, C. E. (1996). Evaluation of Gaussian Processes and Other Methods for Non-linear Regression. PhD thesis, Department of Computer Science, University of Toronto, Toronto, Ontario.

[49] Rasmussen, C. E. (2002) Reduced rank Gaussian process learning. Technical report, Gatsby Computational Neuroscience Unit.

[50] Rasmussen, C. E. and Joaquin Quinonero-Candala. (2005). "A unifying view of sparse approximate gaussian process regression." Journal of Machine Learning Research 6:1939-1959.

[51] Rasmussen, C. E. and K. I. Williams. (2006) Gaussian Processes for Machine Learning. The MIT press.

[52] Scholkopf, B., S. Mika, et al. (1999). "Input space versus feature space in kernel-based methods." IEEE Transactions On Neural Networks 10(5): 1000-1017.

[53] Scholkopf, B., A. Smola, et al. (1998). "Nonlinear component analysis as a kernel eigenvalue problem." Neural Computation 10(5): 1299-1319.

[54] Schölkopf, B. and A. J. Smola (2002). Learning with kernels: support vector machines, regularization, optimization, and beyond. Cambridge, Mass., MIT Press.

[55] Sclove, S.L. (1973). "Least squares problem with random regression coefficients", Technical Report No.87, IMSSS, Stanford University,

[56] Seeger, M., Teh Y.W., and M. I. Jordan. (2004) Semiparametric latent factor models. Technical report, University of California at Berkeley. See www.cs.berkeley.edu/mseeger.

[57] Shawe-Taylor, J. and N. Cristianini (2004). Kernel methods and pattern analysis. Cambridge University Press.

[58] Shurygin, A. M. (1983): The linear combination of the simplest discriminator and Fisher's one. Applied Statistics, Nauka (ed.), Moscow, 144-158.

[59] Smola, A. J. and Bartlett, P. L. (2001). Sparse greedy Gaussian process regression. In Leen, T. K., Dietterich, T. G., and Tresp, V., editors, Advances in Neural Information Processing Systems 13, pages 619-625, Cambridge, Massachusetts. MIT Press.

[60] Tikhonov, A. N. and V. I. F. A. F. Arsenin (1977). Solutions of ill-posed problems. Washington New York, Winston; distributed solely by Halsted Press.

[61] Tipping, M. E. (2000). "The relevance vector machine." Advances in Neural Information Processing Systems 12.349-355.

[62] Tipping, M. E. (2001). "Sparse Bayesian learning and the relevance vector machine." Journal of Machine Learning Research 1:211-244.

[63] Tsoukalas, L. H. and R. E. Uhrig (1997). Fuzzy and neural approaches in engineering. New York, Wiley.

[64] Urmanov, A. M., A. V. Gribok, et al. (2002). "Information complexity-based regularization parameter selection for solution of ill conditioned inverse problems." Inverse Problems 18(2): L1-L9.

[65] Vapnik, V. N. (2000). The nature of statistical learning theory. New York, Springer.

[66] Waugh, S. (1995). Extending and benchmarking Cascade-Correlation, PhD thesis, Computer Science Department, University of Tasmania.

[67] Williams, C. (1997). Prediction with Gaussian Processes: From linear regression to linear prediction and beyond." Technical Report NCRG 97 012. Neural Computing Research Group. Dept of Computer Science and Applied Mathematics. Aston University, Birmingham, United Kingdom.

[68] Woodbury, M. A. (1950). inverting modified matrices. *Statist. Res Group, Mem. Rep. No. 42.* Princeton University, Princeton, New Jersey.

[69] Written, I.H. and E. Frank (2000). Data mining: practical machine learning tools and techniques with Java implementation. San Diego, CA, Academic Press.

Appendices

Appendix A

Notational Conventions

Scalars are represented by italicized letters such as y or λ.

Vectors are represented by bold lower case letters such as $\mathbf{x}$ and λ. The first component of a vector $\mathbf{x}$ is a scalar x_1. Vectors are single-column matrices. For example, if $\mathbf{x}$ has n components then

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

Matrices are represented by bold capital letters such that the entry in the i-th row and j-th column of $\mathbf{X}$ is X_{ij}. If $\mathbf{X}$ has n rows and p columns then

$$\mathbf{X} = \begin{bmatrix} X_{11} & X_{12} & \cdots & X_{1p} \\ X_{21} & X_{22} & \cdots & X_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ X_{n1} & X_{n2} & \cdots & X_{np} \end{bmatrix}$$

When needed commas will be supplied to prevent confusion as in the case $X_{1,11}$.

