DIAGNOSTICS AND EXTRAPOLATION IN MACHINE LEARNING

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF STATISTICS
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Giles Hooker

July 2004

© Copyright 2004 by Giles Hooker

All Rights Reserved

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Jerome Friedman
(Principal Advisor)

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Trevor Hastie

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Art Owen

Approved for the University Committee on Graduate Studies.

Abstract

The subject of this thesis is the interaction between the problems of diagnostics and extrapolation in machine learning.

I present a suite of tools for understanding high dimensional prediction functions that are based on the Functional ANOVA decomposition and argue that these are optimal in an idealized setting. I then show that they can be distorted to an arbitrary extent if the predictor space contains large regions of extrapolation.

This thesis gives a criterion of extrapolation and details tree-based methods to evaluate it. This methodology provides a comprehensible representation of the distribution of training data and a diagnostic for functional behavior in regions of low data density. I then discuss the issue of making predictions at points of extrapolation. I suggest a strategy for stabilizing a general learning algorithm away from training data that is motivated by a Bayesian heuristic not unlike ridge regression and which bears some resemblance to Kriging.

Finally, I advocate a modification to the Functional ANOVA that uses this estimate to avoid the effects of bad extrapolation while retaining many of the useful properties of the decomposition.

All the ideas in this work are designed to be fully general and compatible with any machine learning algorithm.

Acknowledgments

I owe too many thanks to too many people to properly acknowlege in a single page.

My advisor, Jerry Friedman, has been a constantly-available advice and encouragement that has extended even to my critique of his own work.

The research group run by Trevor Hastie and Rob Tibshirani has been another excellent source of academic support and inspiration. Art Owen has had thoughtful comments and pointed out connections between this work and other research.

Saharon Rossett provided the initial idea for DARE and many great discussions. Bogdan Popescu pointed out the credit agency example.

Many friends and colleagues in Statistics have been sources of help and advice. Matthew Finkelman and Victoria Stodden deserve particular credit for more proof reading than I had any right to ask them for.

Financially, I have been supported while at Stanford by a Fulbright Scholarship, sponsored by the Victorian Workcover Authority.

My friends and family, now in more places around the world than I can keep track of, have always been a bottomless source of encouragement and goodwill. My parents and sister deserve particular thanks for enduring the inevitable grad. student angst.

Finally, thanks must go to the Stanford Outing Club and the National Parks Service with whom I have spent just as many weekends as I could manage. I would like to claim the Sierra Nevada as a source of inspiration, but I suspect it only provided distraction. Either way it was a lot of fun.

Contents

List of Figures

Chapter 1

Introduction

The rise of computational power in the past few decades has lead to a huge increase in the sophistication of mathematical models that are used to describe data. This has been enabled not only by the speed and memory increases that allow much more complicated search strategies but also by the massive increase in the amount and complexity of stored data. Automated data collection has allowed many private and public sources to accumulate data warehouses containing thousands and sometimes millions of records each made up of tens to thousands of measurements. This data complexity has necessitated the development new statistical techniques for modeling and describing data, a task which has largely been undertaken under the moniker of machine learning.

One of the central concerns that machine learning has is in making predictions. The goal is to build a mathematical model to predict responses on future data based on some measured predictors. Assume that we have a d-dimensional predictor vector:

$$x = (x_{\{1\}}, \ldots, x_{\{d\}})$$

for which there is a response:

$$y = f(x) + \epsilon$$

for some unknown function f and a random component ϵ distributed according to some distribution $P(\epsilon)$. "Good" prediction then makes some loss criteria $L(\hat{y}, y)$ – the loss from predicting $\hat{y}$ if the truth turns out to be y – small. The optimal predictive function is

$$F(x) = \underset{G(x)}{\operatorname{argmin}} E_{y,x} L(G(x), y). \tag{1.1}$$

Typically, L is designed to correspond to $P(\epsilon)$ so that $F(x) = f(x)$, but this is not necessary. This model can be employed in a regression setting, with $F(x)$ being real valued and ϵ typically assumed to be normally distributed. It also fits a classification setting, with $F(x)$ giving a class label and ϵ multinomially distributed. This thesis is primarily concerned with the regression problem. I provide a cursory overview of techniques to extend these techniques to classification.

The problem posed by machine learning is then to develop an estimate of F. We are given access to a data set

$$D = \{x_{i,1}, \ldots, x_{i,d}, y_i\}_{i=1}^{N}$$

of training examples and are asked to provide a function that has good predictive accuracy on future examples. A host of methods have been suggested for both the classification and regression problems, each with their own particular model assumptions and methods for optimizing (1.1). There is a large body of on-going research into this problem and the many issues it entails. This thesis, as much as is possible, takes a generalist approach and tries to avoid discussion of the technicalities of specific methods beyond the properties listed below as holding across a large body of research.

The focus of the larger part of research in machine learning has been in developing flexible models for learning such prediction functions. Being a universal approximator - able to approximate any reasonable function arbitrarily well - is usually regarded as a desirable property for a learning methodology. This approach makes good sense in a setting where very little is known about system dynamics; placing constraints on the set of prediction functions without prior knowledge that these are reasonable can produce functions that perform very poorly when the true target badly violates those constraints.

There are two negative consequences of this flexibility with which this thesis is concerned. These are in the areas of interpretation and extrapolation. Both of these are secondary concerns when the central criterion for success is predictive accuracy. Nonetheless, for real-world applications both can be important. We naturally desire to understand a system, and doing so may allow us to produce better models for it. Moreover, while extrapolation may seem irrelevant for predictive accuracy on well behaved data, in practice delinquency is common in real-world data and there are times when we deliberately wish to evaluate a function at points of extrapolation.

Diagnostics

Once machine learning has been employed in a prediction task, and a prediction function has been produced and performs satisfactorily, a natural question to ask is "What is it doing?" Understanding the dynamics of systems is, after all, the central purpose of science and fundamental to human curiosity. There are also good practical reasons for desiring interpretation. Firstly, interpretation represents a check on system dynamics, or our intuition - does what this function is doing look reasonable? If not, either our intuition or the function need to be changed. Secondly, it allows us to understand specific predictions - banks, for instance, are required to give reasons for rejecting loan applications. Finally, gaining an understanding of the dynamics of

a system may allow us to design better models for it and improve predictive accuracy.

Few of the procedures that have been suggested in the machine learning literature provide predictor functions that are readily understandable. There are some which purport to provide aids to understanding. Tree-based methods have a diagrammatic representation that can be easily followed. Neural Networks also have a graphical representation, although this is harder to interpret. Many other methods; nearest neighbors, radial basis functions and Support Vector Machines, for example, do not provide diagnostic insight. Moreover, the diagnostic tools for both trees and neural networks are highly variable under perturbations of the training data, making interpretation based on them problematic [2]. Interpretation also becomes more difficult as both tree glyphs and Neural Network diagrams become more complicated. Ensemble methods are often used in conjunction with trees (c.f. [10], [2] and [8]), and while these do often increase predictive accuracy, they lose the interpretability of the single glyph.

While it may be possible to tailor diagnostic tools for specific models, this work is concerned with the problem of making predictions for a general "black box" function. I assume that we know nothing about the structure of this function, but are given access to the data that was used to train it, and that for any point of predictor space, we can request the output of the function at that point. I am interested in a geometric interpretation of the function in terms of predictor variables. It is natural to ask which predictors make a large difference in the values of the prediction. We also want to know what that difference looks like as the predictor is changed. Such first order statistics effectively treat the function in question as being additive, and it is worth asking the same questions for pairs, triples and higher combinations of variables.

The most common tool for providing these diagnostics is the Functional ANOVA decomposition, first introduced in [16]. The display of functional components based on the Functional ANOVA has already been developed in [28] and [17]. We may

also be interested in evaluating the importance of these components in providing a good approximation to a prediction function. This has been the subject of part of the research in this thesis and I present a measure of importance and argue that it is natural in these circumstances. These diagnostics must be efficiently calculable and I give a method for doing this and graphical displays of the results.

Extrapolation

The second problem that this thesis addresses is that of extrapolation. Extrapolation occurs when predictions are made at points which are far away from known examples in predictor space. This is not likely to occur in a well-behaved low dimensional setting. However, in a machine learning context, extrapolation can be relatively common. A first reason for this is is simply dimensionality – typically, the number of points required to cover a region increases exponentially with the dimension of that region, so that data can become very sparse as the number of predictors increases.

In a real world context, training data sets are also very rarely representative of the distribution of future data. To begin with, the systems that generate such data are rarely static and the general distribution of the predictor variables can shift; income shifting with inflation is an example of this, more general sociological trends – type of employment, number of children and so forth – can also be evident. Often, the set of predictor variables contain measurements under the control of the agency doing the data mining. In previous data sets these have been chosen in an *a priori* manner based on predictors which the agency can't control. The purpose of the data mining, however, may precisely be to optimize over a set of actions that can be taken, including those not previously tried for the current example. This amounts to deliberately making evaluations at points of extrapolation. Finally, typographical errors and mistakes are common in many data sets and can result in points that are, mistakenly, far away from any known data point.

A first issue, therefore, is to be able to diagnose points of extrapolation. This diagnosis can then be put to various uses. An increasing amount of extrapolation, for example, can provide an indication of concept drift and may suggest that a model may need to be re-trained. Points at extrapolation can also be classified as such and given a "don't know" label, or at least a low confidence score. These might warrant further investigation to detect errors in the data set. In this thesis, I develop a tree-based method that has the additional advantage of providing a graphical description of the distribution of the data and a diagnostic tool for investigating the behavior of a prediction function at points of extrapolation.

Making predictions at points of extrapolation is a problem when flexible predictors are used. Philosophically, a flexible approximator has few model assumptions and we therefore have no reason to believe the predictions that it makes when not influenced by data. Far away from data points, the predictions made by such methods are influenced more by the particular details of their learning scheme than by the training data and this can lead to a number of consequences. The first of these is that for models that do not extrapolate as constants, it is possible to make predictions that are very different from the responses in the training data, even at superficially reasonable places. The second is that these predictions, even when reasonable, can be highly variable under perturbations of the training data. This also decreases the trust that should be put on predictions in those regions.

There are occasions when it is necessary to make predictions at points of extrapolation. This is the case, for example, in the commercial settings mentioned above, and particularly when a choice of actions is available, some of which are extrapolatory. I present a heuristic that suggests that predictions should be shrunk toward a stable null model which gives reasonable, if not very accurate, predictions everywhere. The simplest of these is a constant which I argue is the most natural model. More complicated models can be used, however, if more prior information is available – for example, if we know that a response should be monotone increasing with some value

of the predictor variable. This shrinkage has the benefit of both ensuring that reasonable predictions are made and of stabilizing predictions at points of extrapolation. I show that this shrinkage can be achieved with any machine learning algorithm by adding data to influence it toward our chosen base model.

Interactions

Extrapolation is of particular importance for diagnostics, although the connection between these two has largely been overlooked. The Functional ANOVA, along with most other diagnostic tools, ignores the distribution of the training data. The Functional ANOVA is itself defined specifically for a product measure that fills the range[1] of the distribution with points even when much of that space is empty of training points.

If we are cautious about making predictions in regions of low data density, we should be even more concerned about allowing predictions in those places to influence our diagnostics. We are, after all, interested in understanding the dynamics of the system that is being modeled and using predictions at extrapolation distorts this with artifacts from the specific modeling procedure used. Indeed, I show that it is possible to produce arbitrary distortion of ANOVA-based diagnostic tools in realistic settings.

The final concern of this thesis is a generalization of the Functional ANOVA to non-product measures. I show that this can be achieved while losing relatively few of the desirable properties of that decomposition, and none that are important for the purposes of machine learning diagnostics. Once the Functional ANOVA can be made to accept a general weight function, we can restrict the calculation of functional effects away from regions of extrapolation and (hopefully) achieve a more realistic picture of system dynamics. The formulation I give for this new decomposition makes it

[1] "Range" is used in a multi-dimensional setting throughout this thesis to denote the hypercube bounding the data.

challenging to estimate and I suggest novel methods to do this. Using these estimates, all the diagnostic tools developed and expounded in the earlier parts of the thesis can then be employed while avoiding evaluating the function at points of extrapolation.

Summary of Thesis

This thesis follows the structure broadly outlined in this introduction. I begin by reviewing existing work on diagnostics in Chapter 2. I specifically focus on an exposition of the Functional ANOVA, its uses in providing displays of effects and some variations on it that exist in the literature. Chapter 3 extends displays of Functional ANOVA components to an evaluation of their importance and present a graphical aid to understanding these scores. I argue that the importance scores I propose are the natural ones to use in this context and I present an efficient method for producing a graphical description of the ANOVA structure of a generic black box function.

Chapter 4 shift the focus to extrapolation. I describe the problem of extrapolation and the repercussions of ignoring the issue. I also show that extrapolation can severely distort the interpretational diagnostics developed in Chapters 2 and 3. In order to combat this problem, Chapter 5 presents a diagnostic of extrapolation, which can be thought of as an outlier-detection device. This is based on trees and I take advantage of the tree glyph to explore the extent of bad extrapolation in a function. Chapter 6 is concerned with the problem of making predictions at points of extrapolation. I propose a shrinkage of predicted values that is proportional to the data density at the point of prediction. I show how this can be implemented stochastically with any machine learning algorithm that takes observation weights.

The interaction between diagnostics and extrapolation is the subject of Chapter 7. Here I propose a generalization of the Functional ANOVA, hitherto defined only on product measures. I present a list of desirable properties for functional effects

and show that this generalized Functional ANOVA provides them. Estimating such generalized effects is non-trivial and I introduce a novel estimation technique. This generalized decomposition can then be employed to provide versions of the displays and diagnostics from Chapters 2 and 3 that are robust to bad extrapolation. Chapter 8 provides an over-view of the work, as yet unanswered questions and further directions in research.

Chapter 2

Diagnostics and the Functional ANOVA

The first aim of this thesis is to develop diagnostic tools to aid us in understanding the behavior of a high dimensional function. Many machine learning procedures produce functions which cannot be written down as a readily understandable formula; any algebraic representation of the function involves too many terms of too high dimensionality to gain an intuitive understanding of how predictions change with the values of the predictors. This situation is referred to as having a "black box" - a function which takes inputs and produces an output without an explanation of how it arrives at them. It is therefore desirable to have some alternative avenue in which to gain insight into functional behavior: an x-ray of the black box. Graphical displays of various sorts have been the most popular and effective diagnostic tools – not least because they cater to the mathematically less-inclined – a direction which I maintain in this work.

This thesis focuses on the problem of prediction and understanding a high dimensional function that is the result of a learning algorithm. However, there are other situations in which we may seek to understand complex, high-dimensional functions.

Likelihood functions for complex probability models are such a case. In this setting the data are fixed, but there is a high dimensional set of parameters and we wish to understand the behavior of the likelihood as those parameters change. Other situations include complicated simulations in engineering or physical problems when there are a large set of input variables and no close-form for the answer. For these methods, there is no training data. In this case, we can simulate the uniform distribution normally associated with the Functional ANOVA. This thesis contributes measures of interaction importance and displays for them in Chapter 3. There is no variance associated with the function, however, and variability at extrapolation is no longer an issue. The concerns of the remainder of the thesis are therefore irrelevant in such situations.

2.1 Specific Methods and Diagnostics

A number of machine learning tools do come with diagnostic aids of more or less effectiveness. I devote this section to exploring some specific methods and interpretability as a means of identifying the important properties of a good diagnostic tool.

2.1.1 Tree-Based Methods

The most obviously and successfully interpretable methods in machine learning have been trees. There are two main interpretational tools available here. The first of these is the tree glyph, demonstrated in Figure 2.1. This tree was learned to predict the median house price in a suburb of Boston dependent on measured demographic variables in the Boston Housing Data [12], which will be the practical example used throughout this thesis. The glyph takes the form of a graphical model: the nodes

represent decisions and decision boundaries with edges leading to subordinate decisions. This representation is immediately appealing in providing a direct method to manually make predictions; the order of variables appearing in splits provides a notion of variable importance and specific predictions can easily be explained by the sequence of decisions.

An alternative approach to understanding trees is as a set of rules. This is taken in [24]. The set of decisions leading to each prediction forms a rule which can be written down and the rule set can then be examined. Although requiring more intellectual labor, this logician's method can be useful in presenting regions of constant prediction, particularly for small trees. A list of rules, however, does not provide a easy indication of how predictions change as one or more of the predictors is altered.

Both the tree-glyph and rule sets become more difficult to interpret as trees get larger and hence more flexible. A tree of a hundred levels, with possibly thousands of leaves induces more rules than can be readily understood. Tree-glyphs also quickly get unreasonably large to be followed or viewed. Even very large trees do retain the advantage of being able to specify how much predictors change in order to reach a desired result. This capability has been of particular use by credit card companies who must show cause for such decisions as denying an application.

2.1.2 Neural Networks

Neural Networks are another commonly used machine learning technique, and these also have a glyph to represent network structure. The glyph again takes the form of a graphical model, weights on each edge specifying the coefficients of linear terms within the model, and nodes sigmoid functions that "squash" the linear combination coming into them. Figure 2.2 provides an example of a 13-2-1 neural network trained on the same Boston Housing Data.

Neural networks have achieved popularity due to an analogy with the structure

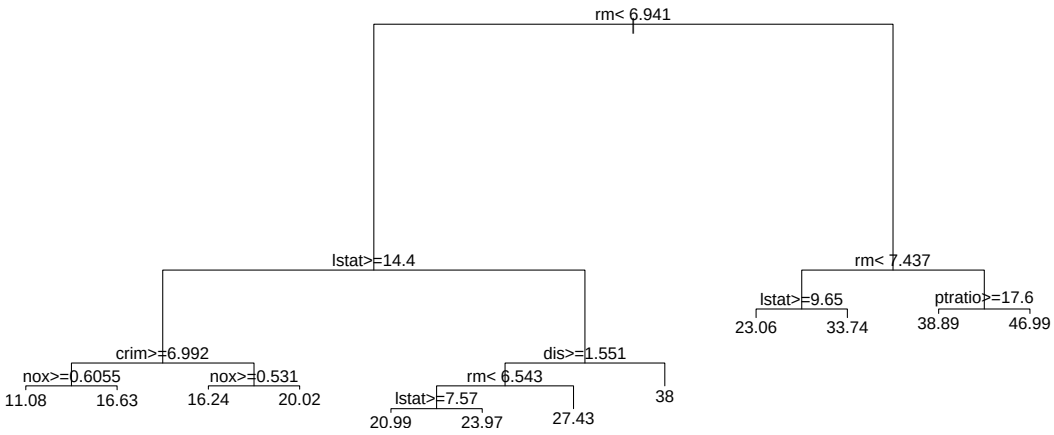

Figure 2.1: A tree learned on the Boston Housing Data.

of the human brain, which the glyph helps to emphasize. In this case the glyph represents considerably more complicated dynamics than those implied by the representation of trees. It is more difficult to use the weights on edges to examine behavior. Nonetheless, contemplating Figure 2.2 it is possible to see that the second hidden node takes a linear combination of variables with much larger weights on predictors "indus," "age," "b" and "lstat", providing most of the alteration in the prediction as values are changed and acting, for the most part, like a Generalized Linear Model.

As with trees, the interpretability of neural networks decreases rapidly as the size of the network increases. Having hundreds of hidden layers induces thousands of weights which are difficult to keep track of mentally. As with trees, small networks put large restrictions on the flexibility of the resulting models, restricting the class of functions that can be approximated well.

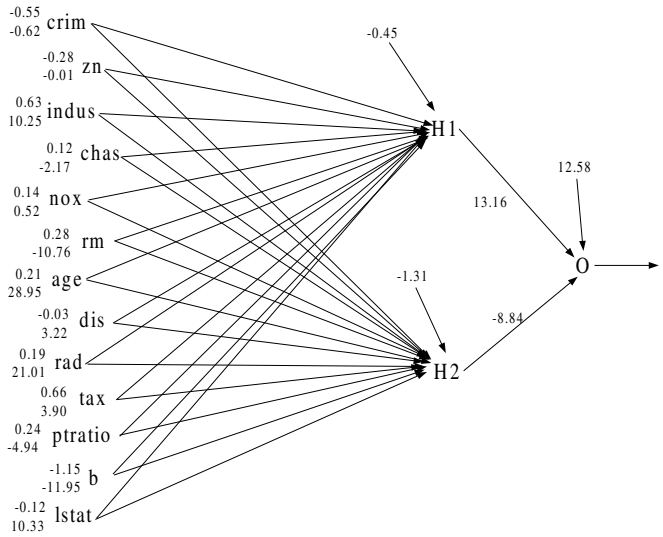

Figure 2.2: A 13-2-1 neural network fit to the Boston Housing Data. Weights for each input are given to the left of the input node in order of the hidden layer nodes.

2.1.3 Less Interpretable Methods

There are numerous other models and methodologies that do not have immediate graphical diagnostic tools. The general set of kernel methods, including support vector machines and nearest neighbors are among these. These methods produce an estimate of the form

$$\sum_{i=1}^{N} \beta_i K(x, x_i; \alpha_i)$$

for some "kernel" function $K(\cdot, \cdot; \alpha)$. Even the most direct of these are difficult to interpret in high dimensions, although they can be imagined in bivariate situations as adding a "bump" centered on each data point x_i. Even for a relatively small number of kernel centers – support vector machines provide a sparse representation of the function – such functions are not readily interpretable in high dimensional space.