The transpose of a matrix is represented by an upper subscript T. So, if $\mathbf{Y} = \mathbf{X}^T$ then $Y_{ji} = X_{ij}$.

The m-dimensional identity matrix , a square matrix with diagonal entries of 1 and 0 elsewhere, is written as $\mathbf{I}_m$.

The inverse of a square matrix $\mathbf{X}$ is written $\mathbf{X}^{-1}$where

$$\mathbf{X}\mathbf{X}^{-1} = \mathbf{X}^{-1}\mathbf{X} = \mathbf{I}_m$$

Estimated or uncertain values are distinguished by the use of the hat symbol. For example, $\widehat{\lambda}$ is an estimated value for λ.

Appendix B

Genetic Algorithm ICOMP Tree Program

The Genetic Algorithm ICOMP Decision Tree program sets up and runs genetic algorithm based variable subset search utilizing a kernelized ICOMP decision tree. This program was developed and runs on Matlab version 7.

Input is in the form of a data input file which must be MAT. Data is assumed to be in matrix form with the following variables

- xdata: nxp numeric predictor matrix data set
- ydata: nx1 response array
- cdata: 1xp array defining x variable type 0: continuous 1: nominal

Output is a text file storing the following information for each best tree structure as found by the genetic algorithm in succession.

For a regression tree:

- Model Variables
- Kernel

- Covariance
- Number of End Nodes
- Model ICOMP
- Model Fit
- Kernel ICOMP
- Kernel Fit
- Fit using Node average

A classification tree does not include the node average fit, but instead reports:

- ICOMP Classification Model
- Fit using Node Classification

The text output concludes by showing the structure of the best fitting decision tree. An example output of a classification output is as follows:

classification ordinal

Model, Kernel, Cov, Nodes, icompmodel, fitmodel, icompker, fitker, icompcl, fitcl

———————————————————————————-

1 2 3 4, RBF, MLE, 5, 2.54, 0.99, 2.45, 0.99, 5.91, 0.99

1 2 4, POWEXP, CSCE, 9, 3.03, 1.00, 3.03, 1.00, 4.79, 1.00

2 3 4, POWEXP, ME, 3, 1.50, 0.97, 1.61, 0.97, 4.71, 0.97

1 2 3, POWEXP, ME, 3, 1.95, 0.94, 1.22, 0.82, 3.68, 0.94

Node 1: x1 <= 5.7 x2 >= 2.6 x3 <= 3 (50 50 50)

Node 2: setosa (48 0 0)

Node 3: x3 <= 4.8 (2 50 50)

Node 4: versicolor (2 46 3)

Node 5: virginica (0 4 47)

In addition, for each tree mentioned in the text output, a MAT file is generated. The file is labeled as NAME_sub_KERNEL_COV_VARIABLES. The final selected model will not contain the "_sub_" identifier.

Examples of file output names:

- irisnom_RBF_MLE_0010.mat
- irisnom_sub_CAUCHY_ME_1011.mat
- irisnom_sub_RBF_NONE_0010.mat

The MAT file includes a cell structure array label Treebest describing the tree structure. Its structure is defined with the following fields:

Structure Field	Description ([C] classification trees only)
.method	Method (classification,regression,kfda)
.rbf	Kernel Function
.smooth	Covariance Smoother
.node	Node number
.parent	Parent node number
.class	Class assignment for points in node if treated as a leaf
.var	Column j of X matrix to be split, or 0 for a leaf node, or -j to treat column j as categorical
.op	Defines split operator Continuous 0: <= 1:>= Discrete 0:= 1:!=
.cut	Cutoff value for split (Xj<cutoff goes to left child node), or index into catsplit if var is negative
.max	Maximum data point in Node

Structure Field	Description ([C] classification trees only)
.min	Minimum data point in Node
.cen	Node Center
.h	Node smoothing parameter
.rbfpar3	Third (optional) parameter for Kernel Function (example : Power Exponential)
.children	Matrix of child nodes (2 cols, 1st is left child)
.nodeprob	Probability p(t) for this node
.nodeerr	Resubstitution error estimate r(t) for this node
.risk	R(t) = p(t)*r(t)
.nodesize	Number of points at this node
.npred	Number of predictors
.catcols	Defines if predictor variable is 0:Continuous or 1: Discrete
.prunelist	List of indices that define pruned subtrees. One entry per node. If prunelist(j)=k then, at the kth level of pruning, the jth node becomes a leaf (or drops off the tree if its parent also gets pruned).
.alpha	Vector of complexity parameters for each pruning cut
.ntermnodes	Vector of terminal node counts for each pruning cut
.catsplit	Call array for categorical splits, left categories in column 1 and right categories in column 2
.classprob	[C] Vector of class probabilities
.classname	[C] Names of each class
.classcount	[C] Count of members of each class
.nclasses	[C] Number of classes