Ensemble methods are another area of machine learning in which interpretation

becomes difficult. Ensembles have most frequently been used to provide additive combinations of trees. Taking a linear combination of hundreds of trees, even small trees, effectively destroys the interpretability of the tree glyph and creates far too many rules to be manageable.

These methods are some of the best-performing models in the machine learning literature in terms of predictive accuracy. The work in this thesis is aimed at a general approach precisely to provide a set of diagnostic tools compatible with any black box function, including those just listed.

2.1.4 Structured Models and Interpretation

Interpretation does become easier in situations where more structure is imposed upon the model, although this comes at the expense of flexibility[1]. Two common approaches to restricting the flexibility of the model class which result in improved diagnostic understanding are Generalized Linear Models [20] and Generalized Additive Models [13]. We will examine these as providing an indication of the type of information that we would like to glean from a general function.

Generalized Linear Models

The interpretability of Generalized Linear Models comes from the algebraic formulation that is lacking in more flexible machine learning models. In this case the model can be written in the form

$$g(\sum_{i=1}^{d} \beta_i x_{\{i\}})$$

for a known, usually monotone increasing, function g. The β_i (suitably scaled to reflect the variance of $x_{\{i\}}$) then provide the relative importance of each predictor

[1]This restriction can also result in improved accuracy through regularization and is often the main motivation for it. This will not be our concern for diagnostic purposes, however

and the behavior of the resulting prediction as $x_{\{i\}}$ varies depends only on the shape of g and the magnitude and sign of β_i.

That view is somewhat simplistic. While it is true that a larger coefficient equates to a steeper prediction for a standard linear model, the shaping function g can have a strong effect on the over-all diagnostic. If logistic regression is used to classify well-separated classes, for example, a very large value of β can actually translate into a prediction function that is close to flat where data are observed, even though it will be precipitous in the vicinity of the class boundary.

Diagnostics also become more difficult when interaction terms are included in the model. This can be done to provide more flexibility. Including sets of terms

$$\sum_{i=1}^{d}\sum_{j=1}^{d}\gamma_{ij}x_{\{i\}}x_{\{j\}}$$

along with the linear terms above results in "quadratic regression", used as an example in Chapter 6. Here the γ_{ij} represent coefficients of an *interaction* and their relative sizes indicate the importance of each. They do, however, confuse the way in which the prediction function varies with a single predictor.

Generalized Additive Models

An alternative method in which interpretability comes with a restricted model class is in the realm of additive models. Here the prediction function is given as

$$\sum_{i=1}^{d} f_i(x_{\{i\}})$$

for unknown functions f which must be estimated. Since each effect f_i is univariate, however, they can each be plotted; for example in Figure 2.3. In this case such a set of d plots captures all the behavior of the function.

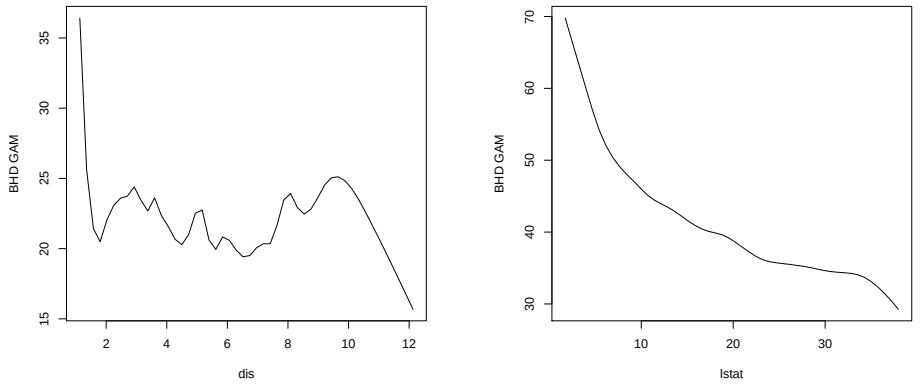

Figure 2.3: The effects for "dis" and "lstat" of a Generalized Additive Model built on the Boston Housing Data.

Second order effects in the form of terms $f_{ij}(x_{\{i\}}, x_{\{j\}})$ are the natural generalization of interaction terms for Generalized Linear Models. These can also be viewed in contour or mesh plots, although they do not have an immediate importance score attached them. Higher order interactions can be included in GAM models, but are more difficult to understand.

2.2 The Functional ANOVA

The development of displays for more structured methods, particularly those in §2.1.4 points toward the approach of this thesis. While defining a "reason" for a given prediction is restricted to trees, all diagnostic tools try to explain the behavior of a function as some given predictor, or predictors vary. This thesis targets a *general* diagnostic tool that can accompany any predictive function, and I concentrate on developing a GAM-like representation. This will shift the focus somewhat away from the interpretational tools above. The interpretation of GLMs and neural networks is most easily thought of as examining the behavior of a function *with the other*

variables fixed. Instead, this work focuses on how a function changes *in aggregate* with a particular predictor or set of predictors. The central tool used to do this has been the Functional ANOVA.

The Functional ANOVA decomposition has been studied in many contexts and appears in the literature as far back as [16]. Modern applications have been in the study of Quasi Monte Carlo methods for integration [22] and functional data analysis [27]. It has been used more directly in a machine learning context in, for example, [31] for fitting additive models of low dimensional components, an approach also followed in [15]. [28] provides an account of the use of the standard Functional ANOVA for the visualization of high-dimensional functions.

The standard definition of the decomposition is given as follows. Let $F(x) : [0,1]^d \to \mathbb{R}$ be square integrable. For $u \subseteq \{1,\ldots,d\}$, denote by x_u the subset of variables whose indeces are in u. Similarly x_{-u} indicates the variables with indeces not in u. $F(x)$ can be written uniquely as

$$F(x) = \sum_{u \subseteq \{1,\ldots,d\}} f_u(x)$$

with f_u depending only on x_u. $\{u \subseteq \{1,\ldots,d\}\}$ denotes the set of all subsets of $\{1,\ldots,d\}$, so that in more comprehensible terms, this is a mean ($u = \phi$), plus first order effects ($u = \{i\}$), plus second order effects and so on. To make such a decomposition unique, each effect f_u is defined as

$$f_u(x) = \int_{x_{-u}} \left(F(x) - \sum_{v \subset u} f_v(x) \right) dx_{-u},$$

the projection of F onto the variables x_u minus all the lower order effects. A number of properties result from this definition. Firstly, for each effect f_u and each $i \in u$ and each $x_{u \setminus i}$

$$\int f_u(x_{\{i\}}, x_{u\setminus i})dx_{\{i\}} = 0; \tag{2.1}$$

the effects integrate to zero in each co-ordinate direction for every value of the other predictors.

This gives us, secondly, that the f_u belong to orthogonal subspaces under the usual inner product with a uniform measure.

Orthogonality, in turn provides the third property: that the function variance ($\sigma^2(f) = \int f^2 dx$ with $\sigma^2_\phi = 0$) may be decomposed as

$$\sigma^2(F) = \sum_{u\subseteq\{1,\ldots,d\}} \sigma^2_u(f_u). \tag{2.2}$$

Particularly, if F is additive[2] in some set of variables u, then $f_u(x_u) = 0$ must hold and F is recovered exactly with the smallest possible number of terms. Note that the definition can be generalized trivially to any product measure.

2.2.1 Plotting Low-Order Effects

The Functional ANOVA provides an ideal tool in which to identify the "effect" of a predictor, or set of predictors. The quantity

$$\sum_{v\subseteq u} f_v(x_v)$$

represents the aggregate behavior of F on x_u, averaged over x_{-u}. Moreover, this quantity represents the $\mathcal{L}^2$ projection of F onto x_u making it optimal in the sense of being as close to F as any function that only makes use of x_u.

This decomposition was used in [17] and [28] to provide bivariate plots of functional dependence. Here, the plot given for the effect of x_u would be

[2] I employ the term *additive in u* to indicate that F can be written as $\sum_{i=1}^{q} f_{v_i}(x_{v_i})$ with no $v \supseteq u$.

$$F_u(x_u) = \sum_{v \subseteq u} f_v(x_v),$$

the direct projection of F onto the subspace of predictors x_u.

Plots of bivariate and univariate dependence can then be plotted together in a "full grid of plots" - a strategy advocated in [28]. Univariate effects and Normal theory confidence intervals are given as plots on the diagonal of a grid, with off-diagonal elements being filled with the corresponding bivariate effects. Since there are two available positions for each bivariate effect, the lower triangle can be used to give the variance of the whole prediction function at each point of the bivariate space. Formally, in the ijth element of the grid, for each value of $x_{\{i\}}$ and $x_{\{j\}}$ the plot provides

$$\mathrm{Var}(F(x)|x_{\{i\}}, x_{\{j\}}).$$

the univariate version of which is used for the confidence intervals in the plots on the diagonal.

The use of this variance is two-fold. It describes how much variance is not being captured by the effect f_{ij}. It is also an indication of model structure - a constant variance implies that

$$F(x) = f_{\{i,j\}}(x_{\{i\}}, x_{\{j\}}) + f_{\{1,\ldots,d\}\setminus\{i,j\}}(x_{\{1,\ldots,d\}\setminus\{i,j\}}).$$

In other words, f_{ij} is an additive component of F. Figure 2.4 provides a subset of the "full grid of plots," using the predictors "lstat" and "dis".

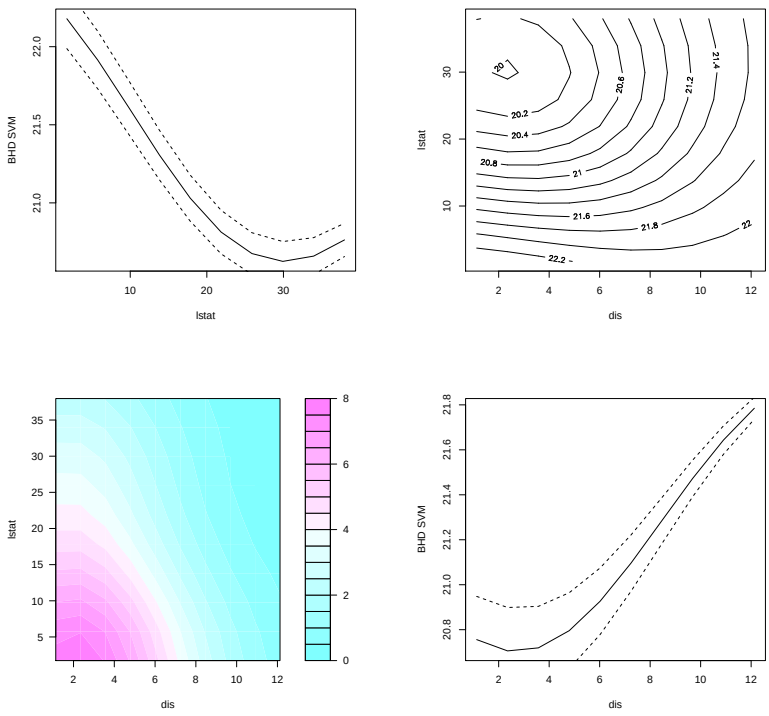

Figure 2.4: Matrix of plots of effects for a regression Support Vector Machine trained on the Boston Housing Data. The effects for "dis" and "lstat" are plotted in the diagonal elements. The upper diagonal provides the bivariate effect and the lower diagonal the functional variance conditioned on the bivariate values of the two predictors.

2.3 Partial Dependence Plots

A variation on the Functional ANOVA effects that provides less probability distortion is the Partial Dependence Plot. This is described in [10] for ensembles of trees, although the same techniques were employed for kernel methods in [18]. [10] defines the *partial dependence* of a function $F(x)$ on x_u to be

$$F_u(x_u) = E_{x_{-u}} F(x) = \int F(x_u, x_{-u}) dP_{-u}(x_{-u}) \tag{2.3}$$

where $P_{-u}(x_{-u})$ is the marginal distribution of x_{-u}. Both [10] and [18] note that this

estimate recovers additive or multiplicative components up to constants. That is for $F(x) = f(x_u) + g(x_{-u})$, (2.3) returns $f(x_u) + c$, and for $F(x) = f(x_u)g(x_{-u})$, it gives $cf(x_u)$.

This comes at the expense of distorting the distribution of predictor variables. (2.3) makes the implicit assumption that there is a product distribution between x_u and x_{-u}, in the sense that it satisfies the optimality criterion

$$f_u(x_u) = \underset{g \in \mathcal{L}^2(\mathbb{R}^u)}{\operatorname{argmin}} \int \left(g(x_u) - F(x)\right)^2 dP_u(x_u) dP_{-u}(x_{-u})$$

In the framework of taking projections, however, there is a sense in which this is the best we can do.

Theorem 2.3.1. *Among all distributions $Q(x)$ such that*

$$\underset{g \in \mathcal{L}^2(\mathbb{R}^u)}{\operatorname{argmin}} \int \left(g(x_u) - F(x)\right)^2 dQ(x) \tag{2.4}$$

recovers $h_u(x_u)$ for all functions of the form $h_u(x_u) + h_{-u}(x_{-u})$, $P_u(x_u)P_{-u}(x_{-u})$ has the smallest Kullback-Leibler divergence from the true distribution $P(x)$.

Proof. To begin with, note that among all functions of the form $Q_u(x_u)Q_{-u}(x_{-u})$, the product of marginals, $P_u(x_u)P_{-u}(x_{-u})$, has the smallest Kullback-Leibler divergence from $P(x)$. This is a trivial generalization of the standard theorem that the closest product distribution to $P(x)$ is the product of its marginals [14].

Supposing $Q(x)$ cannot be written as a product distribution on x_u and x_{-u}, the solution to (2.4) is

$$\int \left(F(x_u) - g(x_{-u})\right) dP(x_{-u}|x_u)$$

and since $P(x_{-u}|x_u)$ is by assumption not constant in x_u, a function $g(x_{-u})$ can be found such that

$$g'(x_u) = \int g(x_{-u}) dP(x_{-u}|x_u)$$

is also non-constant. This can be constructed directly: for some y_{-u}, $P(y_{-u}|x_u)$ is not constant. Letting $g(x_{-u}) = \delta_{y_{-u}}(x_{-u})$, $g'(x_u) = P(y_{-u}|x_u) \neq c$[3]. $Q(x)$ must therefore be a product measure in x_u and x_{-u} and the closest of these to $P(x)$ is $P_u(x_u)P_{-u}(x_{-u})$. □

[10] provides an efficient algorithm for producing plots of these functions for ensembles of trees. A data-driven approximation can be produced for any black box function F by calculating

$$\hat{F}_u(x_u) = \frac{1}{N} \sum_{i=1}^{N} F(x_u, x_{i,-u}) \tag{2.5}$$

where $\{x_i\}_{i=1}^N$ is the sample used to learn F. The set $\{F(x_u, x_{i,-u})\}_{i=1}^N$ can also be used to calculate an empirical variance for confidence intervals. Figure 2.5 is the equivalent to Figure 2.4 using Partial Dependence Plots. Some differences can be seen: a slight shift in the location of the mode leading to a considerable alteration in the effect for "dis".

2.4 Classification Diagnostics

The diagnostic tools in this chapter have dealt with regression problems. For two-class classification, the same diagnostics can be given, interpreted as an estimation of the classification probability surface. For multiclass classification, the straightforward approach to plotting effects is to simply plot an effect for each $P(C_i|x)$, the probability of class C_i given x. Given K classes, there are K plots for each effect.

[3] $g(s_{-u})$ is not strictly in $\mathcal{L}^2(\mathbb{R}^u)$, but with some continuity constraints on $P(x_{-u}|x_u)$, g can be taken to be the indicator of a small interval around y_{-u}.

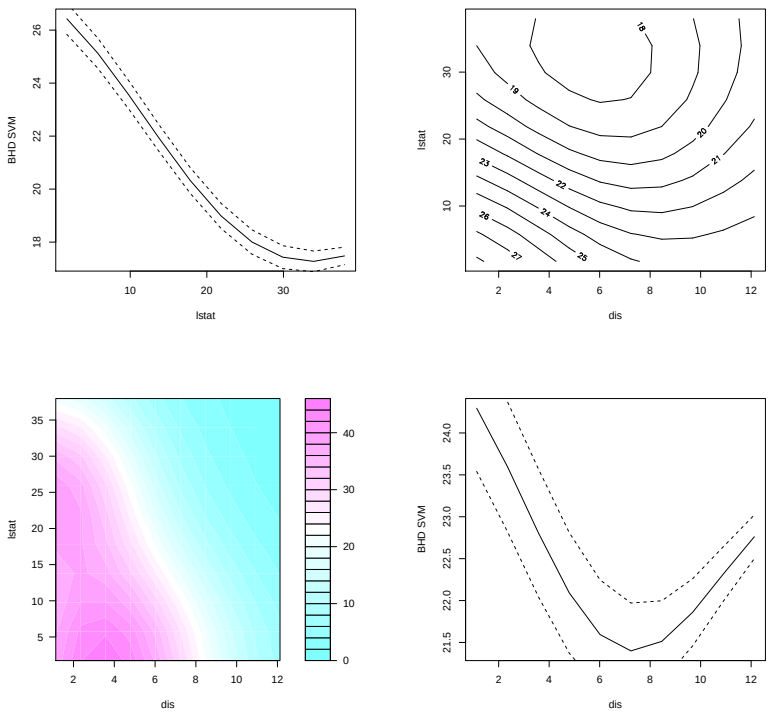

Figure 2.5: Partial Dependence Plots for a regression Support Vector Machine trained on the Boston Housing Data.

This can become unwieldy for large K, in which case plots can be given only for common or important classes.

The importance scores introduced in Chapter 2 can also be given on a class by class basis. If an over-all score is required for effect u, the natural measure to give is

$$\sum_{i=1}^{K} P(C_i)\text{Imp}_i(u)$$

where Imp_i is the importance score for u for the function $P(C_i|x)$. These are summed over classes, weighted by the prior probability of each class.

2.5 Degree of Explanation

So far the Functional ANOVA, and variants on it, have been used to provide plots of low order functional effects. A remaining question is "How much explanatory power do these effects have?" The variance decomposition (2.2) provides a quantification of the amount of variance explained by each effect, although this does not extend easily to Partial Dependence Plots. [30] proposes a number of measures of effect importance based on the quantities σ_u^2. [10] has alternative importance measures. Neither of these are targeted at providing a measure of how much of the over-all functional behavior has been recovered. Neither provides a measure for a set of plots taken in combination. This is the task of the next chapter.

Chapter 3

Variable Interaction Networks

In attempting to visualize the black boxes that result from machine learning, we are restricted to examining low dimensional behavior. It is possible to present contours of a function in at most three dimensions, and only bivariate contours can be examined without considerable intellectual effort. These may provide a good description of functional behavior. However, they do not provide an indication of how important each component is. If the function is truly high dimensional, they may also provide a quite misleading picture of dynamics.

The diagnostic tools presented here also have a limiting function. Chapter 2 presented the approach of the "full grid of plots" for black box predictions: producing a matrix of bivariate plots, each describing aggregate behavior on two of the predictors. These are complemented with plots of conditional variance: how much functional variation is not being accounted for at each value of those two predictors. I regard this as being a useful diagnostic aid on which to build. If the function in question is additive up to first order interactions, then such a plot-matrix exactly captures its dynamics: one only needs to sum up the values of the plots, a relatively simple cognitive procedure. Plots of bivariate conditional variances may indicate where something intrinsically higher-dimensional is going on, but they do not indicate what variables to

look for as additional interactions, or indeed how many variables might be included in this interaction. The work that I present here solves this problem.

In keeping with the philosophy of Theorem 2.3.1 and in anticipation of the extrapolation problems exemplified in Chapter 4, the calculations in this chapter are based on Partial Dependence Plots where a uniform distribution would normally be employed. For cases where an original data sample is not available, a uniform distribution can, of course, be substituted.

3.1 Tests of ANOVA Structures

Throughout this section I assume that the predictor variables are drawn from a product distribution in order to retain the interpretational properties of the Functional ANOVA. In §3.2 I give a data-driven approximation that is similar to Partial Dependence Plots (2.5).

A function f is said to have ANOVA structure described by a collection $\mathbf{U}$ of subsets of $\{1,\ldots,d\}$ if $f_u(x) = 0$ for all $u \subset \{1,\ldots,d\}$ that are proper supersets of some element of $\mathbf{U}$. Expressing the subsets of $\{1,\ldots,d\}$ as a lattice space ordered by the subset operator, this is equivalent to $\mathbf{U}$ being a least upper bound on the set of elements u of the lattice with non-zero $\sigma_u^2(f_u)$. In concrete terms, a function of the form $f_1(x_1) + f_2(x_2, x_3)$ would be said to have structure $\{\{1\},\{2,3\}\}$, describing the terms that are necessary to recover the function.

Theorem 3.1.1. *The $\mathcal{L}^2$ projection of F onto the set of functions with ANOVA structure described by $\mathbf{U}$ is given by:*

$$G_{\mathbf{U}}(x) = \sum_{i=0}^{|\mathbf{U}|} (-1)^{|\mathbf{U}|-i} \sum_{v \in \cap_i \mathbf{U}} E_{-v} F(x) \tag{3.1}$$

where $|\mathbf{U}|$ is the cardinality of $\mathbf{U}$, $\cap_i \mathbf{U}$ represents the collection of i-way intersections among the elements of $\mathbf{U}$, and $E_{-v}F(x)$ represents expectation conditioned on x_v.

To see this, we take the use the following lemma:

Lemma 3.1.1. *Define the collection* $\bar{\mathbf{U}} = \{v : \exists u \in U, v \subseteq u\}$*, then*

$$G_{\mathbf{U}}(x) = \sum_{v \in \bar{\mathbf{U}}} f_v(x_v)$$

for $G_{\mathbf{U}}$ *defined as in (3.1).*

Proof. This will be done by induction. It is trivially true for $\mathbf{U} = \phi$. Suppose it is true for some $\mathbf{U}$, and consider $\mathbf{U} \cup \{u\}$.

Firstly, this is true for $\mathbf{U} \cap u = \{v \cap u, v \in \mathbf{U}\}$. To get this, we take the expectation of $G_{\mathbf{U}}$ over $-v$ and observe that (2.1) means that any effect outside v cancels out. Now

$$\begin{aligned} G_{\mathbf{U}\cup\{u\}}(x) &= G_{\mathbf{U}}(x) + \sum_{i=0}^{|\mathbf{U}|} (-1)^{|\mathbf{U}|-i+1} \sum_{v \in u \cap \{\cap_i \mathbf{U}\}} E_{-v} F(x) \\ &= \sum_{v \in \bar{\mathbf{U}}} f_v(x) + \sum_{v' \subseteq u} f_{v'}(x) - \sum_{v'' \in \overline{\mathbf{U} \cap u}} f_{v''}(x) \\ &= \sum_{v \in \overline{\mathbf{U} \cup \{u\}}} f_v(x) \end{aligned}$$

Since the intersection of $\bar{\mathbf{U}}$ and $\bar{u}$ is exactly $\overline{\mathbf{U} \cap u}$. □

We can now proof Theorem 3.1.1:

Proof. We observe that from Lemma 3.1.1,

$$\begin{aligned} F(x) &= G_{\mathbf{U}}(x) + (F(x) - G_u(x)) \\ &= \sum_{v \in \bar{\mathbf{U}}} f_v + \sum_{v \notin \bar{\mathbf{U}}} f_v. \end{aligned}$$

By (2.1) the second term is orthogonal to any function with ANOVA structure $\mathbf{U}$ and therefor for any other G,

$$\int (F(x) - G(x))^2 dx = \int (F(x) - G_{\mathbf{U}}(x))^2 dx + \int (G_{\mathbf{U}}(x) - G(x))^2 dx$$

which is larger than the distance of F from $G_{\mathbf{U}}$. □

The natural measure of goodness of fit for this projection is therefore:

$$E(F(x) - G_{\mathbf{U}}(x))^2$$

with the expectation taken over the underlying measure.

Consider a 3 dimensional function, $f(x_1, x_2, x_3)$, with underlying uniform measure. Project this function onto the set of functions with ANOVA structure $\{\{1,2\},\{2,3\}\}$: additive in x_1 and x_2 and constant in x_3. The projection is given by:

$$\begin{aligned} E(f|x_1,x_2) + E(f|x_2,x_3) - E(f|x_x) + E(f) &= \int f(x_1,x_2,x_3)dx_3 \\ &+ \int f(x_1,x_2,x_3)dx_3 \\ &- \int f(x_1,x_2,x_3)dx_1dx_3 \\ &+ \int f(x_1,x_2,x_3)dx_1dx_2dx_3 \\ &= f_{12}(x_1,x_2) + f_{23}(x_2,x_3) \\ &\quad + f_1(x_1) + f_2(x_2) + f_3(x_3) + f_0 \end{aligned}$$

the corresponding two second order effects plus the first order effects and the mean.

Generally, we are interested in the significance of an interaction u, indexed by a subset of $\{1,\ldots,d\}$. Putting this formally, we ask: $\exists v \in \mathbf{U} : u \subseteq v$: *Is there a*

non-zero functional ANOVA component v that contains u? Alternatively: do we need to consider u or its supersets to exactly recover F? "Significance" here is defined in terms of its practical use in explaining F, which is a different notion from statistical significance.

We measure this by projecting F onto functions with ANOVA structure $\mathbf{U} = \{\{1,\ldots,d\}\backslash\{i\}\}_{i\in u}$: the most general structure that does not contain an effect for u. This is an upper bound for the set of ANOVA structures not containing u. In this case (3.1) simplifies to:

$$G_u(x) = \sum_{v\subseteq u}(-1)^{d-|v|-1}E_vF(x). \tag{3.2}$$

Here the quantity of interest, $E(F(x)-G_u(x))^2$, corresponds, in functional ANOVA terms, to the measure

$$\bar{\sigma}_u^2 = \sum_{v\supseteq u}\sigma_v^2(f_v), \tag{3.3}$$

the sum of variances for effects containing f_u.

In the three dimensional example above, $\bar{\sigma}_1^2$ measures the error associated with approximating $f(x_1,x_2,x_3)$ with a function of the form $g(x_2,x_3)$, leaving out x_1. The interaction $\{x_2,x_3\}$ can be tested by projecting onto the set of functions of the form $g_1(x_1,x_2)+g_2(x_1,x_3)$ and so forth.

This measure is of interest in the statistical quadrature literature [19] although it differs from the measures of subset importance given by [30]. It can be viewed as the $\mathcal{L}^2$ cost of excluding the interaction u from the model and is alternatively labeled the $\mathcal{L}^2$ *Cost of Exclusion* or $\mathcal{L}^2$CoE.

3.2 Empirical Estimates

In order to estimate the quantities (3.3) empirically I perform a simple Monte Carlo integration using the training data, in the vein of the partial dependence plots described in §2.3. Here,

$$\hat{E}_{-v}F(x) = \frac{1}{N}\sum_{i=1}^{N} F(x_v, x_{i,-v})$$

is the empirical partial dependence function of F on v. This is used to construct the empirical projection $\hat{G}_{\mathbf{U}}(x)$ from (3.1) and measure

$$\frac{1}{N}\sum_{i=1}^{N}(F(x_i) - \hat{G}_{\mathbf{U}}(x_i))^2. \tag{3.4}$$

Although I assumed that the data are drawn from a product distribution in §3.1, this is not strictly necessary. A function which exactly fits a given ANOVA structure is exactly recoverable with estimates based on partial dependence functions. However, this is no longer the $\mathcal{L}^2$ projection of the prediction function under a known measure onto a space of functions defined by a given ANOVA structure. In this sense, the exact interpretation of the empirical $\mathcal{L}^2$CoE is less clear.

Note that there is a variance associated with the estimate of both $G_u(x_i)$ and the outer sum of (3.4). The computational cost of (3.4) is $O(N^2)$. However, if for each i we estimate $\hat{G}_{\mathbf{U}}(x_i)$ using a randomly drawn subsample of size N_1, we can write down

$$\hat{G}_{\mathbf{U}}(x_i) = G_{\mathbf{U}}(x_i) + \epsilon_i$$

where ϵ_i has mean zero and variance $\frac{1}{N_1}E(F(x) - G_{\mathbf{U}}(x))^2$. This gives

$$E\frac{1}{N}\sum_{i=1}^{N}(F(x_i) - \hat{G}_{\mathbf{U}}(x_i))^2 = \left(1 + \frac{1}{N_1}\right)E(F(x) - G_{\mathbf{U}}(x))^2$$

with a variance of $O\left(\frac{1}{N} + \frac{1}{NN_1}\right)$. Setting $N_1 = 1$ provides an $O(N)$ estimation scheme without sacrificing the order of variance. At this point, the estimate of $\bar{\sigma}_u^2$ becomes

$$\hat{\bar{\sigma}}_u^2 = \frac{1}{2N}\sum_{i=1}^{N}\left(F(x_i) - \sum_{i=0}^{|\mathbf{U}|}(-1)^{|\mathbf{U}|-i}\sum_{v\in\cap_i\mathbf{U}}F(x_{i,v}, x_{r(i),-v})\right)^2$$

where $r(i)$ is an integer randomly chosen in $\{1,\ldots,N\}$. Some algebra shows that if $\bar{\sigma}_u^2 = 0$ (the function does not have an interaction in u) then the estimate also returns zero. This estimate is very similar to the Monte Carlo estimates employed by [19]; although that paper bases its estimates solely on uniform distributions.

3.3 Graphical Displays

In order to make use of the cost measures developed above as a diagnostic tool, it is helpful to have a graphical representation of the ANOVA structure of a function. In a learned-function context, very rarely is the importance score for any particular interaction identically zero. Were this the case, such structure might be discovered from algebraic manipulation of the model equation. It is very likely that the details of the modeling procedure will produce small, spurious, interactions that should be excluded. Therefore, a representation that conveys which interactions are deemed significant, as well as the overall importance of interactions, is needed.

For first-order interactions the set of significant interactions can be represented as edges in a graph that uses predictor variables as nodes. In this case, the graph has a similar interpretation to a Bayesian Network. That is: F is additive in $x_{\{i\}}$ and $x_{\{j\}}$ given the collection of variables x_u if any path from $x_{\{i\}}$ to $x_{\{j\}}$ crosses x_u. In fact, a Bayesian Network represents exactly this structure for the log density of the variables. An equivalent functional interpretation is that F is additive in $x_{\{i\}}$ and $x_{\{j\}}$ if for any fixed $x_{-\{i,j\}}$, $F(x_{\{i\}}, x_{\{j\}}; x_{-\{i,j\}}) = f(x_{\{i\}}) + g(x_{\{j\}})$ for some functions f and g. This implies the statement

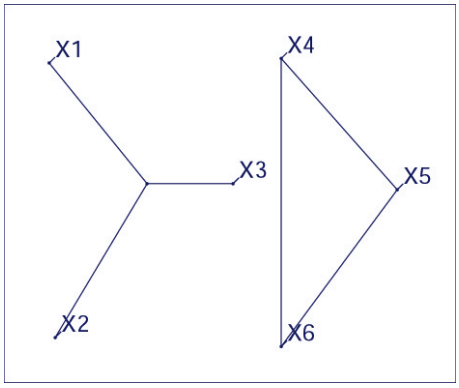

Figure 3.1: An example of the difference in the VIN graph for functions of the form $f(x_1, x_2, x_3)$ and $g_1(x_4, x_5) + g_2(x_4, x_6) + g_3(x_5, x_6)$.

$$\int F(x_{\{i\}}, x_{\{j\}}, x_u) dx_u = f(x_{\{i\}}, x_{-\{i,j\}}) + g(x_{\{j\}}, x_{-\{i,j\}}).$$

I have labeled such a representation a Variable Interaction Network (VIN).

The inverse of the $\mathcal{L}^2$CoE measure can be taken as a distance between nodes, so that the graph representation may be incorporated into a multi-dimensional scaling routine. The package XGvis [4] has been used to produce the graphs in Figures 3.1, 3.2, and 3.3, although the nodes have been positioned by hand. I have found that while dynamic views of the scaled graphs – having a plot rotate with a three dimensional representation – are informative about interaction strengths, static two-dimensional plots have not provided a good representation of interaction strengths.

A representation limited to edges in a graph with variables as nodes still does not distinguish higher-order interactions, except in so far as they must appear as cliques in the graph; a fact used in §3.4. These higher interactions are therefore represented by a "cartwheel" to distinguish them from a set of additive first order terms. The VIN network that is now produced can be interpreted as a hyper-graph, with cartwheels representing multi-dimensional edges. This difference is demonstrated in Figure 3.1.

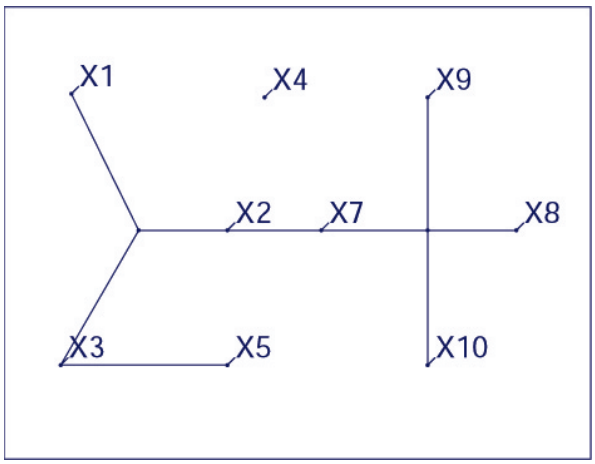

Figure 3.2: Variable Interaction Network for the function (3.5) showing non-additive structure.

As an example, consider the function[1]

$$F(x) = \pi^{x_1 x_2}\sqrt{2x_3} - \sin^{-1}(x_4) + \log(x_3 + x_5) - \frac{x_9}{x_{10}}\sqrt{\frac{x_7}{x_8}} - x_2 x_7. \tag{3.5}$$

The true VIN components for F are

$$\mathbf{U} = \{\{1, 2, 3\}, \{2, 7\}, \{4\}, \{3, 5\}, \{7, 8, 9, 10\}\} \tag{3.6}$$

which induces the plot in Figure 3.2.

Here edges $\{1, 2\}$, $\{1, 3\}$ and $\{2, 3\}$ would normally form a clique if only first-order interactions were considered. A similar clique occurs for $\{x_7, x_8, x_9, x_{10}\}$.

Note that these cartwheels do not alter the graphical interpretation if we allow as paths any route through an "intersection".

These plots may also be thought of as a graphical version of the representation for hierarchical log-linear models given in [7]. There, a log-linear model is specified

[1] I am indebted to Matthew Finkelman for dreaming up this along with many other wondrous functions as test examples.

on categorical data with the highest order interaction terms listed as in (3.6). In that case, the absence of an interaction represents independence between categorical variables instead of additivity of a response in either categorical or continuous predictors.

3.4 The VIN Algorithm

While the evaluation of the strength of a particular interaction can be made in $O(N)$ functional evaluations, there are still 2^d interactions to be evaluated. Denoting by $|u|$ the size of an interaction, the complexity of evaluation also scales as $2^{|u|}$. This becomes prohibitive as d and $|u|$ increase. However, for many functions the strength of interactions drops off quickly with their size, enabling a very aggressive search strategy.

Algorithm 3.4.1 makes use of the following monotonicity property:

$$u \subset v \Rightarrow \bar{\sigma}_u^2 \geq \bar{\sigma}_v^2. \tag{3.7}$$

That this holds is clear from equation (3.3). Monotonicity provides that a d-way interaction can only be considered significant if all its $(d-1)$-way interactions are significant[2]. Thus the algorithm begins by considering main effects - removing one variable - giving a measure of variable importance. It then proceed to first order interactions whose components are all in the significant list. Second order interactions are only considered if all the first order interactions that they contain are included, and so forth. The algorithm here bears strong resemblance to the *Apriori* algorithm for association rules [1]. Both of these rely on a monotonicity property to dramatically reduce the search space as the complexity of interactions (or item-sets) that are being considered increases.

[2]Significance is defined by whether its $\mathcal{L}^2$CoE exceeds some threshold ϵ.

Algorithm 3.4.1. *VIN*
Inputs: *Monotone loss measure P_F, nominally $\mathcal{L}^2$ CoE for a function F with associated data matrix X. Threshold $\epsilon > 0$.*
Output: *List of interactions u for F with $P_F(u) > \epsilon$.*

$i = 1$
$\mathbf{U} = \phi$
`Loop:`
  $K = \{u \in \{1, \ldots, d\} : |u| = i, v \subset u \ \&|v| = i - 1 \Rightarrow v \in \mathbf{U}\}$
  `For Each` $u \in K$`:`
    `Calculate` $P_F(u)$
    `If` $P_F(u) \leq \epsilon$
      $\mathbf{U} \leftarrow \mathbf{U} \cup u$
      $\mathbf{U} \leftarrow \mathbf{U} \setminus \{v \subset u\}$
    `End If`
  `End For`
$i \leftarrow i + 1$
`End Loop (`$K = \phi$ `or` $i > D$`)`

Suppose I have a threshold $\epsilon \geq 0$, and I wish to find $u : \bar{\sigma}_u^2 \leq \epsilon$. Algorithm 3.4.1 provides a least upper bound for this set.

So long as the function in question does not exhibit very high-dimensional behavior, this algorithm examines only a very small subset of interactions, which themselves are of low order. I believe that it is unlikely that learned functions are intrinsically high dimensional. [22] explores many apparently high dimensional functions and finds that many are very close to additive. In the event of interactions exceeding some given order (say 6), the algorithm can be curtailed as an indication that interpretation will become very difficult.

3.5 Upper Bounds on Lattice Spaces

In §3.4, Algorithm 3.4.1 is designed to find $u : \bar{\sigma}_u^2 \leq \epsilon$. The natural measure of concern is the overall error resulting from a choice of interaction terms. It would be

ideal to find a minimal $\mathbf{U}$ to give

$$E(F(x) - G_{\mathbf{U}}(x))^2 < \epsilon$$

for $G_{\mathbf{U}}$ defined in 3.1: a representation that explains almost all of the function. Here, minimality is taken to be a least upper bound on a lattice. Instead of the set of elements with non-zero score $\sigma_u^2(f_u)$ given in §3.1, the item of interest is an upper bound $\mathbf{U}$ that gives[3]

$$\sum_{u \prec \mathbf{U}} \sigma_u^2(f_u) \geq 1 - \epsilon. \tag{3.8}$$

Unfortunately, this problem is ill-defined and many such $\mathbf{U}$ may exist.

3.5.1 Breadth and Depth Searches

In order to specify the problem, we can define a score on the set of $\mathbf{U}$ that satisfy (3.8). Importance can be given either to having low-order interactions, or a small number of variables. For the former, a breadth-first search can be performed, including all low-order interactions until the fitting requirement (3.8) is met. In this case the algorithm successively includes the term with highest $\mathcal{L}^2$CoE among those candidates of lowest order. In doing this I maintain the hierarchical requirement that possible candidates must already have all of their subsets included. Algorithm 3.5.1 provides formal pseudocode for doing this.

The estimate

$$\hat{\sigma}_u^2(u) = \sum_{v \subseteq u} (-1)^{|u|-|v|} \hat{\bar{\sigma}}^2(v).$$

can be employed here. The effect of Algorithm 3.5.1 is to place a penalty on the

[3] $u \prec \mathbf{U}$ is here taken to indicate $\exists v \in \mathbf{U} : u \subset v$.

Algorithm 3.5.1. *Breadth Search VIN*
Inputs: *Monotone loss measure P_F, nominally $\mathcal{L}^2$ CoE for a function F with associated data matrix X. Threshold $\epsilon > 0$.*
Output: *List of interactions* $\mathbf{U}$ *for* F *with* $\int (F - G_{\mathbf{U}})^2 dx < \epsilon$.

$S = 0$
$\mathbf{U} = \phi$
`Loop:`
  $K_1 = \{u \in \{1, \ldots, d\} : v \subset u \ \&|v| = i - 1 \Rightarrow v \in \mathbf{U}\}\}$
  $K_2 = \{u \in K_1 : \ |u| = \min_{v \in K_1} |v|\}$
  `For Each` $u \in K_2$`:`
    `Calculate` $P_F(u)$
  $v \leftarrow$ `argmax`$_{u \in K}$ $P_F(u)$
  $\mathbf{U} \leftarrow \mathbf{U} \cup v$
  $S \leftarrow S + \sigma^2(v)$
`End Loop` $S > 1 - \epsilon$

maximum size of an interaction, only including higher-order terms after all the lower-order have been entered. Here the target is a graph that minimizes the size of the greatest "cartwheel" in favor of many lower-order edges.

The alternative is a depth-first search: including the term with highest $\mathcal{L}^2$CoE among those candidates of highest order. The pseudo-code is identical to that above, replacing the definition of K_2 with

$$K_2 = \{u \in K : |u| = \max_{v \in K} |v|\}.$$

This does the exact converse to the breadth-first search and includes a new predictor variable only after all interactions in the current set of predictors have been allowed. The graph from this is expected to have large "cartwheels" but a smaller number of nodes.

3.5.2 Diagnostics and Greatest Lower Bounds

Given that the problem (3.8) is under specified, I believe that the algorithm as originally stated provides a reasonable set of interactions. The interactions resulting from it can be interpreted as a *greatest lower bound* on the sets $\mathbf{U}$ that satisfy (3.8).

Theorem 3.5.1. *The collection* $V = \{u : \bar{\sigma}_u^2 \geq \epsilon\}$ *represents a lattice greatest lower bound on collections* $\mathbf{U}$ *that satisfy (3.8).*

Proof. Suppose that $\bar{\sigma}_u^2 < \epsilon$, then the collection $\mathbf{U} = \{v : (v \not\supset u)\}$ has

$$\begin{aligned}\sum_{v \prec \mathbf{U}} \sigma_u^2(f_u) &= 1 - \sum_{w \in V} \sigma_w^2(f_w) \\ &> 1 - \epsilon.\end{aligned}$$

Conversely, for $\bar{\sigma}_u^2 \geq \epsilon$, any $\mathbf{U}$ with $u \not\prec \mathbf{U}$ has

$$\begin{aligned}\sum_{v \prec \mathbf{U}} \sigma_u^2(f_u) &= 1 - \sum_{w \in V} \sigma_w^2(f_w) \\ &\leq 1 - \epsilon.\end{aligned}$$

□

This gives a satisfying explanation of VIN as a collection of subsets that *must* be included to provide a good fit. Further, it aids interpretation significantly by providing a smaller collection of interactions.