Also included is a binary vector Treemodel that describes which of the original variables have been selected for the model. For example, if the the best model has choses only variable 3 out of four

potential candidates, Treemodel = [0 0 1 0]. It is important to note that the final Treebest structure only recognizes that one variable is needed. Also note that the output tree would label the output splits as x1, not x3. When entering data into a function, the format xdata(:,find(Treemodel)) can be applied to subset the data.

B.1 Copyright Information

B.2 Programs included by other authors

B.2.1 Genetic Algorithm Optimization Toolbox

The Genetic Algorithm Optimization Toolbox, GAOT, implements simulated evolution in the Matlab environment using both binary and real representations. The toolbox also includes ordered base representation. GAOT is described in the technical paper "A Genetic Algorithm for Function Optimization: A Matlab Implementation" by Chris Houck, Jeff Joines, and Mike Kay, NCSU-IE TR 95-09, 1995. Its implementation is very flexible in the genetic operators, selection functions, termination functions as well as the evaluation functions that can be used. Due to its flexibility, GAOT has been embedded into to authors program.

While included in its original form with this program, one can also download the entire toolbox can be download either as a compressed tar archive (GAOT.tar.gz) or a ZIP file (GAOT.zip) via anonymous ftp from the directory ftp://ftp.eos.ncsu.edu/pub/simul/GAOT as well as other GA related papers. These downloads include the postscript and dvi versions of the companion paper.

B.2.2 Pseudo Inverse Matrices Algorithm

A pseudo inverse matrices algorithm based on a full rank Cholesky factorization as described in the paper Fast Computation of Moore-Penrose Inverse Matrices by Courrieu, P., published in Neural Information Processing: Letters and Reviews, 8(2) 2005 is utilized by this program. The resulting pseudo inverse matrices are similar to those provided by other algorithms. However the computation time is substantially shorter, particularly for large systems.

This program is required to solve large least square systems in order to obtain weights. Moore-Penrose inverse matrices allow for solving such systems, even with rank deficiency, and they provide minimum-norm vectors of synaptic weights, which contribute to the regularization of the input-output mapping. Fast and accurate algorithms for computing Moore-Penrose inverse matrices are of interest when performing kernel methods. Courrieu, in this paper, proposes an algorithm based on a full rank Cholesky factorization.

B.2.3 Recommended Programs

In order to better visualize the tree structure, included is Matlab code to layout and draw graphs. The plot code is written in Matlab, while the final layout is obtained by interfacing with GraphViz (AT&T). The Matlab function is gatree_to_dot.m. Using the gatree_to_dot(Treebest,'fileout','appendtree.dot','na
which will generate the dot appendtree.dot that will be used by GraphViz to generate the decision tree found in Figure B.1. A prune level of zero is an unpruned full tree. The maximum pruning allowed can be determined from Treebest cell array structure, max(Treebest.prunelist). Note that the variable labeling is also corrected using the names input variable.

If you want to use this code you need to install the GraphViz. GraphViz can be found at http://www.research.att.com/sw/tools/GraphViz. Also for the initial Matlab code inspiration and other ideas visit http://www.eecs.harvard.edu/~pesha/Public/DATA.html, Dr. Leon Peshkin website.