There remains the question of choosing ϵ. In §3.6 I have employed a criteria of explaining 99% of the variance of the function on the empirical distribution of the training data. An alternative is to consider an ordered plot of scores for main effects - these typically die off exponentially - and choose a cutoff manually using a natural break in the scores. This cutoff could be employed throughout, or re-chosen at every

interaction size. It may well be advantageous to be more or less aggressive in choosing how many terms to include when the relative scores are seen.

3.6 Example: Boston Housing Data Support Vector Machine

I examine the ANOVA structure of the support vector machine from §2.3. Here I picked $\epsilon = 0.68$ which corresponds to 1% of the over-all variance of the function, calculated from the predictions given at the original data. At this cut-off, all but variables "chas," and "zn" are included. Of the remaining variables, "indus," "b," "age," "tax," and "rm" have interactions with "lstat". There are also interactions between "rad" and "crim" and "rm" and "ptratio". These are very similar findings to those of [28]. There are no second-order interactions, although they have been found in Figure 8.1 for neural networks trained on the same data. The resulting VIN plot is given in Figure 3.3. An additive model fit using these interactions and cubic splines improved test error by 2% indicating that this structure is already sufficiently flexible to model the data well.

From a diagnostic point of view, Figure 3.3 indicates that this prediction function can be well represented by a sum of low dimensional terms. This means that plots of bivariate representations of the function do provide a close-to-comprehensive account of functional behavior.

3.7 Conclusions

The work presented in this chapter can be viewed as a lattice search for a greatest lower bound on the set of hierarchical functional ANOVA components that can be used to represent a learned predictor function. This allows the complexity of a function

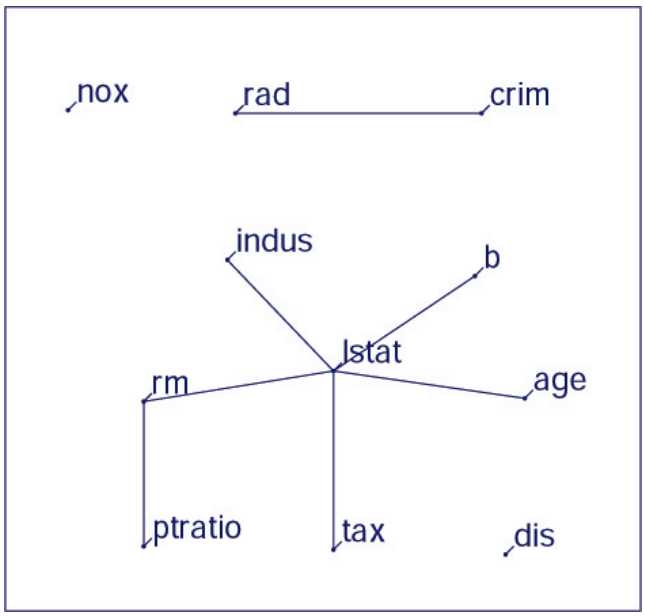

Figure 3.3: Variable Interaction Network for a support vector machine trained on the Boston Housing Data, with cutoff $\epsilon = 0.68$.

to be represented in terms of the size of its non-additive interaction components, providing not only an indication of the intrinsic dimensionality of the system, but which predictors interact in a non-additive manner.

This algorithm employs the monotonicity of the $\mathcal{L}^2$CoE measure to provide an efficient search through this lattice. Additional sampling theory demonstrates that this can be done in $O(N)$ function evaluations without compromising variance. I have developed a graphical display to make the results more accessible to the user and demonstrated that it has good interpretational properties.

The empirical estimators which I used are tied directly to the estimation of partial dependence plots. These have been chosen as providing a data-driven method which distorts the distribution of predictor variables less than the uniform distribution normally associated with the functional ANOVA. In Chapter 7 I show that it is possible to estimate $\mathcal{L}^2$CoE measures without requiring any predictor variables to be

independent. Where the underlying distribution is not an issue or is unknown, estimates based on a uniform distribution can be employed. The algorithms presented in this chapter are compatible with any measure of interaction importance for which the monotonicity (3.7) holds.

Chapter 4

Problems of Extrapolation

The diagnostic tools presented in Chapters 2 and 3 have not addressed the issue of extrapolation. Most learners also do not specify the behavior of their resulting functions in the absence of training data. They are usually designed to be universal approximators – or as close as is practical – and have minimal modeling restrictions. This, in turn, provides very little prior control of the function in regions of little or no data. As a result, in most machine learning settings, we can neither control the behavior of the prediction function at points of extrapolation, nor determine where this is a problem.

Nominally, extrapolation should not be an issue; assuming a representative training sample in a static system, the probability of being required to predict a point of extrapolation is low, almost by definition. However, most training sets are not representative, do not come from static systems and we may well be required to extrapolate. In high dimensions, even empirical data generated from a product distribution can appear to have strong correlation structure. Since functions are learned conditional on an empirical sample, they may still be effectively extrapolating even in regions of theoretically large density.

Even when extrapolation is not an issue for prediction, it does become a problem in

two situations. Firstly, when trying to understand the behavior of learned prediction functions, the diagnostic tools in Chapter 2 require the evaluation of the function on a measure in which the variables of interest are independent of their complement. Such an assumption can move a large amount of probability mass into regions of extrapolation, giving these regions undue influence over a representation of functional behavior.

The second situation in which extrapolation is a direct issue is when one or more predictor variables can be controlled. In this case an agency might want to search over the values of those variables in order to optimize a response. If there is strong historical correlation between these variables and the rest of the predictors – which remain fixed for a given example – such a search involves extrapolation.

4.1 Extrapolation and Prediction

Extrapolation is firstly an issue for predictive accuracy. Few machine learning procedures are built with an eye to extrapolation. This has two main consequences:

- Uncontrolled extrapolation can create obviously unrealistic predictions, even in superficially reasonable points of predictor space.
- Prediction at points of extrapolation exhibits high variance under perturbations of the training data, even when the predicted values appear reasonable.

I illustrate this issue with the aid of a bivariate distribution given in Figure 4.1. The association structure given there consists of an "arm" along each axis and is similar to that found in the Boston Housing Data. Such structure can be observed in Figure 5.4, for example, between variables "zn", "crim" and "indus." The canonical point of extrapolation for the purpose of this exposition is $(4, 4)$: clearly different from the remainder of the distribution, but well within its convex hull.

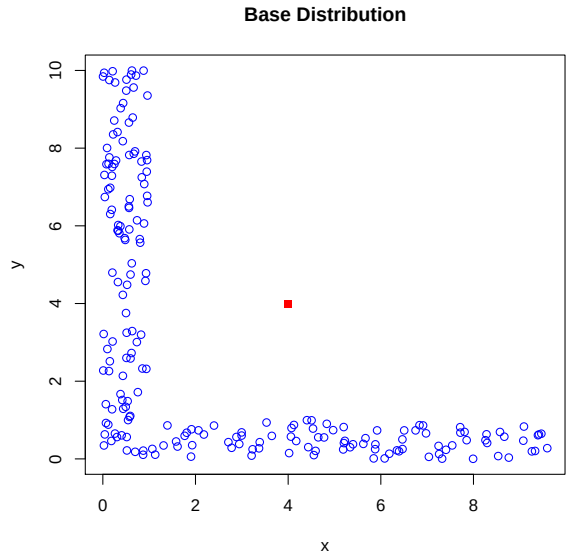

Figure 4.1: An example distribution with strong extrapolation. The square at $(4, 4)$ represents a point of extrapolation which is difficult to diagnose automatically.

The first consequence of extrapolation is that learned functions can produce unrealistic predictions. In this toy example, a polynomial model can exhibit Gibbs effects by which I mean large second derivative, where the function "wiggles" wildly – in the empty region and can also quickly diverge there. Other models, such as MARS [9], can be susceptible to points at the edge of the data – adding a term like $\alpha(x_1 - 1)_+(x_2 - 2)_+$ affects little training data but can become extreme in the empty region. If this example is generalized to K dimensions by placing an arm along each axis, even a generic linear model $x_1 + \ldots + x_K$ produces a prediction at point $(4, \ldots, 4)$ which is $.4K$ times the largest prediction on the training distribution. K need not be very large for this to look unreasonable.

In many instances, extreme predictions are obviously nonsensical and are therefore easy to diagnose. However, that diagnosis does not account for the variance of predictive functions when re-trained with different data. The second consequence of

ignoring extrapolation in designing a learning algorithm is that the resulting predictions can have high variance under perturbations of the training data. This is true for linear models where theoretical prediction intervals increase away from data. It is implicitly true where Gibbs effects are observed – large fluctuations without data to support them are necessarily suspect and can lead to large changes in predictions even with a small change in the function parameters.

This variance exists even for models that revert to constants away from data and which are therefore resistant to extreme extrapolation. Figure 4.2 provides a simple illustrative example for trees and neural networks trained on 200 points drawn from the base distribution described by Figure 4.1. The response is $x_1 - x_2 + \epsilon$ with ϵ independent Gaussian random variables with zero mean and variance 1. A histogram of predictions at the point of extrapolation over 100 resamplings of the training data demonstrates strong bi-modality around -4 and 4: about as much variance as it is possible to achieve. Effectively, the trees split one "arm" from the other at random and this then governs which arm the point is assigned to. I also provide a histogram for the predictions of a 2-20-1 neural network. Bi-modality – an artifact of the tree-building process – is not in evidence but large variability can still be observed. This can be explained by the observation that many different weightings in the network can provide similar predictions on the training data, but the resulting functions diverge away from these data.

4.2 Deliberate Evaluation at Extrapolation

As noted, extrapolation is not nominally a concern for a static system with a representative training sample that is large with respect to its dimensionality. This scenario is not common in data mining and even in this case, there are at least two situations in which prediction functions may be deliberately evaluated at points of extrapolation.

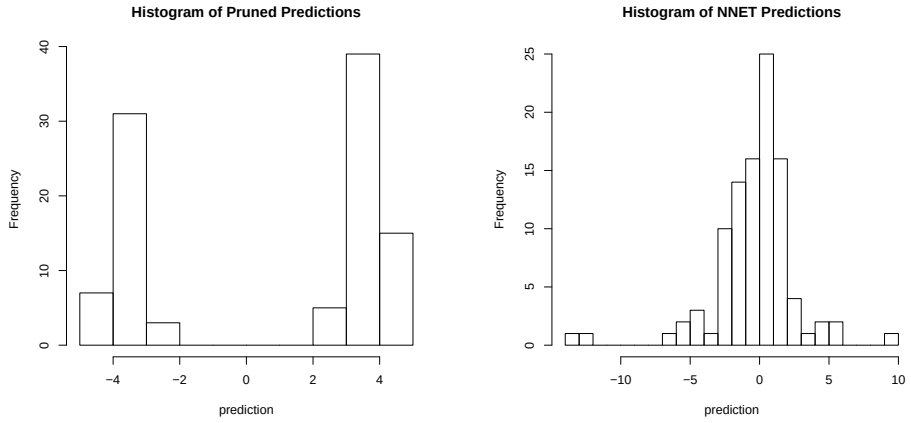

Figure 4.2: A histogram of CART predictions (left) at $(4, 4)$ point for 100 samples from the data given in Figure 4.1 and response $x_1 - x_2 + \epsilon$. The right hand plot gives a histogram of predictions from a 2-20-1 neural network. Both are highly variable.

4.2.1 Controllable Predictors

The first of these is when one of the predictor variables can be controlled by the agency that commissions the machine learning task. This might be the case, for example, for a credit agency dealing with customers who default on payments. One predictor of payment is the action that the agency takes to remind a customer that the payment is due: the frequency of letters, telephone calls, personal visits etc. Historically, the more severe the problem, the more extreme the action taken. If the agency then wants to build a predictor system for the probability of default so that they can choose a most cost-effective action, searching over the action space evaluates the function at points of extrapolation. In Figure 4.1, x_1 might represent an amount already paid with x_2 being the reminder action taken. Then a search over actions x_2 at $x_1 = 4$ evaluates at $(4, k)$ for $k \in [1, 10]$. A diagnostic tool for extrapolation at least allows such a procedure to flag predicted values that are untrustworthy. It also provides a diagnostic for areas of the predictor space in which the agency can conduct experiments to better gauge a response.

4.2.2 Product Measures and Extrapolation

The second scenario in which prediction functions are deliberately evaluated at points of extrapolation occurs in creating functional diagnostics. The Functional ANOVA decomposition given in Chapter 2 is defined on a uniform distribution on the unit hypercube, although I have noted that it can be extended to any product measure without harming the properties mentioned there. However, using a product measure ignores the fact of machine learning that prediction functions are almost never learned or evaluated on data drawn from independently distributed predictors.

The most important aspect of this is that a product distribution then places potentially large probability mass in areas of extrapolation. I have already demonstrated that learned prediction functions often cannot be considered trustworthy in these regions. This can lead to significant distortions in the effects that are presented. Gibbs effects that result from the use of polynomial models and models using corresponding kernels can significantly distort the effects that we see, often presenting a much larger amount of variation than actually exists. More conservative models that extrapolate as constants are also at issue in allowing potentially interesting effects to be dampened out by large, empty regions in which the function is essentially flat. Further, we showed both Gibbs effects and even constant extrapolation can be highly variable under re-sampling of the data, making for very unstable effects. Even when a function is regarded as being fixed, it is worthwhile asking whether a low dimensional representation should incorporate a large effect, even if it is true, if it occurs in a region of low probability.

4.2.3 Partial Dependence Plots and Extrapolation

The Partial Dependence Plots described in Chapter 2 can be regarded as mitigating the problem of extrapolation in the sense of Theorem 2.3.1. However, while they focus probability mass closer to the training data than a uniform distribution, the

implicit assumption of independence between the predictors of interest and the rest can still involve large amounts of extrapolation. I illustrate this point below.

An Example of Inadequacy

Despite the optimality cited in Theorem 2.3.1, Partial Dependence Plots remain susceptible to bad extrapolation. A simple example suffices to illustrate this. Consider the function

$$F(x, y) = x + y^2$$

defined on the probability measure:

$$P(x, y) = U(0 < \{x, y\} < 2, x < 1 \wedge y < 1). \tag{4.1}$$

This is a unit square minus the upper right quadrant; the distribution for Figure 4.1, curtailed to the 2-square for the purpose of making this demonstration intelligible. Further suppose that this function has been learned perfectly except for a spurious interaction

$$\hat{F}(x, y) = F(x, y) + 10 * (x - 1)_+^2 (y - 1)_+^2$$

which only occurs in the empty quadrant. This is not dissimilar to the bases used in the MARS algorithm [14], for example, and could plausibly occur as a result of an outlier. For the purpose of this demonstration, points in this quadrant are regarded as points of extrapolation.

The distortion in the underlying probability distribution is illustrated in Figure 4.3. Despite having probability mass zero in distribution (4.1), the product distribution places mass 1/9 in the same region.

The effect of this extrapolation can be seen in Figure 4.4. The example here has

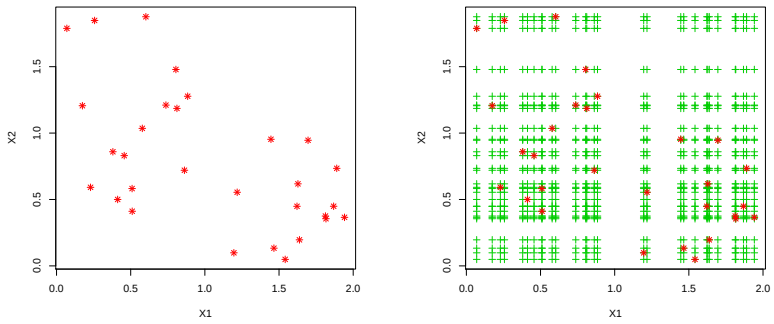

Figure 4.3: A base set of 30 points from distribution 4.1 used to learn a function, and the set of evaluation points for a Partial Dependence Plot defined in (2.5).

been chosen to provide for plots on a comparable scale to the effects that we would like to find. However, it should be clear that the distortion can be made arbitrarily large – returning to the 10-square in Figure 4.1 allows the spurious term to all but swamp the true effect.

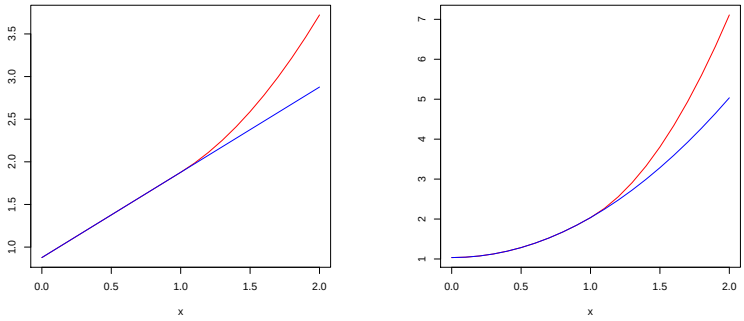

Figure 4.4: Partial Dependence Plot distortion due to the movement of probability mass into regions of extrapolation as detailed in Figure 4.3. Blue lines represent the effect that we would like to capture, red the actual Partial Dependence Plot.

While they have not been mentioned so far, conditional dependence plots, given by $E(F(x)|x_u)$, do escape the problem of extrapolation. However, they remain unsatisfactory in failing to recover low-order components. Figure 4.5 presents the conditional dependence for the example above. The problem is essentially the same as trying to do a series of univariate linear regressions on correlated predictor data: the correlation structure leads to duplication, or cancellation, of effects.

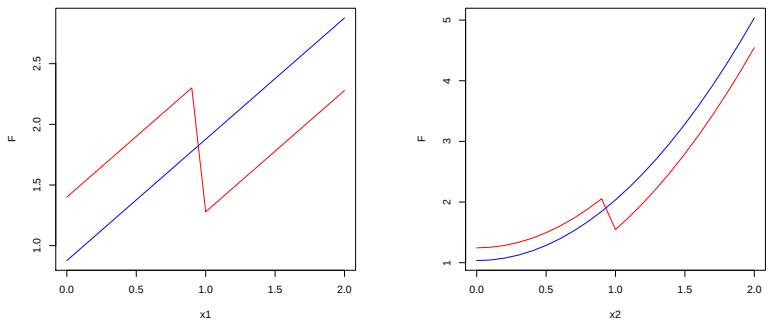

Figure 4.5: Conditional Dependence Plots (red) and desired effects (blue) for the underlying distribution (4.1).

4.3 Dealing with Extrapolation

So far, this chapter demonstrates the dangers and problems caused by extrapolation. The rest of this thesis is devoted to developing methods to mitigate these issues. The first task, of course, is to identify regions of extrapolation. In the examples above, what constitutes extrapolation is clear. However, this is less easy to identify in high dimensions. Indeed, the very notion of extrapolation is not formally defined. Chapter 5 is devoted to developing a theoretical measure of extrapolation and then finding estimates for it. This at least allows points of extrapolation to be flagged as such.

Having identified a point of extrapolation, how should we make a prediction for it? Ideally, we should report "don't know" and leave it at that. This is not always possible. However, it is possible to reduce the variability of predictions at points of extrapolation. Chapter 6 develops a method to induce a flexible learner to revert to some trusted prior model away from training data.

Finally, even for stabilized functions, prediction tools that rely on product distribution tend to revert to that prior model if the distribution of training points is highly concentrated. In Chapter 7, I develop a generalization of the Functional ANOVA that accepts a general weight function and retains most of the Functional ANOVA's desirable properties.

Chapter 5

Confidence and Extrapolation Representation Trees

Chapter 4 demonstrated the problems that result from the fact of extrapolation in machine learning. Given this phenomenon, the first problem to address is to find which parts of predictor space represent points of extrapolation. Diagnosing extrapolation is of interest in itself in terms of enabling an understanding of the distribution of underlying predictor variables. The ideas presented in this chapter represent one of the few interpretable estimates of density of which I am aware.

I present a new tool to address how we determine whether a given point in predictor space is a point of extrapolation. Such a determination has multiple uses. It represents a new diagnostic for outliers, both in the training data set and in unseen data. It provides a diagnostic for shifts in system dynamics; we can quantify how much extrapolation we are seeing and whether or not the rate of extrapolation is increasing. Such an observation suggests a shift in the distribution of the predictor variables and a need to retrain the model. I show that this tool provides an interpretable density estimate in high dimensions, giving a rough diagnostic for nonlinear covariance structures. Finally, in producing an estimate of the distribution of

the training data, it provides an important element in building diagnostic techniques that are resistant to bad extrapolation.

5.1 A Measure of Extrapolation

I propose to measure an extrapolation quotient as being the Neymann-Pearson test statistic at the point of interest for distinguishing the data distribution from a uniform distribution null hypothesis on the same range. This test is equivalent to the classification probability of a point being generated by the process generating the empirical data as opposed to the uniform distribution with each distribution is given prior probability of one half. Formally, I define:

$$\text{Extrap}(x) = \frac{\hat{P}(x)}{\hat{P}(x) + U(x)}. \tag{5.1}$$

Here, $\hat{P}$ is an approximation to the data distribution, P, which has compact support. This would normally be an empirical approximation given a data set. More formally, however, a distribution with non-bounded support can be truncated so as to leave some small probability mass outside the support. U is then a uniform distribution on the range of the support of $\hat{P}$.

This approach has a number of advantages. I formalize the question, *"Is this a point of extrapolation?"* to a test: *"Would I choose to believe this point to be generated from the same distribution as the training data or from a uniform distribution?"* Here we wish to determine if a new data point indicates a move away from the processes generating the training data and, if so, assess how much confidence can be placed in the predictions that our learned function produces.

I have chosen the uniform distribution as a natural null hypothesis in the sense that it is least informative about how a point of extrapolation might be generated. It also corresponds to a best-case underlying distribution from the point of view of

extrapolation - the coverage of the space may be sparse, but it provides the same confidence in predicted values at all areas of the space. This means that $\text{Extrap}(x) \geq 1/2$ indicates that x is in an area of relatively dense points. If not, then some caution is warranted in using a predicted value[1]. I also show that the uniform distribution is computationally convenient to use in the procedures below.

A further advantage is that under this framework, $\text{Extrap}(x)$ may be coarsely estimated using generic classification tools. This avoids the need to find a good density estimation algorithm. Of course, the density can be recovered from the classifier using the approach in [14]:

$$\hat{P}(x) = \frac{\text{Extrap}(x)U(x)}{1 - \text{Extrap}(x)}.$$

The use of a score for extrapolation is advantageous over a simple classification for several reasons. To begin with, it provides a sense of the relative density of training points - a handful of very sparsely distributed points in some region may increase the confidence placed in a prediction, but not as much as having many training examples nearby. A confidence score can be used when searching over possible actions to weigh the potential gain from each action. In Chapter 7, I show that it is possible to design diagnostic tools that make use of a score to provide low dimensional plots which capture more behavior than a simple classification of outliers. Of course, when such a classification is required, a simple cutoff can be applied.

[1]The confidence placed in predictions, of course, must be tempered with the overall ratio of the number of points to number of dimensions. This can be done directly for a uniform distribution, treated like a prior; the measurement here represents a refinement of that confidence.

5.2 Confidence and Extrapolation Representation Trees

I propose to estimate (5.1) directly via a classification tool. In particular, tree-based methods provide a useful framework in which to do this. Particular advantages of this approach include

- Interpretability, and in particular an interpretable set of regions in which to display summary diagnostics for functional behaviour. This is demonstrated in Figure 5.3
- The resulting regions are hyper-rectangles. For more sophisticated diagnostic tools, the leaves of the tree can be turned into product measures, providing a mixture-of-products approximation to the data density.
- Trees are both computationally efficient and a known and accepted part of machine learning, making the understanding of a new diagnostic tool easier.
- Trees can take a known distribution as an argument. In particular, an empirical distribution can be classified against a theoretical distribution. This provides a marked improvement in the resulting estimation.

I have labeled this use of trees as Confidence and Extrapolation Representation Trees, or CERT.

5.2.1 Monte-Carlo Data and the Curse of Dimensionality

An initial proposal might be to simulate a random sample from a uniform distribution and then use CART [3] to classify the two data sets. This approach, however, turns out to produce highly variable measures. Moreover, in high-dimensional situations in which real data occupies a space of small Lebesgue measure, a tree is often

able to exactly separate the real data from a Monte-Carlo uniform sample long before it captures the distribution of the real data. This situation has the potential to leave regions of extrapolation marked as regions of confidence.

By way of illustration, consider N points drawn from a multivariate "porcupine" distribution, P_1, defined by the natural extension of the base distribution in Figure 4.1 to d dimensions, placing an arm along each axis. Following the demonstrations in §4.2.3 the "arms" are each of length 2 and width 1. We leave an L-shaped marginal on each pair of predictor variables as in Figure 4.3. We will formally define this as the set

$$\left\{ x : \sum_{i=1}^{d} I(x_i > 1) \leq 1 \right\} :$$

only one of the coordinates may be greater than one.

The support of P_1 has $U(0,2)^n$ Lebesgue measure $d+1$ and requires $2d(d-1)$ splits to define. Now let us define a larger density P_2 by taking P_1 and selecting a subset u of k variables, and using their maximum in the definition above. In other words either the maximum of the x_u is greater than one, or at most one of the other variables is. Formally, P2 is defined by

$$\left\{ x : I(\max_{i \in u} x_i > 1) + \sum_{j \in -u} I(x_j > 1) \right\}.$$

The area covered by P_2 has Lebesgue measure $2^k + d - k$. Following the Monte-Carlo strategy of classifying N points from P_1 away from N uniformly distributed points, the probability of defining P_1 with a tree after getting as far as P_2 is the probability of some uniformly distributed random point falling in the support P_2 but not P_1, or

$$1 - \left(1 - \frac{2^k(d-k+1) - d - 1}{2^d} \right)^N.$$

N must scale linearly with d for this probability to remain constant. However, there are $\binom{d}{d-k}$ alternative distributions that can be defined this way, so the probability of producing a tree that does not describe all the detail of P_1 is large.

The important observation here is that the support of P_2 includes a region that is assigned large confidence values, despite having zero probability mass. Note that it is possible for points in this region to be $2^{k/2}$ distance from each other. So the probability of missing significant extrapolation is large.

A similar analysis can be performed for the real data in this situation, as this also must scale by N to ensure that the density is not empty above the value $1/2$ in some variable. However, for a conservative estimate, this is a smaller problem. Additionally, since we are considering extrapolation on a function learned given the predictor distributions, it makes sense to call this extrapolation for the data that we have, even if there is a high probability of a different training set placing weight there.

Note that while this demonstration is given for trees, it can be made to apply to other classifiers. The essential issue is that for high dimensional data, a Monte Carlo sample of the same size can be too-easily separable from the training data. This problem can be alleviated somewhat by including more Monte Carlo points and giving each a correspondingly lower weight. However, for the distribution described above, the number of points still needs to increase at a combinatorial rate, quickly making the approach infeasible.

5.2.2 CART and Distributional Information

Trees are capable of taking an exact distribution as an argument instead of an empirical data set. For each split, we merely replace the number of Monte-Carlo uniform data points on each side of the split with the expected number. This both reduces sources of variance in the original data and allows a much finer approximation to the empirical data density.

To illustrate this point, the results of a simulation are reproduced in Figure 5.1. I simulated 50 data sets of 1000 points with a Gaussian distribution in 5 dimensions. I then used CART to classify them away from another 1000 points, uniformly distributed on the empirical range of the data. A tree was also built for each of the data sets using the technique described above. Figure 5.1 compares their accuracy, stability, and resolution on a further set of 1000 normally and 1000 uniformly distributed points.

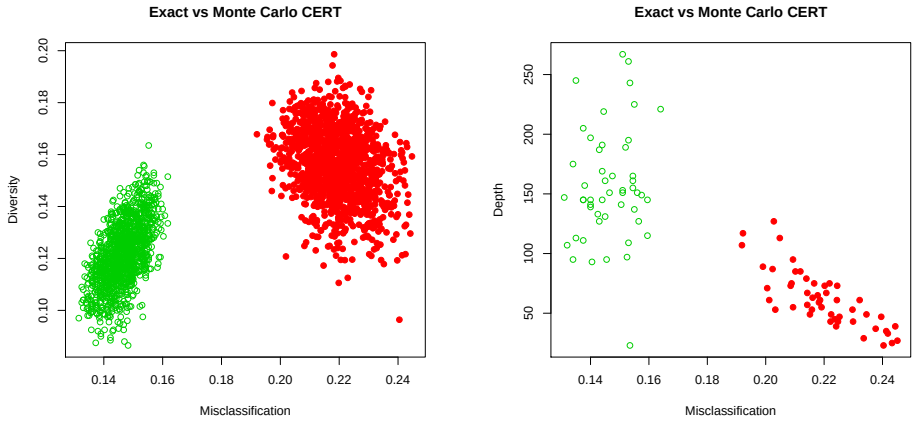

Figure 5.1: Performance for CART with Monte Carlo uniform samples (solid) and with uniform expectations (hollow). Average accuracy is plotted on the x axis for pairs of trees against diversity on the y axis, measured by the proportion of points classified differently (left). Right: Misclassification accuracy is plotted for individual trees on the x axis against size of tree given by number of nodes on the y axis.

For these instances the trees were pruned in a standard manner - using the 1 standard error rule and cross validation for the standard CART trees. A separate test set of normally distributed points along with the expected uniform points was used to prune the trees incorporating distributional information. It can be seen from these experiments that the use of distributional information improves the trees in all three dimensions, providing improved classification performance, greater classification stability and greater resolution.

5.3 CERT Details

There are several components to the CART algorithm, not all of which are necessarily appropriate in this setting. This section provides a list of specific criteria included in CERT.

5.3.1 Splitting Criteria

All tree-building algorithms use a greedy search strategy, picking the next split on some score. CART uses the Gini index, also employed in CERT. C4.5 [24] uses binomial entropy, which is equally valid.

5.3.2 Pruning

The CART procedure builds a large tree and then "prunes" it - removing lower splits which do not increase estimated predictive accuracy. In CART the amount of pruning done is chosen based on cross-validated misclassification error. The trees used here have been pruned based on test-set misclassification error, again using distributional information for the uniform distribution misclassification rate.

Alternative scores can be employed for different purposes. If an estimate of density is required, then using binomial entropy calculated by a test set may provide more accuracy. Empirically, that strategy leads to larger trees. Alternatively, a measure of independence between variables may be useful for developing a representation that can be used with diagnostic tools.

A final possibility for some applications is the C4.5 rule-pruning strategy. This does not maintain the tree structure. However, the density model produced from this approach now takes the form of a mixture of overlapping uniform distributions, also indicating a potential fuzzy clustering.

5.3.3 Missing Values and Surrogate Splits

When missing values are encountered on new data which are required at some split, CART tries to determine which branch to follow based on correlations between predictor variables. This uses a technique called surrogate splits. CERT diverges from CART on this issue and I employ the strategy of C4.5. Suppose that $x_{\{i\}}$ is missing, then the natural measure of extrapolation should be

$$\text{Extrap}(x_{\{-i\}}) = \frac{P(x_{\{-i\}})}{P(x_{\{-i\}}) + U(x_{\{-i\}})}$$

This can be written alternatively as

$$P(D|x_{\{-i\}}) = \sum_{k=1}^{K} P(D|L_k)P(L_k|x_{\{-i\}}),$$

where D is an indicator that x was generated from the data distribution and $\{L\}_{k=1}^{K}$ represent the terminal nodes of the tree. Now observe that $P(L_k|x_{\{-i\}})$ is exactly the weight given to each leaf if the tree is traversed according to the C4.5 strategy - going left and right at splits involving $x_{\{i\}}$ with probability equal to the proportion of total (real and uniform) weight in that direction.

5.3.4 Priors and Loss Functions

The quantity Extrap(x) as given in (5.1) assumes equal priors on the uniform and data distributions. Theoretically, this does not harm the accuracy of the procedure. If Extrap(x) is taken from known distributions, then changing the prior weight on the data distribution corresponds to a monotone transformation of Extrap(x); there is a one to one correspondence between cut-off values of Extrap(x) – say for the purposes of outlier detection – and prior weights.

Priors can make a difference in CERT, however. Here they are equivalent to

changing the values in the loss matrix and this in turn can affect the splits that are chosen in the learning process. Placing more weight on a uniform distribution will result in a tree that aggressively attempts to find regions with no real data. Correspondingly, up-weighting the real data will result in a tree that seeks regions of high data density. If the problem is, for example, outlier detection, priors should be chosen to reflect the relative importance of misclassifying outliers and training data.

5.4 Outlier Detection

An obvious first use for any measure of extrapolation is as an outlier detection device. I simply call a point x an outlier if Extrap(x) is less than some constant C. Figure 5.2 compares the performance of CERT with the common technique of excluding points based on their Mahalanobis distance based on the training covariance from the mean of the training data. We have chosen two distributions to test this. The first is a 5-dimensional Gaussian with a tri-diagonal correlation matrix that takes 0.3 on the non-zero off-diagonals. Mahalanobis distance should be optimal for this situation. The second is a 5-dimensional extension of the distribution from §2.3 - an "arm" extending along each axis, which should be well-described by CERT. I used 200 training examples and outliers were generated by a uniform distribution. In order that the performance of the two methods be comparable, I have drawn new samples from the training distribution and plotted the percentage of misclassified "outliers" against the percentage of misclassified "real" examples.

From Figure 5.2, Mahalanobis does, as expected, outperform CERT on a multivariate Gaussian. This discrepancy becomes larger as the correlation between variables becomes stronger. CART allows splits to be made on linear combinations of variables – although that option is rarely used. It seems reasonable to speculate that employing such splits with CERT might improve CERT's performance. For an independent Gaussian distribution – also provided in Figure 5.2 – the performance of

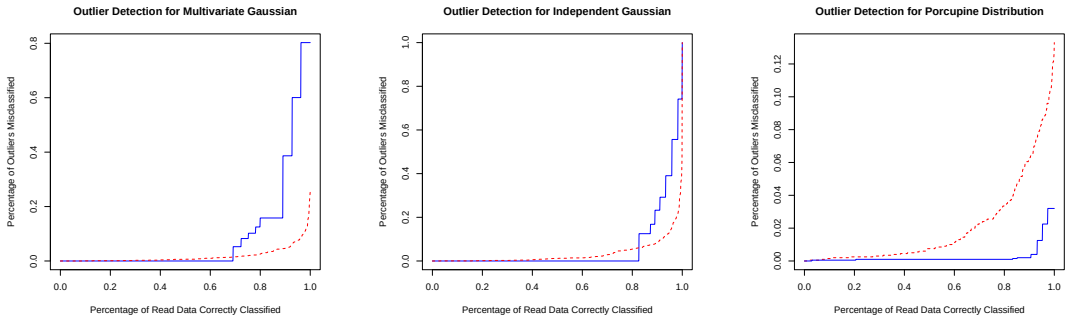

Figure 5.2: Performance of outlier detection techniques. CERT (solid) and Mahalanobis distance (dashed) on a multivariate Gaussian (left), an independent Gaussian (middle) and porcupine distribution (right). When the contours of the distribution of data is non-convex, CERT has a significant performance advantage.

the two methods is much closer. Having more training examples also improves the relative performance of CERT. In the second distribution, however, it is quite clear that CERT outperforms Mahalanobis distance; any ellipsoid that tries to span the training data in this case inevitably contains large amounts of empty space.

Although I do not present results here, given a measure of extrapolation, concept drift can be measured by the amount of extrapolation occurring. If the sum of extrapolation scores taken over some period shows an increase beyond an acceptable level, there is good reason to retrain the model. CERT can also be used in this context to indicate into which regions the underlying predictor distribution is moving.

5.5 Descriptive Statistics

Beyond the utility of being able to detect extrapolation well, diagnostic tools can also be helpful in understanding where the training data lies and the associations that exist between variables. A large factor in the choice of trees as a classification tool is that they provide an interpretable set of regions in which to examine the behavior of a

learned function. These regions remain high-dimensional and do not admit any easier display of a function. However, they do allow a summary of the behavior of a function within that region. Descriptive statistics such as function means and variances on each region as well as the proportion of original data points in a region provide an assessment of how serious extrapolation may be. In particular, diagnose areas where large Gibbs effects occur and allows the function to be artificially smoothed, or predictions within those regions to be flagged as suspect.

In Figure 5.3 below, I present the graphical representation of the underlying predictor space for the Boston Housing data and diagnostic statistics for a 12-20-1 Neural Network trained on these data. In particular, for each leaf l, I report four numbers in order:

- The number of real data points in the leaf, N_l.
- The expected number of uniform points in the leaf, U_l.
- The mean value of the prediction function evaluated on a uniform sample in that leaf, μ_l.
- The variance of the function evaluated on the same uniform sample, σ_l^2.

We would then be concerned about predictions in leaves with a small number of real data points. This concern would be strengthened if we observe a large variance or an extreme mean in that leaf. Such concern incorporates a heuristic belief that predictions are likely to be highly variable – under resampling of the training data – in regions where $\hat{F}$ changes a great deal without data supporting such variation.

In this context, it is tempting to produce a pseudo-prediction interval:

$$F(x) \pm \Phi^{-1}(\alpha/2)\frac{\mathrm{Var}_{\Omega_x}(F)}{N_x|\Omega_x|}. \tag{5.2}$$

Here $\mathrm{Var}_{\Omega_x}(F)$ represents

$$\mathrm{Var}(F(x)|x \sim U[\Omega_x])$$

where Ω_x is the leaf of the CERT model that contains x and N_x represents the number of training samples in that leaf. For differentiable F, this interval would formalize a belief that

$$\mathrm{Var}_{\{x_i\}_{i=1}^N}\left(\hat{F}\left(x;\{x_i\}_{i=1}^n\right)\right) \sim \frac{||\frac{d}{dx}F(x)||^2}{NP(x)},$$

which is assumed to be approximately constant in each leaf. (5.2) is then a δ-method estimate of this quantity.

Despite the intuitive appeal of this approach, the variance of $F(x)$ with respect to resampling the training data depends crucially on the model assumptions and fitting procedures involved in producing F and such an estimate could be highly misleading.

5.6 Example: Boston Housing Data

I used the Boston Housing Data to create a representation of the underlying predictor space, leaving out a test set of size 50 for pruning purposes. The resulting tree is reproduced in Figure 5.3. Reported in the leaves of the tree are the means and variances of a 12-10-1 neural network trained on the data using the default values of the `R` package `nnet` [25]. Here it can be seen that the fourth leaf from the left contains almost all the data mass. Concerns might be raised about predictions in the fifth and last leaves as having no data mass and relatively low values. The first leaf, too, is problematic in having very low data mass, yet a very high variance.

In order to capture some idea of the representativeness of the tree, Figure 5.4 provides a matrix of scatter plots of the data on bivariate axes along with contours of the bivariate marginal densities as calculated in the tree-based representation. For the sake of an alternative depiction of nonlinear correlation structure, the plots are

given with Lowess curves fitted for the predictors on the upper diagonal. CERT can be seen to capture some, though not all, of the bivariate associations. Part of this is due to data fragmentation – an individual leaf may not have enough data to be distinguished from a uniform distribution.

Comparing CERT and Mahalanobis distance to detect uniform outliers found that the probability of "outlier" misclassification differed by at most 0.004 for any value of Type 2 error. An examination of Figure 5.4 indicates a combination of approximately linear relationships, with some of the form of Figure 4.1.

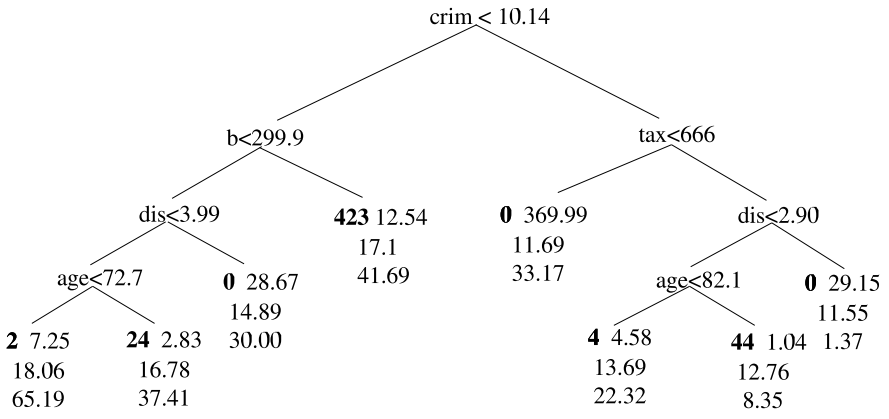

Figure 5.3: Tree-based density model for the Boston Housing Data with a 12-10-1 neural network.

5.7 Conclusions

CERT provides a new tool in diagnosing unusual points. I have defined a natural measure of extrapolation as being the relative likelihood of the data distribution versus a uniform distribution. This measure may be efficiently estimated with a variant of CART, and I have demonstrated that the inclusion of exact distributional information aids both the accuracy and stability of the resulting estimate.

This tool can now be used in several settings. To begin with, it is a straight forward diagnostic tool for high-dimensional covariance structures in a set of predictor variables. It also aids our understanding of function dynamics in areas of low data density. The measure can be used as an outlier-detection device: rejecting any data point that is in a region of low probability. It is also a diagnostic for shifts in system dynamics. The mixture of products representation of the data density implied by CERT can be used to create diagnostic tools as is shown in Chapter 7.