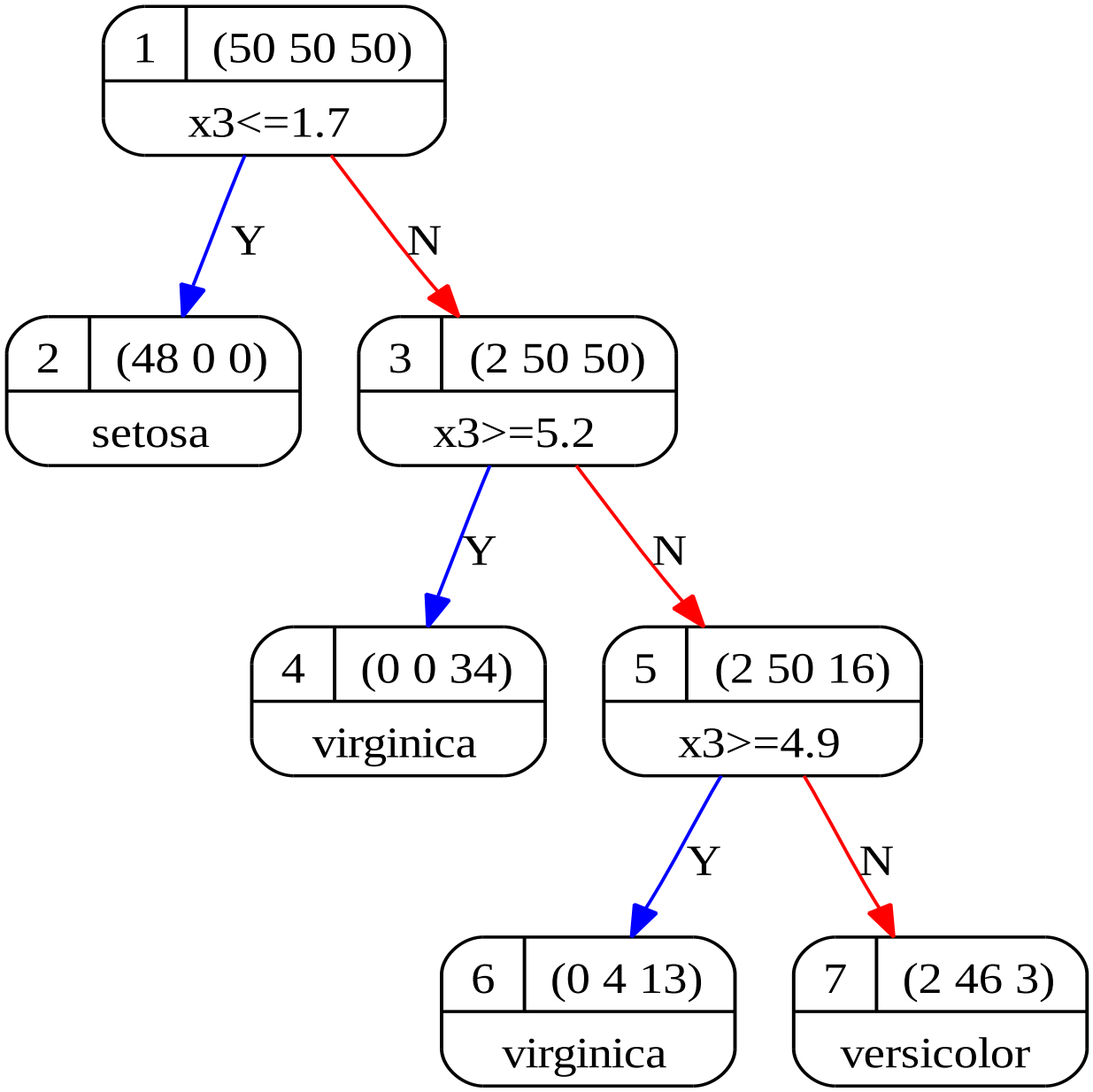

Figure B.1: GraphViz Decision Tree Output

B.2.4 List Of Programs and Functions

A list of the Matlab programs are lsisted below.

Type	Program	Description
Main Interface	gaicomptree.m	Sets up a runs Genetic algorithm Based variable subset search utilizing a Kernelized ICOMP Decision Tree.
	gatree_to_dot.m	Creates a GraphViz (AT&T) dot format file representing the tree structure, TREE, produced by the GATREE function
Covariance	covsmooth.m	Returns Smoothed Covariance estimate. Data is assumed to be in nxp matrix form. If data is in nx1 column vector the program defaults to MLE cov functions. Smoothing Methods are Maximum Likelihood, Maximum likelihood / empirical Bayes, Maximum entropy - Fiebig 1982, Maximum entropy / empirical Bayes, Stipulated ridge - Shurygin 1983, Stipulated diagonal - Shurygin 1983, Convex sum - Chen 1976, and Ledoit - Ledoit & Wolf 2003
	covspd.m	Test if a matrix (data) is a % symmetric positive definite covariance matrix. Returns a logical Flag (F)