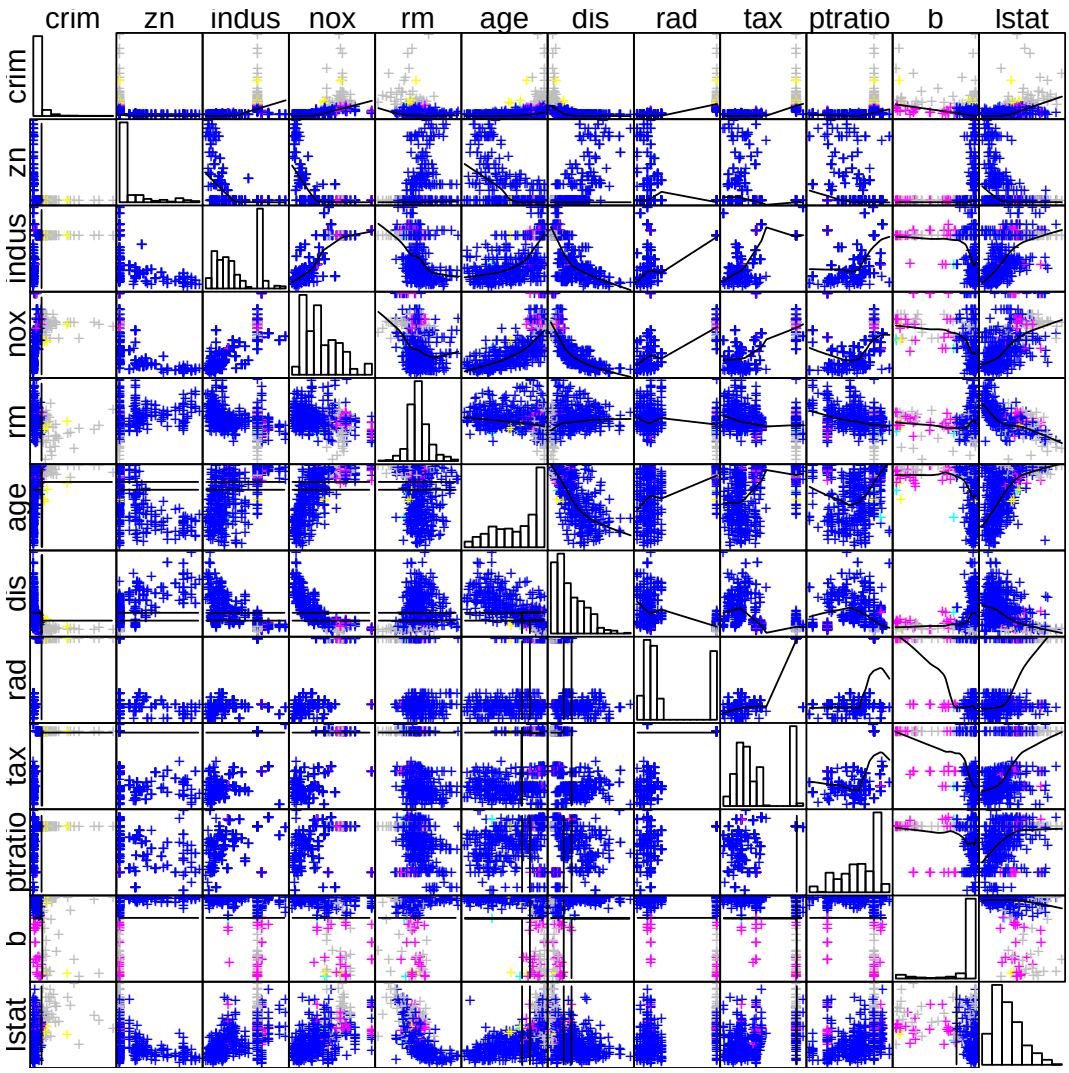

Figure 5.4: A plot of marginal density contours and points (lower triangle) and points with Lowess curves (upper diagonal) for the Boston Housing Data.

Chapter 6

Data-Augmented Regression for Extrapolation

The tools demonstrated so far in this thesis are concerned with diagnostics and extrapolation. In this chapter I will deal with the problem of making predictions at points of extrapolation. Given that we know that the next new example is a point of extrapolation, how are we to make a prediction for it? Ideally we should decline to do so. This, however is not always possible. It is possible, however, to stabilize the predictions that we give by reverting to a prior, stable base model as points become further from known examples. I show how that can be achieved through a process of adding data to the training sample to influence the predictive function in this direction and suggest that for most purposes a constant represents an appropriate base model.

The main motivation for this technique comes from the problem of extrapolation and I focus on this problem, believing the usefulness of regularization to be well understood. Nonetheless, I show that the techniques I present often lead to an improvement in predictive accuracy on the data distribution. §6.1 makes an argument for reverting to a simple null model as we are asked to evaluate a learned function further away

from training data. §6.2 presents a method of augmenting training data with simulated data designed to encourage a learner to make this reversion automatically. This technique is compatible with any learning procedure. In §6.3 I examine the connections between this technique and existing methods. §6.4 will examine some practical issues in choosing parameter values. §6.5 demonstrates the method in practice.

6.1 Heuristics for Prediction at Extrapolation

My purpose in this chapter is to propose a sensible approach to making predictions at points of extrapolation. I argue that, in the context of learning a function with a universal approximator, the further a point is from training data, the less information the function has about the distribution of responses at that point and the more it is guided by details of its learning process. Under these circumstances, it is appropriate to revert to some known "base model", $m(z)$, at points far away from the training data. The most simple of these is a constant; in the context of squared error loss this translates to the mean response which I generally assume to be zero.

Where prior knowledge about system dynamics is available, using a simple, highly-structured model rather than a constant as a null may be practical. Care should be taken in choosing these, however; if our prior knowledge about model structure pertains in some small region of predictor space, applying that structure may still lead to prediction functions that extrapolate poorly. I therefore believe that strong prior knowledge is needed on the whole space before discarding a constant base model.

Bayesian estimates can often be thought of as a form of regularization and this is true of these ideas as well - shrinking predictions toward a more stable model. In terms of the bias-variance trade-off, this can also produce models with improved predictive accuracy as demonstrated in §6.5.

6.2 Data-Augmented Regression for Extrapolation

The requirement that a function return to a base model away from training data has not been implemented as part of the learning procedure in most universal approximators. However, this behavior can be achieved by stochastically generating uniform data with response given by the base model and adding it to the training data. The resulting procedure, which I term Data-Augmented Regression for Extrapolation (DARE) can supplement any regression method. A formal description is given in Algorithm 6.2.1.

Algorithm 6.2.1. *DARE*

Inputs: *Regression algorithm $\mathcal{L}$, training data D, bounding region in predictor space $\mathcal{R}$, background model $m : \mathcal{R} \to \mathbb{R}$ (typically $m \equiv 0$), total "weight" for augmented sample s.*

Output: *Prediction function $f : \mathcal{R} \to \mathbb{R}$*

1. *Draw an "augmentation sample" U of size N_u with total weight s as follows:*
   (a) *Draw N uniform points in $\mathcal{R}$, denote them $z_1, ..., z_{N_u}$*
   (b) *Assign to each z_i response $m(z_i)$*
   (c) *Give each observation z_i weight s/N_u.*
2. *Apply $\mathcal{L}$ to $D \cup U$ and output $f = \mathcal{L}(D \cup U)$.*

6.3 Connections

6.3.1 Gaussian Process Priors

DARE can be thought of as placing a random field prior on the prediction values $f(x)$ with a covariance function that is zero away from the origin. Using the Monte

Carlo population in place of the "large" sample, and assuming that L uses a cost function c, we can write this cost as:

$$\sum_{i=1}^{N} c(f(x_i), y_i) + \gamma \int_{\mathcal{R}} c(f(x), m(x))dx. \tag{6.1}$$

More precisely, as $N_\mu \to \infty$, the cost associated with any estimate f tends to (6.1) by the strong law of large numbers.

For squared error loss, this translates to a maximum *a posteriori* estimate for f with $y \sim N(f(x), \sigma^2)$, and a Gaussian field prior $f(x) \sim N(m(x), 1/\gamma)$. Different loss functions imply different priors, for example, using absolute loss implicitly assumes a double exponential field prior.

6.3.2 Generalized Ridge Regression

DARE can be viewed as a non-parametric generalization of ridge regression when $m(x) = 0$. In a linear regression context, ridge regression uses a coefficient vector β that minimizes

$$\sum_{i=1}^{N} (y_i - x_i^T \beta)^2 + \lambda \sum_{j=1}^{d} \beta_j^2 \tag{6.2}$$

which has been shown to provide superior predictive performance [14]. Ridge has a Bayesian interpretation, the solution to (6.2) being the maximum a posteriori estimates for β with $y_i \sim N(x_i^T \beta, \sigma^2)$ and β having prior $N(0, \sigma^2/\lambda)$.

It is easy to see that DARE corresponds to ridge regression in a when a linear function is substitute in (6.1) and the uniform distribution is centered on the mean of the training data. A little algebra gives that the second term of (6.1) becomes:

$$\int \left(\sum_{j=1}^{d} x_{\{j\}} \beta_j \right)^2 dx_{\{1\}} \dots dx_{\{d\}} \propto \sum_{j=1}^{d} \beta_j^2,$$

the ridge regression penalty. Ridge can also be viewed as adding data to the linear optimization problem. The penalty term in (6.2) can be re-written

$$\lambda \sum_{j=1}^{d} \beta_j^2 = \sum_{j=1}^{d} (0 - \lambda e_j^T \beta)^2$$

where e_j is 1 in the jth coordinate and zero elsewhere. Thus ridge is equivalent to augmenting the original data with $\lambda e_1, \ldots, \lambda e_d$ and giving these extra records response 0. I view this as exploiting the linearity of f to achieve exactly what DARE targets stochastically.

6.3.3 Kriging and Reversion to the Mean

The reversion to a mean response behavior is already achieved by the techniques of Kriging; viewing a function as a random Gaussian process [6]. The use of Kriging in machine learning has been suggested in, for example, [32]. Placing regression in a Gaussian process setting provides a useful theoretical framework in which to discuss regression. The responses are regarded as being Gaussian, but with a known covariance structure that depends on the predictors. This allows Kriging to be viewed as a subset of kernel methods, but also provides uncertainty estimates for prediction and, particularly a reversion to the response mean - the marginal distribution for the response of a new point is Gaussian about the response mean, and the posterior trends back to that as the covariance vector between the new response and known examples tends to zero. Thinking of (6.1) as incorporating a prior in the same manner as ridge regression provides an analogous framework. Kriging can also incorporate a reversion to some more complicated null model in a similar manner to DARE.

Kriging has the same disadvantages as kernel methods: computational cost and the choice of a variogram or covariance function. The variogram does allow prior knowledge of covariance structure to be incorporated into the model, but must be specified arbitrarily if such knowledge is unavailable. The cost of training the model

is $O(N^3)$ in the number of data points, and $O(N)$ for each prediction. DARE may be regarded here as substituting an alternative learning procedure both as a pseudo-estimate of covariance and in providing faster training and prediction.

6.3.4 Prior Information, Virtual Examples and Classification

The concept of adding data has been advocated in [21] in the context of classification. Here, prior knowledge about regularity conditions in the data – rotational symmetry, for example – can used to create "virtual examples" to be added to the data set. The sorts of prior information discussed there are highly dependent on the learning context and are often less obviously available for regression problems. They do not discuss classification strategies that lack prior knowledge.

Prediction variance at points of extrapolation is, of necessity, constrained in a classification setting, although examples like those in Figure 4.2 can still be easily constructed. A DARE approach would take one of two forms; uniform data could be added with a new "don't know" label to indicate that this is not a point that can be safely classified. If classification predictions are required, the DARE philosophy suggests that a "don't know" output should revert to a decision that is uninformed by predictor variables: predicting the most numerous class.

6.4 Parameter Values and Rules of Thumb

The relative weight of D and U in Algorithm 6.2.1 governs the extent to which learned functions are allowed to extrapolate away from the base model $m(x)$. I assume the procedure $\mathcal{L}$ accepts weights on observations. In this case, the size N of the uniform sample should be as large as computationally feasible, to best emulate a uniform prior distribution (see §6.1 and §6.3 for motivation). The balance between D and U is achieved through the weighting factor s/N. If $\mathcal{L}$ cannot take observation

weights, this trade-off has to be governed directly by N. In order to cover the space well, it may be necessary to average the process with repeated resampling of U.

Methods without Observation Weights

An alternative to averaging DARE with repeated samples from U, if the learning process is already computationally expensive and L does not take observation weights is to observe that if F minimizes

$$\int c(F(x), y) dP(x, y)$$

then the minimizer of (6.1) is given by

$$P(D|x)f(x) + (1 - P(D|x))m(x)$$

where $P(D|x)$ is the classification probability of x being generated by the data distribution as opposed to the uniform distribution, with prior weight determined by γ. For $\gamma = 1$ this exactly translates to the measurement Extrap(x) from Chapter 5. It is then possible to learn $\hat{F}$ on the training data in the usual manner and to estimate $P(D|x)$ – for example, by CERT – and to combine these two quantities. If $\hat{F}$ suffers large Gibbs effects, this strategy will be less effective at dampening them than DARE. Nonetheless, if DARE is not feasible, this may provide a partial answer.

Amount of Monte Carlo Data

How much Monte Carlo data is sufficient depends on the particular learning procedure used. A very conservative rule of thumb would be to use some estimate of the maximum density of training points, if one is available. It seems unnecessary to provide a greater density than this uniformly across the range of the data. For high dimensional examples, however, that goal may still be infeasible.

Prior Weights

A good value for the relative weight of true and Monte Carlo points can be estimated by cross validation. Since this weight affects test-error performance as well as stability, re-sampling the uniform distribution should be done along with standard cross validation. The results in §6.5.1 and §6.5.2 provide separate estimates of stability and test error (the error on Monte Carlo uniform test points estimates pointwise stability). The two can be combined and weighted to represent relative importance of prediction and stability, and thus to select an "optimal" relative weight.

DARE and Model Structure

In some situations regularization may not be necessary for prediction purposes and may actually be harmful. This is typically the case in highly structured learning situations, where the size of the model space considered (via VC dimension or degrees of freedom) is reasonably small compared to the training data available. DARE may still be useful for controlling extrapolation in such cases, and we are then faced with a choice between prediction on the "standard" data and control of extrapolation.

6.5 Experiments

In §6.5.1 and §6.5.2 I present the results of using DARE on simulated and real world data. In each, I employ DARE with the RPART implementation of CART and with "quadratic regression" – using all linear and quadratic terms as basis functions in a least-squares fit. In each case, the augmentation sample had size 5000. I graph test error and variance against the log of the relative weight on Monte Carlo samples – increasing weight providing more regularization. Solid lines provide test error on independent test sets, plotted with 95% confidence intervals. Dashed lines are the mean pointwise variance on a further 5000 uniformly distributed points.

All the plots show a region of relative weight where predictive performance is not hurt, and is often helped, by DARE. At the same time, within these regions I am able to achieve a significant improvement in the over-all stability of predictions as the relative weight of the augmentation sample increases.

6.5.1 A Simulated Example

I examine a simulated example in which true probability distributions are known. I take as predictor variables 500 examples of a 30 dimensional mixture of two Gaussians:

$$(x_1, \ldots, x_{30}) \sim 0.5N(\mu_1, I) + 0.5N(\mu_2, I)$$

in which μ_1 has value 1 in the first fifteen entries and zero in the second fifteen and μ_2 is zero in the first fifteen entries and 1 in the second. The response is given by the linear sum

$$y = \sum_{i=1}^{15} x_i - \sum_{j=16}^{30} x_j + \epsilon$$

with ϵ iid $N(0, 1)$. There are two groups; the first having largely negative response and the second largely positive. I repeated the experiment 20 times to form variances and give the results in Figure 6.1. For both CART and quadratic regression, the improvement in predictive accuracy is statistically significant.

6.5.2 DARE and Boston Housing Data

To provide a real-world example I used the Boston Housing Data [12]. In this case I again employed CART and quadratic regression. Cross validation was undertaken using a test-set sample of size 100 randomly drawn each of 100 times. The Boston Housing Data was selected as exhibiting structure similar to Figure 4.2, creating severe extrapolation. For flexible methods, this results in stabilization at points of extrapolation more than regularization on the data distribution, as can be seen in a

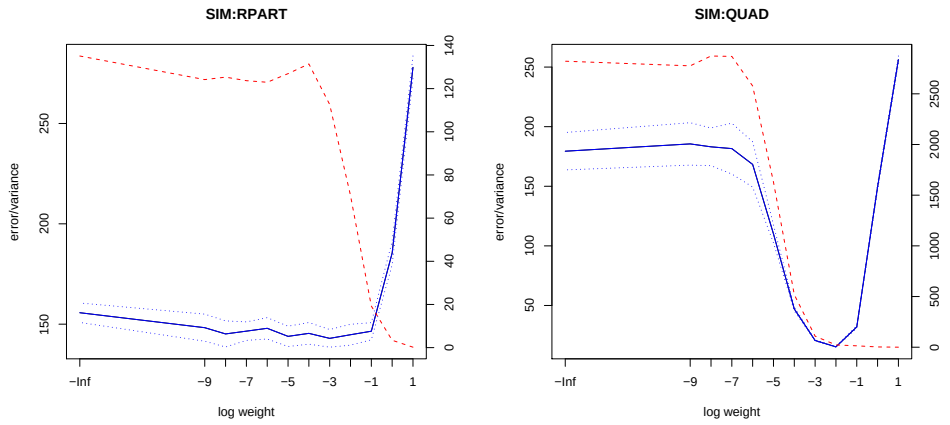

Figure 6.1: DARE results on a simulated data set using CART (left) and quadratic regression (right). Solid lines give test error performance as a function of relative weight given to the added data, dotted lines around it provide 95% pointwise confidence intervals. Dashed lines are mean pointwise variances for a uniform distribution over twenty resamplings. Variances are plotted with respect to the axis on the right hand side.

less significant improvement in predictive accuracy. Nonetheless, some improvement in predictive performance can be seen and DARE can provide much more stable extrapolation without harming performance.

6.6 Conclusions

In this chapter I have considered the problem of making predictions at points of extrapolation. Few learning procedures are designed to produce stable results at points of extrapolation; even constant extrapolators can exhibit high variance away from training data. In order to stabilize these predictions and recognize their semi-arbitrary nature, I propose that predictions should be shrunk toward a base model in proportion to the density of training points near them. This follows the heuristic argument that as new examples get further away from known examples, the model

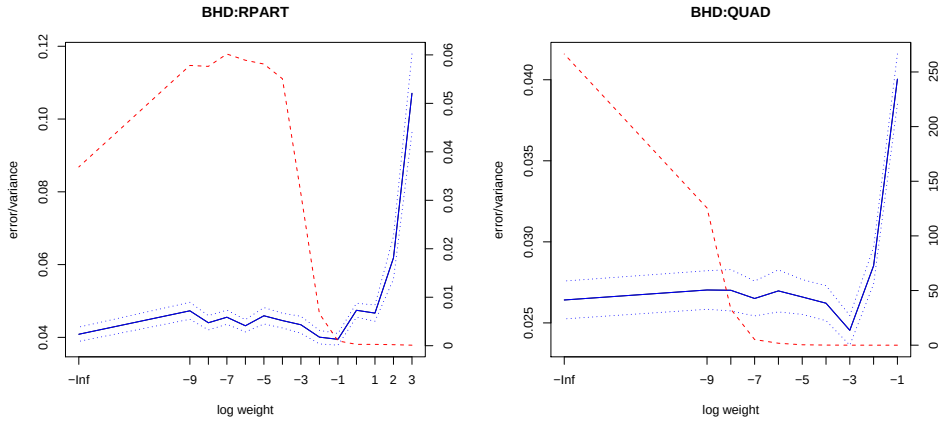

Figure 6.2: DARE results on the Boston Housing Data using CART (left) and quadratic regression (right). Solid lines give test error performance as a function of relative weight given to the added data. Dashed lines are mean pointwise variances for a uniform distribution on the data range over 100 resamplings. Variances are plotted with respect to the right-hand axis.

predictions becomes less informed about the response.

In order to carry out this shrinkage, I propose a very simple procedure of generating new uniformly distributed data, giving it the response associated with the base model and augmenting the training set with this data. Unless strong prior knowledge is available that pertains to the whole space, I recommend that an appropriate base model should be constant. This idea has the advantage that it can be applied to any learning method. Viewed from a Bayesian perspective, this method amounts to placing a random field prior on prediction values, basing predictions on a null model unless empirical data – in the form of nearby training data – provides evidence to the contrary. I also show that when linear regression is employed, the method is a stochastic form of ridge regression.

The extent to which DARE regularizes depends on the flexibility of the learner that is employed and the concentration of the training examples in predictor space. These are the factors that also influence the extent of our concern about extrapolation.

Regularization often also has a positive effect on predictive accuracy. I have demonstrated on simulated and real examples that it is possible to simultaneously improve predictive accuracy on the data distribution and stability at points of extrapolation.

Chapter 7

Extrapolation-Resistant Diagnostics

Chapters 2 and 3 have already explored the use of the Functional ANOVA in providing diagnostic aids to accompany machine learning procedures. There it was shown to provide a desirable set of properties for low-order effects; optimality of the effects, additivity preservation, and comprehensibility. I have noted in Chapter 4, however, that the Functional ANOVA relies on having a product measure as an underlying source for data, and that when this does not pertain the effects that are presented are subject to distortion that can be arbitrarily bad. In most cases, a product measure is not a reasonable representation of the distribution of the underlying predictors. These typically have complex, non-linear dependence structure, leaving large parts of space empty. When training data are drawn from this distribution, these holes in the data translate to regions of extrapolation. Here, functional behavior is dictated by the learning process rather than the training data. For "good" extrapolation, this may be close to constant, but many functions can exhibit Gibbs effects, and even constant extrapolation can exhibit high variance between different sets of training data. I have shown that by evaluating a prediction function on a product distribution, the

Functional ANOVA moves a large amount of probability mass into those regions of extrapolation, potentially swamping plots with Gibbs effects, or damping out true effects with disproportionate weight on areas where the function does not change.

In this chapter, I propose a generalization of the Functional ANOVA that can incorporate a non-product measure. One of the important criteria that I want to satisfy is that the decomposition must preserve additivity. The key to doing this is to represent the entire set of Functional ANOVA effects as satisfying a joint constrained optimality criterion. This constrained criterion can then be generalized to accept any measure.

7.1 Desirable Properties of A Functional Effect

Having illustrated the problems associated with current diagnostic tools and before embarking on developing new ones, it is important to ask what it is that a low-order representation of functional behavior should provide. I suggest four main properties below.

Comprehensibility

The plots should represent some understandable and appropriate quantity. Since any low-dimensional representation of a function necessarily removes information about functional behavior, in what sense such a representation captures that behavior should be well understood and appropriate. One approach to doing this is through an explicit optimality criterion. In the case of the Functional ANOVA, effects give the projection of the prediction function onto a subset of the variables.

Optimality

A low dimensional representation should capture as much of the functional behavior as possible, in some well-defined metric. We would like to understand as much of

the function as possible through visualizable effects; we would also like to know how much of the function is not explained by these effects. However, this can model a function too closely: low dimensional plots should provide this information in aggregate, leading to the criterion of additivity. Functional ANOVA effects are the closest $\mathcal{L}^2$ approximation to the full prediction function using only the effect variables.

Additivity

A function that is truly additive in some set of variables should be exactly recoverable in terms of the additive components. Here, plots of conditional dependence are not satisfactory in being too aggressive; they explain more behaviour individually than partial dependence plots but collectively provide a bad approximation to functional behavior. The variance decomposition of the Functional ANOVA ensures that additivity is preserved.

Extrapolation-Resistance

As demonstrated, product measures move probability mass to points of extrapolation where we do not wish to measure, or place a large emphasis on, functional behavior. A good measure puts low weight on prediction values at these points.

I have commented that the Functional ANOVA fits first three criteria, and this property is managed through the use of a product measure. In some sense, so long as we are only interested in the learned function as given and are happy to accept its extrapolation, this is all that is needed provided we accept some discrepancy due to the distortion of the underlying probability measure. However, if the dynamics of the system in question *per se* are the point of interest, then this can provide very misleading representations. As I show below, the solution lies in estimating the effects jointly, rather like the solution to the linear least squares problem.

7.1.1 Optimality Criteria for the Functional ANOVA:

The Functional ANOVA effects f_u trivially satisfy the optimality criteria:

$$f_u(x_u) = \underset{g_u \in \mathcal{L}^2(\mathbb{R}^u)}{\operatorname{argmin}} \int \left(g_u(x_u) + \sum_{v \subset u} f_v(x_v) - F(x) \right)^2 dx.$$

In fact, all the terms in the Functional ANOVA can be jointly defined by a constrained optimization.

Theorem 7.1.1. *The effects of the Functional ANOVA decomposition jointly satisfy*

$$\{f_u(x_u)\}_{u \subset \{1,\dots,d\}} = \underset{\{g_u \in \mathcal{L}^2(\mathbb{R}^u)\}_{u \in \{1,\dots,d\}}}{\operatorname{argmin}} \int \left(\sum_{u \subset \{1,\dots,d\}} g_u(x_u) - F(x) \right)^2 dx \tag{7.1}$$

under the constraints

$$\forall u \subset \{1,\dots,d\},\ \forall i \in u \int f_u(x_u) dx_{\{i\} \cup -u} = 0. \tag{7.2}$$

I remark that these constraints have been changed to incorporate the integral over x_{-u}. This has no practical effect on the standard Functional ANOVA. However, in the generalization below, I need the marginal distribution on x_u in (7.2) for a solution to exist.

Proof. This relies on a calculus of multiple variations, provided here for the sake of demonstration. Further arguments of this type have been relegated to Appendix B. Assume $\{f_u\}_{u \subset \{1,\dots,d\}}$ satisfies the criteria above. For any $\{h_u \in \mathcal{L}^2(\mathbb{R}^u), \epsilon_u > 0\}_{u \subset \{1,\dots,d\}}$ that also satisfy (7.2) let

$$\{g_u \in \mathcal{L}^2(\mathbb{R}^u)\}_{u \subset \{1,\dots,d\}} = \{f_u + \epsilon_u h_u\}_{u \subset \{1,\dots,d\}}$$

gt for $\epsilon_u > 0,\ u \subset \{1,\ldots,d\}$. Then taking the gradient of (7.1) at $\{g_u\}_{u\subset\{1,\ldots,d\}}$ with respect to ϵ_u and evaluating at zero gives the equations: $\forall h_u \in \mathcal{L}^2(\mathbb{R}^u)$, $\forall u \in \{1,\ldots,d\}$

$$\int f_u(x_u)h_u(x_u)dx + \int \left(\sum_{u\not\subset v} f_v(x_v)\right) h_u(x_u)dx = \int \left(F(x) - \sum_{v\subset u} f_v(x_v)\right) h_u(x_u)dx.$$

The second term in the left hand side is zero by the conditions in (7.2) and the remaining equations provide the result.

This does not guarantee a unique minimum. The convexity of the problem follows from the general result of Theorem B.2.1 □

The generalization of this is now very simple; merely add a general finite measure $w(x)$ in place of the uniform measure. The criterion then becomes

$$\{f_u^w(x_u)\}_{u\subset\{1,\ldots,d\}} = \underset{\{g_u\in\mathcal{L}^2(\mathbb{R}^u)\}_{u\in\{1,\ldots,d\}}}{\operatorname{argmin}} \int \left(\sum_{u\subset\{1,\ldots,d\}} g_u(x_u) - F(x)\right)^2 w(x)dx \quad (7.3)$$

under the constraints

$$\forall u \subset \{1,\ldots,d\},\ \forall i \in u \int f_u^w(x_u)w(x)dx_{\{i\}\cup -u} = 0. \quad (7.4)$$

These constraints are that $f_u(x_u)$ must integrate to zero with respect to the *marginal* of w on x_u in each coordinate direction i in u, for all values of $x_{u\setminus i}$.

Appendix B provides regularity conditions necessary for the existence and uniqueness of solutions to this generalized criterion. Corollary B.2.1 states that a decomposition is unique for $w(x)$ that has support on an open set in $\mathbb{R}^d$. In fact, considerably weaker conditions are sufficient and I use these in the approximation schemes in §7.2. In the machine learning spirit of creating acronyms, I have titled this generalization

the Functional ANOVA for Measures of Extrapolation, or FAME.

A natural measure to choose in this case is the density function $p(x)$ governing the underlying predictor variable distribution. However, the existence of extrapolation, particularly in high dimensional situations, may suggest a function with support closer to the actual points observed. Even if such points are generated from a product distribution, in very high dimensional cases for a specific training sample, large holes can still appear in predictor space. It may be desirable to place lower weight in those regions. CERT provides one algorithm for doing this in a principled manner.

Note that the conditions (7.2) guarantee orthogonality for the standard Functional ANOVA. In this case some orthogonality is lost, and hence the variance decomposition is also lost. However, the new conditions (7.4) do provide hierarchical orthogonality:

$$\forall v \subset u : \int f_u(x_u) f_v(x_v) w(x) dx = 0 \tag{7.5}$$

since

$$\int f_u(x_u) w(x) dx_{-v} = 0 \ \forall x_v.$$

Equation (7.3) does not easily satisfy the criterion of comprehensibility. Further, it seems unrealistic to attempt to estimate all 2^d effects at once. However, using (7.5), we can write an optimization criteria for a single effect, defining $f_u(x_u)$ as

$$\underset{\substack{g_u(x_u),\, \{g_v(x_v)\}_{v\subset u},\\ \{g_{-v}(x_{-v})\}_{v\subseteq u}}}{\operatorname{argmin}} \int \left(\begin{array}{c} g_u(x_u) + \sum_{v\subset u} g_v(x_v) \\ + \sum_{v\subseteq u} g_{-v}(x_{-v}) - F(x) \end{array} \right)^2 w(x) dx \Bigg|_u \tag{7.6}$$

with $\operatorname{argmin}_{\{a_i\}_{i=1}^n} J(a_1, \ldots, a_n)|_k$ denoting the kth component of the optimal vector $a_1, \ldots, a_n$. Here the $\{g_{-v}\}_{v\subseteq u}$ are subject to the relaxed conditions

$$\int g_{-v}(x_{-v})w(x)dx_{-u\cup v} = 0.$$

These replace the set of conditions given for each $v \subseteq u$. Effectively, $\sum_{v\subset -u} g_v$ is subsumed into g_{-u}, treating x_{-u} as a single variable, and the constraints are relaxed accordingly.

Letting the size of u be k, the number of terms has been reduced to 2^{k+1}. Moreover, the effects can be grouped into two. Firstly, a projection, $\sum_{v\subseteq u} g_v(x_v)$ onto x_u that we would like to visualize and secondly, the remaining terms which act to control the effect of covariance. In this setting there is now a comprehensible optimality criterion: find the best fit of F on the space of functions with no higher interaction than u. Alternatively, like multiple linear regression, this is the effect for u when the correlation with other effects is taken into account. Further, an examination of the form of the functions being optimized in (7.6) shows that if F really does have an additive component F_u, then the criterion must exactly recover F, and therefore F_u. Finally, w guarantees that we do not measure spurious effects, the four criteria that were originally set out have been satisfied.

7.2 Product δ-measures and Pointwise Estimation

Equation (7.6) now presents a feasible number of terms to be estimated, at least for small k. However, 2^k nuisance functions of dimension up to $d-1$ still need to be estimated. The important realization here is that there is no interest in those functions except to control for the dependence structure in the underlying predictor space. It is therefore possible to confine ourselves to estimating all the terms above only at a carefully chosen set of points.

Attempting to estimate these functions at a single point is, nominally, an underdetermined task; assuming the points to be distinct in all dimensions, the function

can be recovered exactly at those points using only one predictor. However, the conditions found in Theorem B.2.1 provide an approach: if the functional effects are estimated on a grid of points, there is a unique decomposition.

Taking a grid with uniform weights is equivalent to approximating w by a product of empirical marginals - themselves sums of δ-measures. This is exactly the approach used for Partial Dependence Plots in Chapter 6. Now, however, each point in this grid can be given a differential weight according to w. Corollary B.2.2 states that so long as no point is left as the sole point with non-zero weight in any direction of the grid, (7.1) translates into a solvable linear system.

7.2.1 Estimation for Visualizing Effects

Start with a set of N points $\{x_i\}_{i=1}^N$ in $\mathbb{R}^d$ generated by a uniform distribution. These are generating points for a grid comprised of:

$$\{z_k\}_{k=1}^{N_{\{1,\dots,d\}}} = \{x_{i_1,u_1}, x_{i_2,u_2}, \dots, x_{i_k,u_k}, x_{j,-u}\}_{i_1,\dots,i_k,j=1}^{N} \tag{7.7}$$

where $x_{-u,j}$ is again treated as a single value. Appendix A gives a comprehensive account of the notation I use to describe grid values. (7.6) now translates into the problem of finding

$$\begin{Bmatrix} \text{argmin} \\ g_u(z_{i,u}), \{g_v(z_{i,v})\}_{v\subset u} \\ \{g_{-v}(z_{i,-v})\}_{v\subseteq u} \end{Bmatrix}_{i=1}^{N_{\{1,\dots,d\}}} \sum_{i=1}^{N_{\{1,\dots,d\}}} w(z_i) \begin{pmatrix} g_u(z_{i,u}) + \sum_{v\subset u} g_v(z_{i,v}) \\ + \sum_{v\subseteq u} g_{-v}(z_{i,-v}) - F(z_i) \end{pmatrix}^2 \tag{7.8}$$

under the constraints

$$\begin{aligned}
\forall v \subseteq u, \forall j \in v, \quad & \forall k \in \{1, \ldots, N_v\} : \\
& \textstyle\sum_{i=1}^{N_j} \left(\sum_{l=1}^{N_{-v}} w(z_{i,j}, z_{k,v\setminus\{j\}}, z_{l,-v}) \right) f_v(z_{i,j}, z_{k,v\setminus\{j\}}) = 0 \\
\forall v \subseteq u, \forall j \in -v \setminus -u, \quad & \forall k \in \{1, \ldots, N_{-v\setminus\{i\}}\} : \\
& \textstyle\sum_{i=1}^{N_j} \left(\sum_{l=1}^{N_v} w(z_{i,j}, z_{k,-v\setminus\{j\}}, z_{l,v}) \right) f_{-v}(z_{i,j}, z_{k,-v\setminus\{j\}}) = 0 \\
\forall v \subseteq u, \quad & \forall k \in \{1, \ldots, N_{-v\setminus -u}\} : \\
& \textstyle\sum_{i=1}^{N_u} \left(\sum_{l=1}^{N_v} w(z_{i,-u}, z_{k,-v\setminus -u}, z_{l,v}) \right) f_{-v}(z_{i,-u}, z_{k,-v\setminus -u}) = 0.
\end{aligned}$$

In order for this sampling scheme to provide reasonable approximations, I assume w to be a density. This is sufficient for all practical purposes.

Setting the gradient to zero, this results in a very large, very sparse, weighted linear system. To provide an idea of the practical size of the system, using a grid with N points in each dimensions, this system has N^k equations in $\sum_{j=0}^{k} \binom{k+1}{j} N^j$ unknowns and a further $\sum_{j=1}^{k} \binom{k+1}{j} N^{j-1}$ constraining equations. The effect values are the parameters of the system to be estimated and the values $F(z_i)$ take the place of the response. Solving such a system is computationally feasible for small k and moderate N. Given that we are interested in visualizing effects of order only 1 or 2, this is quite adequate. Note that where $w(z)$ is a product of univariate functions, this system exactly reproduces the equivalent estimation of Functional ANOVA effects, using the z_i as evaluation points. This can be seen by taking the grid of z's as point masses which form a product distribution. Theorem 7.1.1 then gives that the Functional ANOVA on this grid – exactly the empirical estimate of Functional ANOVA effects – solves (7.8).

The structure of this linear system allows for somewhat larger systems if an iterative method is employed. For large N or k, even storing $X^T X$ can be prohibitive. However, $X\beta$ and $X^T y$ can both be calculated efficiently using a sparse matrix representation of X. In this setting, conjugate gradient methods, which only rely on

the multiplication $X^T Xb$ can provide a significant storage saving. See, for example, [23] for a description of these methods and their properties. In Appendix C, I provide a description of the sparsity structure of the system and a calculation of the computational complexity of the problem.

7.2.2 Variations and Extensions

The full product grid on N^k points is also not necessary. Observe that identifiability only requires 2^k points on a product grid under the given constraints. The constructions above used grids of N points in each direction, but it is equally possible to use more points in some directions than others. A larger number of points in the variables x_{-u} may help reduce the variance of effect estimates, for example. The computational complexity of finding a solution can also be reduced significantly by taking the original set of points in smaller groups and only forming products among those groups.

A division into groups of grids decouples the estimation problem unless some points are shared between grids. This increases the amount of noise in the estimate of any individual function value, necessitating a smooth of the estimates when they are plotted. A simple alternative to this would be to employ a finite element approximation for the effect of interest, allowing the finite elements for the other terms to occur precisely as δ-functions[1]. This would allow significant computational saving in both requiring fewer finite elements than observations in the effect of interest, and allowing a more aggressive grouping of points in the control effects. Quasi Monte Carlo techniques can also be employed to produce a set of points that may reduce the variability of these estimates.

[1]The original estimation scheme is effectively this, but using δ-functions for the finite elements in the effect of interest as well.

7.2.3 Estimation for Interaction Importance Scores

While first and second order effects are sufficient for visualization, diagnostic tools for ANOVA structures such as those found in Chapter 3, do require importance scores to be evaluated for higher order interactions.

In this context, however, much less information is sought. The only quantity of interest is:

$$\bar{\sigma}_u^2 = \min_{\{f_{-i}\}_{i\in u}} \int \left(\sum_{i\in u} f_{-i}(x_{\{-i\}}) - F(x) \right)^2 w(x)dx.$$

That the terms on the right hand side are not uniquely specified is not of concern so long as the minimum exists and is estimable. Finding a minimum is generally possible for inconsistent linear systems such as this, even when that minimum not unique. In this case, with a noticeably larger k, the sparsity structure of the resulting linear system can be exploited. It is known that standard conjugate gradient algorithms may not converge for inconsistent systems. However, iterative variants for producing a solution do exist; see, for example, [5]. Again, the regularity of this system leads to the expectation of both fast convergence and very cheap storage.

Using the same estimation scheme as above, there are now $\sum_{i\in u} N_{-i}$ unknowns and one can again group points into smaller sets of product measures: each requires at least 2^k points. Using a small grid will naturally have a large amount of noise associated with the estimate. The importance estimate will be averaged across a decoupled collection of grids in this case, however, so some variance reduction can be achieved. This simplification allows a test of noticeably higher-order interactions, up to say $k = 7$. This, again, can be argued to be sufficient: there is little understanding to be gained by the knowledge that a function is the sum of some intrinsically 8-or-higher dimensional components.

An alternative estimation of effect importance is to simply fit an additive model

with ANOVA structure given by (3.2) to the training data, taking predicted values from the function as a response. The loss associated with this approximation is then an approximation to the $\mathcal{L}^2$CoE for FAME. A rough approximation with additive models may provide a more computationally efficient solution than the numerical linear algebra above for large interactions.

7.2.4 Non-Uniform Sampling Schemes

Using a base set of uniformly distributed points as the seed for the product δ-measure above suffers from the problem of poor representation in large dimensions. In particular, if a measure that is zero in large parts of predictor space is being used, there is a serious risk of leaving the normal equations for the empirical effects undefined by giving weight zero to too many rows. By effectively removing a large number of points from the estimation, the amount of noise associated with the estimate is increased, even when a solution can be found.

An immediate solution to this problem is to try to concentrate the points on regions of high density and adjust $w(x)$ accordingly. Since the grid points must lie on a product distribution, using the original data (or some subset of it) as a seed and dividing $w(x)$ by the product of its marginals is a first solution. This can be justified as being optimal in the sense that the product of marginals is the closest product distribution to the data in a Kullback-Leibler sense [14].

There is a disadvantage to this approach in that we may desire to see the effect plotted at points more uniformly distributed than the empirical marginal of the dimensions, x_u, in question. In this case, taking a uniform sample, or uniformly-spaced points only in x_u may be appropriate. A product distribution can then be formed between these points and the empirical marginal in x_{-u}. The measure then needs to be divided only by the marginal on x_{-u}, treating this collection of variables as a single variate as in (7.6).

Specific models of distribution of the underlying training data can point to alternative, more efficient, estimation schemes. Mixture of Gaussian models, or more general mixtures of products – for example, the CERT models of Chapter 5 – can be used to generate a product δ-measure for each term in the mixture and these can then be employed to provide identifiability at both lower computational cost and avoiding using rows that are given zero weight; in fact the weights given to the least squares equations would all be equal.

A similar scheme can be employed using a clustering of the training data: assuming that the underlying distribution makes up a product measure in each cluster. The points produced using this assumption would need to be appropriately weighted to maintain the prior weight in each cluster. A marriage of these two proposals might be found using the tree-based methods in Chapter 5. Here, the density model takes the form of a mixture of (possibly overlapping) uniform distributions. The training data in each component of this distribution can then be turned into a product measure, with the same weight given to each point in a single component, adjusted to maintain the over-all weight in that component. If we use overlapping regions, points should be chosen to be shared between the grids in each region: estimating on an individual region will bias the result and without coupling the grids, this will result in an offset in the effect estimate for each grid. The advantage of this approach is that it allows the distribution of training data within a component to modify the distribution toward the original δ-measure (in a Kullback-Leibler sense) while at the same time maintaining confidence in the function values at the points of measurement. Any point formed by taking a product δ-measure within one of the hyper-rectangles defining a uniform component must also be in that hyper-rectangle.

7.3 Confidence Intervals

Two distinct estimates of uncertainty are desirable here. Firstly, and most simply, there should be an estimate of $\mathrm{Var}(F(x)|x_u)$; the conditional variance of the function at that point, providing the amount of functional variation not captured by the effect f_u. Secondly, there should be some estimate of the variability of the effect estimate due to sampling.

The decomposition used here is chosen to reflect the specific training data and therefore it is hoped that sources of model variability due to resampling of errors at the given data points is small. Large effect variability under resampling of the errors would indicate significant model instability, reducing the confidence that would be placed in interpretational diagnostics. The same is also true of a straight bootstrap - resampling the training data entirely. This is much more likely to be a source of model instability with the same consequences for interpretation. This form of resampling by itself, then, is sufficient. The basics of doing such resampling are simple and have been dealt with, for example in [28], and are not covered further.

7.3.1 Conditional Variances

The first quantity of variance that is of interest

$$\mathrm{Var}(F(x)|x_u).$$

This can be used to provide a confidence interval to be displayed with the effect $f_u(x_u)$. This is an indication of how much functional behavior is not being accounted for by that effect. The estimate for this can be calculated directly by the weighted empirical variance:

$$\sigma_F^2(x_u) = \frac{\sum_{i=1}^{N_{-u}} w(x_u, x_{-u,i}) \left(F(x_u, x_{-u,i}) - \frac{\sum_{j=1}^{N_{-u}} w(x_u, x_{-u,i}) f_u(x_u)}{\sum_{j=1}^{N_{-u}} w(x_u, x_{-u,i})} \right)^2}{\sum_{j=1}^{N_{-u}} w(x_u, x_{-u,i})} \tag{7.9}$$

for $x_{-u,i}$ sampled uniformly on x_{-u} space - for example, from the product grid defined in (7.7).

Note that an asymmetric interval can be created by taking an ordering of $F(x_u, x_{i,-u})$, and using this to create an empirical lower and upper 5% bounds based on the weight $\int w(x_u, x_{i,u}) dx_u$ at each of these points. This is a more appropriate quantity to provide in this context. However, the estimate of $\sigma_F^2(x_u)$ is used in the estimates below and provides a single value for plotting contours of bivariate conditional variance.