Type	Program	Description
	geninv.m	Pseudo inverse matrices algorithm based on a full rank Cholesky factorization as described in the paper Fast Computation of Moore-Penrose Inverse Matrices by P. Courrieu published in Neural Information Processing: Letters and Reviews, 8 (2) 2005
	Ledoit.m	This function computes the covariance matrix estimator introduced by Olivier Ledoit in "Portfolio Selection: Improved Covariance Matrix Estimation" (Job market paper, November 1994, reproduced by the UCLA % Finance Department as Working Paper #5-96) Program was written by Olivier Ledoit (Ledoit@ucla.edu) on 1/29/1996.
Genetic Algorithm Functions	b2i.m	Returns the integer of a binary representation
	i2b.m	Return the binary representation of an integer number given the number of bits to represent each variable
	gap.m	Simple Genetic algorithm through n individuals using simple crossover and mutation

Type	Program	Description
Genetic Algorithm Optimization Toolbox	gaotv5.zip	This zip file contains the complete Genetic Algorithm Optimization Toolbox (GAOT) for Matlab 5. Functions called (not embedded in the main program are listed below.
	b2f.m	Binary to Float conversion used by GAOT
	calcbits.m	Binary precision function used by GAOT
	f2b.m	Float to Binary conversion used by GAOT
	maxGenTerm.m	Termination function Used by GAOT
	normGeomSelect.m	Selection function Used by GAOT
	optMaxGenTerm.m	Termination function Used by GAOT
	optMaxGenTerm2.m	Modified Termination function Used by GAOT
	parse.m	Parse blank separated names used by GAOT
	simpleXover.m	Operator for the Algorithm Used by GAOT
	unifMutation.m	Operator for the Algorithm Used by GAOT
Kernel Functions	ker_ed.m	Computes the squared Euclidean Distance Matrix between all rows of data1 and data2 matrices. If data2 is empty, data2==data1 is assumed
	ker_matrix.m	Computes a kernel matrix between the row-vectors of data1 and data2 If data2 is empty, data2==data1 is assumed. Kernel Methods are Cauchy, Multi Quadratic, Inverse Multi Quadratic, Multivariate exponential rbf, Power exponential, and Gaussian rbf

Type	Program	Description
	ker_md.m	Computes the Mahalanobis Distance Matrix between all rows of data1 and data2. If data2 is empty, data2==data1 is assumed.
	ker_norm.m	Normalizes a nxn kernal matrix
Decision Tree	gaclassfit.m	Fits a classification model by first removing the expected % class probability, P, and then applying a Kernelized Flexible Discriminant Analysis
	gaicompcl.m	Fits a classification tree, TREE, produced by the GATREE function, and a matrix X of predictor values and y classification values to produce a vector id of fits and ICOMP scores defined over a vector of defines pruning levels. Must include a pruning sequence as created by the GATREE
	gaicompreg.m	Fits a regression tree, TREE, produced by the GATREE function, and a matrix X of predictor values and y response values to produce a vector id of fits and ICOMP scores defined over a vector of defines pruning levels. Must include a pruning sequence as created by the GATREE.
	gam.m	Assigns model fit based on minimum distance
	garegfit.m	Fits Gaussian Process Regression Model
	gass.m	Given an nxn kernel matrix K and a group membership matrix C, GASS returns the variance S where S = SUMi || x - ci || /n

Type	Program	Description
	gatree.m	GATREE Fit a tree-based ICOMP model for classification or regression using Mercer Kernel techniques. GATREE creates a decision tree T for predicting response Y as a function of predictors X. X is an N-by-M matrix of predictor values. Y is either a vector of N response values (for regression), or a character array or cell array of strings containing N class names (for classification). Main program called by GAICOMPTREE.M
	gatreefit.m	Fits a decision tree, TREE, produced by the GATREE function, and a matrix X of predictor values to produce a vector id of predicted response values. For a regression tree, id is the fitted average response value for a point having the predictor values X(j,:). For a classification tree, id is the class number to which the tree would assign the point with data X. Must include a % pruning sequence as created by the GATREE

VITA

J. Michael Lanning wandered away from his genetic talent of working with computers to become a chemical engineer. During his travels, he decided to pursue applied statistics for its ability to quantify and understand processes of varying types. Opening a path to be able to provide technical support and instruction for a variety of customers and fields of study, he decided to pursue a PhD. Enjoying the fun of instructing a captive audience, he has become spoiled by the life of living and working in an academia environment. His interests include in application of statistical methodologies in the areas of data mining and information technology. As such, he has returned to the family business of computer science as he develops and applies statistical techniques.