Unlike conditional variances calculated for product measures in the standard Functional ANOVA, having an additive function does not result in a constant conditional variance. However, the variances defined here more accurately reflect the true functional variation with some particular subset of predictors instantiated.

7.3.2 Sampling Variation

The variability in the Monte Carlo approximation can be analyzed in a similar fashion to the standard Functional ANOVA. Observe that the estimate of an effect can be written

$$\hat{f}_u(x_u) = \sum_{i=1}^{N_{\{1,\ldots,d\}}} \alpha_i F(z_i)$$

where z_i are values on a grid generated by from a uniform sample. Note that $x_{i,u} = x_u$ for some of the points. Treating each $F(z_i)$ as independent, $\mathrm{Var}(F(z_i)) = \mathrm{Var}(F(z)|x_u) = \sigma_F^2(x_u)$ if $x_{i,u} = x_u$ and σ_F^2 otherwise. Each $\sigma_F(z_u)$ can be calculated from (7.9) and the resulting estimate is

$$\mathrm{Var}(\hat{f}_u(z_u)) = \sum_{i=1}^{N_{\{1,\ldots,d\}}} \alpha_i^2 \mathrm{Var}(F(z_i)),$$

which can be calculated by replacing the result vector in our linear equations with a vector of variances.

It would not be computationally feasible to solve a large set of equations for each point at which sampling variation is of interest. However, when we are interested in some subset $\{x_{j,u}\}_{j\in M}$ – for example, the points at which the effect is being plotted – these could be treated as jointly fixed and the appropriate conditional variance could be used any place that $x_{i,u} = x_{j,u}$ for some $j \in M$.

Although it produces an estimate, this is clearly a very naive approach. In general, $F(x_{1,u}, x_{1,-u})$, $F(x_{2,u}, x_{2,-u})$ and $F(x_{1,u}, x_{2,-u})$ cannot be said to be all independent. This dependence also an issue in variance estimates for the standard Functional ANOVA evaluated on a product grid. Producing a better description for the covariance structure of function values evaluated on a product grid generated from a uniform random sample is the object of ongoing research.

7.4 Demonstrations

Here I demonstrate the viability of the decomposition that I have proposed. In particular, I start out by demonstrating that we can recover additive components while ignoring spurious effects in a region of extrapolation. This is true even if the additive components share variables.

The example that we present is defined on the unit cube. I take as a function

$$F(x, y, z) = xy + xz + yz + 5I(x > 1/2, y > 1/2, z > 1/2) \tag{7.10}$$

and in this case the last term is considered to be a spurious effect of the learning procedure; I have chosen an indicator function to have particularly visible contours.

To evaluate the Generalized Functional ANOVA, a sample of ten uniformly distributed points were drawn, and the product distribution of the marginals of these points taken as in (7.7). Figure 7.1 presents contour plots of the second order effects for this function defined on three different measures. The first of these is exactly the unit cube providing the standard Functional ANOVA effects and the distortion due to the spurious term is evident. The second sets $w(x) = 0$ in the upper corner from the cube - exactly that part of the cube in which the spurious effect occurs. Here the desired components are recovered exactly. The final distribution further removes all the upper corners from each of the faces of the cube, leaving an "L"-shaped marginal distribution in each pair of variables as was found in Figure 4.1. Here the bivariate effects are again recovered, apart from in the top corner where they are left appropriately undefined.

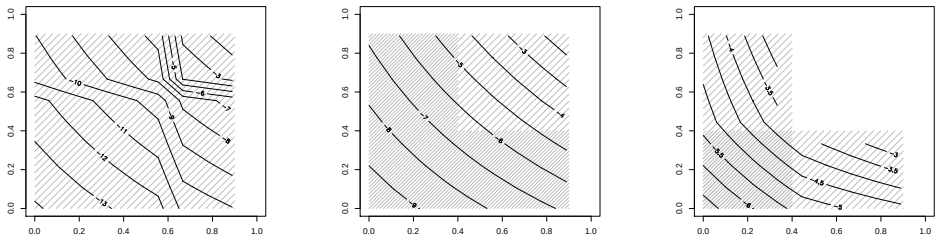

Figure 7.1: Bivariate effects for (7.10) defined on three successively more compact measures, the marginal distribution of each is given by background shading.

To provide a real-world demonstration of the effects, the Support Vector Machine used to provide the plots in Figure 2.4 has been employed again. Here, I have used the CERT model in Chapter 5 to provide an estimate of the density of the predictors for the Boston Housing Data. This has then been used as a density in the Generalized Functional ANOVA and I have taken a set of 10 uniformly distributed points on the dimension of interest in each case, and 20 randomly sampled training points for the

noise dimensions. The density has then been divided by the marginal distribution on the noise variables calculated from the CERT tree in Figure 5.3. Figure 7.2 compares the standard and generalized effects for this function on the variables "dis" and "lstat". We observe that the effects are similar to the partial dependence plots. However, "lstat" seems to be exaggerated in the partial dependence, while the dip in "dis" is stronger for the FAME representation.

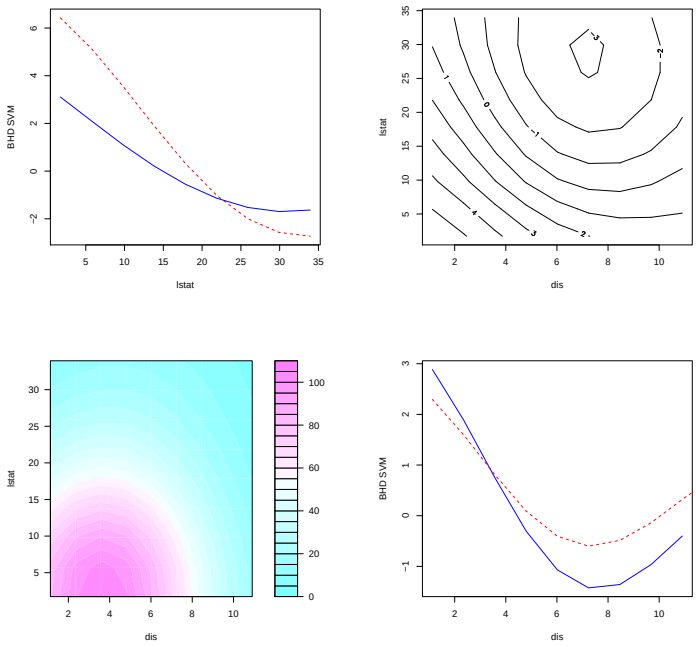

Figure 7.2: The effect of a regression Support Vector Machine trained on the Boston Housing Data for variables "dis" and "tax". Generalized effects are given by solid lines, standard effects by dashed.

Running VIN using the FAME criterion on a grid with 10000 points and taking an average of 10 repetitions to compensate for the smaller sample size reduced the set significant variables from 11 to 7: "nox," "rm," "age," "dis," "tax," "ptratio" and "lstat" with the only significant interaction being "rm"-"lstat". "rm"-"ptratio" and "rm"-"tax" were very close to the significance threshold while all other interactions

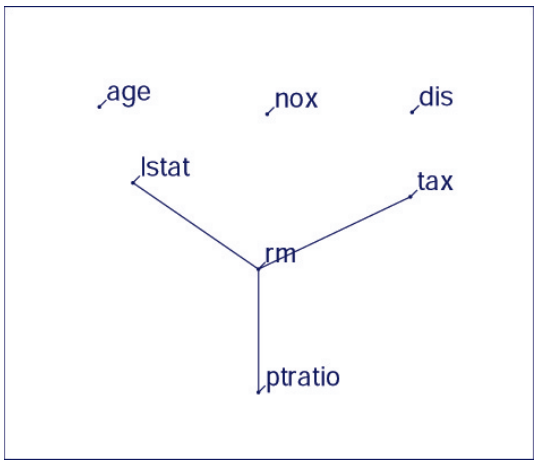

Figure 7.3: Variable Interaction Network for a Support Vector Machine learned on the Boston Housing Data using the FAME criterion.

had very low scores. Including these gives the VIN plot in Figure 7.3. A generalized additive model in these effects doubled test error, indicating that some expansion in model structure is warranted. Theorem 3.5.1 states that this algorithm only recovers the components *necessary* to fit the original function well and this does not guarantee sufficiency. There are likely to be a number of different ANOVA structures that obtain good prediction rates, all of which contain these components.

7.5 Conclusions

FAME represents a functional decomposition that satisfies the criteria I set out for good interpretational diagnostics. It provides a comprehensible quantity defined through an optimality criterion that guarantees that the function is fit as aggressively as possible without compromising additive structure. Further, I am able to do this in a way that avoids the need to evaluate functions at points of extrapolation and is also able to reduce the weight given to the function in regions of low probability mass.

This has been achieved through a generalization of a global optimality criterion and I have provided efficient techniques to solve it. Further, I have provided confidence intervals for the estimated quantities and ways to calculate these. Appendix C provides an account of an efficient numerical implementation of these procedures, particularly taking advantage of the sparsity of the systems involved. A greater understanding and estimation of the orthogonality structures within the decomposition would also be helpful, as would a deeper treatment of uncertainty.

Chapter 8

Conclusions and Conjectures

This thesis has sought to produce general methods for diagnostics in machine learning that are designed to be compatible with the output of any learner. I have focused on understanding system dynamics in terms of the relationship between the response and predictor variables. In this context, the Functional ANOVA represents a natural construction, providing effects that are comprehensible, optimal and which preserve additivity.

These three properties allow a diagnostic in the form of the "full grid of plots" that *jointly* provide a representation of functional behavior that as closely (in an $\mathcal{L}^2$ sense) fits the true function using only first order interactions.

Variable Interaction Networks

The Functional ANOVA effects produced by a full grid of plots, do not account for higher-order effects. Specifically they do not indicate which effects are important and how much functional behavior is captured (or missing) from bivariate plots. The work in Chapter 3 provides this diagnostic. The significance of an interaction can be judged by the cost of leaving it out of the Functional ANOVA. Thresholding significance at some predefined ϵ, the set of significant effects can then be represented

as a hypergraph, providing an addition to the full grid of plots from Chapter 2. These effects can be found hierarchically with an estimation scheme that takes $O(N)$ function evaluations.

The Functional ANOVA is normally defined on a uniform distribution, although it can easily be extended to any product measure. Chapter 3 goes somewhat farther and uses the calculation of partial dependence functions to mitigate the distortion of the underlying predictor distribution. Doing this, however, removes the base measure from the Functional ANOVA and in what sense, if any, partial dependence functions are jointly optimal as descriptors of non-additive functional dynamics is not clear.

Empirically, the ANOVA structure of a learned prediction function can vary noticeably when its learning parameters or training data is perturbed. Figure 8.1 shows the VIN plots for two 13-26-1 networks trained to convergence on the Boston Housing Data from different starting parameters. There are substantial differences in structure. This is partially due to the distortion caused by the independence assumptions used in the calculation of partial dependence functions. Even though they avoid using a full product measure, these calculations can move substantial probability mass into regions of extrapolation. Chapter 5 demonstrated that learned functions can exhibit large variability at points of extrapolation. If this is the case, using the methods described in Chapter 7 may provide some stabilization of these diagnostics. Figure 8.2 shows the same plots calculated with the FAME criterion. These are more similar and more sparse than Figure 8.1. The extent of variability is difficult to quantify – I am not aware of any measure of the variability learned structure in hierarchical models.

The VIN algorithm that I advocate finds those interactions that are necessary to recover the original function up to an $\mathcal{L}^2$ error of ϵ. These are not necessarily sufficient, however; employing FAME as a criterion to produce Figure 7.3 found an ANOVA structure that was too restrictive to model the data well. An upper bound

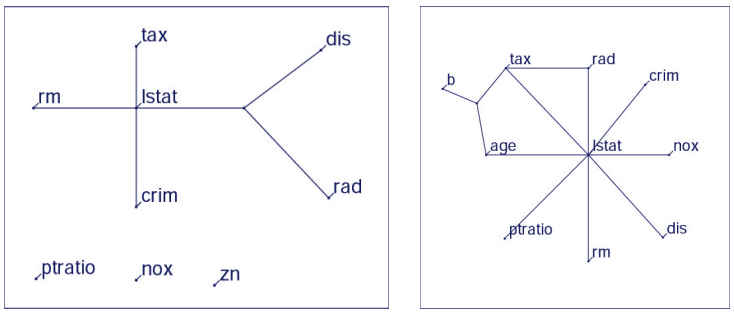

Figure 8.1: VIN plots for two 13-26-1 neural networks trained to convergence on the Boston Housing Data.

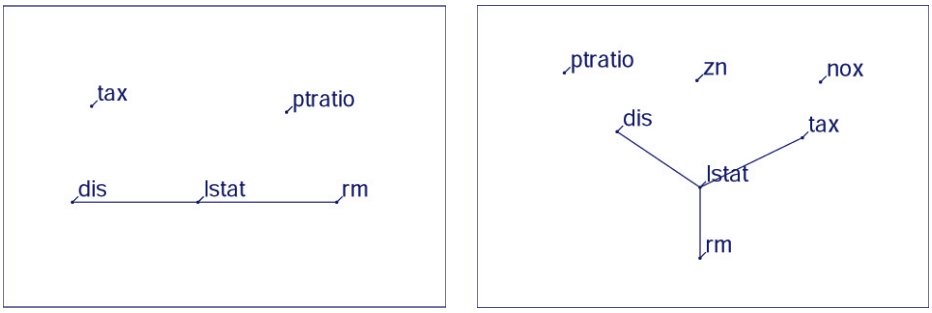

Figure 8.2: VIN plots for the same neural networks as Figure 8.1 but calculated with the FAME criterion.

on the set of minimal ANOVA structures that recover F to within ϵ[1] could be found by recursively searching over all structures and adding a structure as soon as its joint $\mathcal{L}^2$CoE drops below ϵ. This would involve much more computational effort, however, and is unlikely to produce an interpretable result.

Alternative Diagnostics

There are diagnostic questions that the Functional ANOVA does not address. Individual predictions are not explained by low-order effects. The explanatory power

[1]By minimal, I mean $\mathbf{U}$ that recovers F to within ϵ, but that $\mathbf{V}$ doesn't for any $\mathbf{V} \prec \mathbf{U}$.

that rules have in trees is difficult to mimic for general methods. Rules can be thought of as a specific example of an algebraic formulation of a function. Few other formulations rarely have the same interpretive value as rules, however. One approach to rule generation is to build a tree on a very large simulated data set with the corresponding predictions as a response. This suffers the disadvantages of both producing many rules and losing accuracy.

The usefulness of the Functional ANOVA is also limited when there are known underlying processes generating both predictors and response. Such structures are found in functional data, for example. This typically occurs when data are collected on a process evolving over time and the predictor variables are measurements of various aspects of the system. In this setting, functional effects for these measurements entirely miss the temporal nature of the data and machine learning carried out naively with these predictors is likely to perform poorly. In this situation, the rate of change of variables can have important explanatory power and such systems are naturally described by differential equations, an approach begun in [26].

There are other settings in which the effect of a predictor is not a quantity of interest. This is the case, for example, in image processing in which the change associated with a particular pixel is not particularly interpretable. In other very high dimensional situations, examining the effect for individual predictors is likely to swamp our cognitive abilities. These settings often rely on a more comprehensible set of lower dimensional bases; principal components are often used in Natural Language Processing, for example. When this is the case, effects for these bases can provide interpretable diagnostics.

Confidence and Extrapolation Representation Trees

The central theme of this work has been the problem of extrapolation. The comments above about the variability of ANOVA structure mirror the general problems

associated with extrapolation described in Chapter 4. There I showed that extrapolation can distort the Functional ANOVA picture of functional dynamics. In realistic settings, this distortion can be very bad.

CERT provides a first tool to deal with extrapolation: discovering where it occurs. Extrapolation is generally not formally defined. I give it as the classification probability of the data distribution against a uniform distribution on the same range. The uniform distribution makes a natural hypothesis distribution for outliers, as well as a null distribution when we think about the number of points in a given number of dimensions. This classification probability then needs to be estimated, and CERT represents one of the only situations in which I advocate a specific methodology. Trees are natural in this context for two particular reasons:

1. Trees provide a rough representation of the data distribution as a mixture of uniform distributions. This can easily be refined to estimating a product distribution in each leaf, since each product distribution will not spread mass outside its leaf. I suggest ways in which this may be useful in the estimation of effects in Chapter 7.

2. The tree glyph represents an interpretable estimate of density. Moreover, it provides a diagnostic tool for functional dynamics: we can identify areas of low data density and determine if functional behavior in these areas makes predicted values suspicious.

Beyond the representation, both graphical and probabilistic, that they provide, trees can also naturally incorporate exact distributional information about the uniform distribution. It is less obvious how to do this for other classifiers. Incorporating this information into the algorithm leads to larger, more accurate and more stable trees.

As with CART, the use of axis-oriented splits can be limiting when the true density has linear association structure. This limitation can be seen in the experiments in

§5.4. I have not implemented the linear combination split strategy available in CART, but expect that it would substantially increase performance on data sets that exhibit linear dependence. Calculating the volume of hyper-polygonal regions is a non-trivial task here. However, a Monte Carlo strategy used to calculate the score for each split may be used here. Since uniform data will be drawn separately for each split, the problems of separability associated with the Monte Carlo strategy for ordinary CERT should not arise.

I have shown that CERT provides a viable extrapolation detection procedure and suggested ways in which this could be turned into a diagnostic of concept drift in predictor variables. Any outlier detection procedure could be used here. What CERT provides above a generic procedure is an indication of the direction of drift: if a record of the amount of data falling in each leaf of the CERT tree is kept, the trends on each leaf provide an indication of how the distribution of predictor variables shifts over time. This may be best demonstrated in practice.

While I have advocated the use of trees as an extrapolation diagnostic, the measure of extrapolation could be estimated with other methods. Using the strategy of generating Monte Carlo points from a uniform distribution, any classifier could be employed as an estimator. Experimental results suggest that this strategy can yield results with insufficient resolution in trees and this is likely to be the case for other methods. One possible work-around is to use an ensemble of classifiers, each learned with a different Monte Carlo data set. This could be bagged. CERT itself could be bagged or boosted. [29] provides a method for boosting density estimation. This is an alternative strategy to boosting classifiers that might also work for CERT.

Data-Augmented Regression for Extrapolation

When learning is undertaken with a flexible approximator there is no reason to trust the predictions given far away from data. Machine learning, of necessity, makes

an assumption of some form of smoothness – depending on the specific method – in the true target function. This smoothness allows prediction to extrapolate to points nearby the training examples. However, as points get farther away from training data, assumptions of smoothness still allow a very large variability. Making predictions at such points implicitly invokes a Bayesian prior that the true function behaves according to the peculiarities of the learning procedure used. Almost no learners have model assumptions or search strategies that are designed with extrapolation as anything more than an afterthought. They are therefore unlikely to imply a sensible prior distribution on prediction values.

The flexible nature of machine learning procedures does, however, allow their prior to be altered stochastically. I suggest that it is sensible to view a target function as being drawn from a random field prior with a known, stable, null model as a mean. Unless strong prior knowledge is available, an appropriate null model is constant. A universal approximator can then be influenced to use this prior model by generating data that is uniformly distributed across predictor space, giving it the response from the null model and appending it to the training data. I have suggested some general rules of thumb concerning how much uniform data is necessary and how to choose the relative weight of uniform and real data. These are computationally costly, however; the rule of thumb for the amount of uniform data required is likely to be frequently infeasible. More realistic base estimates for these parameters would significantly speed up the search process. These may need to be tailored to the particular learning method being used.

The use of DARE does not need to be restricted to universal approximators. In the case of linear regression, DARE corresponds to a stochastic version of ridge regression and could be used in more general GLMs. In fact, the DARE paradigm provides a new analysis of regularization methods in parametric situations; it is reasonable to ask what prior on predicted values is induced by the priors on the parameters. In many situations, we have more intuition about what predictions should look like than

where parameters should lie. In the case of logistic regression with a ridge penalty, for example, the predicted values implicitly have an inverse logistic Gaussian prior.

In the naive implementation of DARE, the variance of the random field prior is assumed to be the same as that of the error distribution on the training examples. This may be reasonable in many situations. It may also be unreasonable - for example, when we expect the observation errors to be over-dispersed. In this case a different loss criterion should be used for the errors measured on the real data points and those on the uniform points: Huberized on the former and Gaussian on the latter, for example. Doing this would necessitate a significant alteration of the learning procedure. In a gradient boosting situation, described in [10], however, the implementation of separate loss criteria for different observations only requires the gradient calculation for each observation to be changed.

Another tantalizing suggestion that I have not to followed up is the use of exact distributional information in trees in the same manner as CERT. So long as a constant base model is used (or one that is analytically integrable on hypercubes), it is again possible to provide an exact loss calculation for the prior distribution for any loss function. This can then be added with the appropriate weight to all the split scores. Like the use of specific points in ridge regression this would remove the stochastic component of DARE. Such trees can, of course, then be used in any ensemble method for trees.

A final area of interest is in creating confidence intervals for functions. The behavior of DARE resembles Kriging, which is explicitly based on a Gaussian random field prior and can similarly incorporate a null model. Besides the regularization and extrapolation benefits already described, Kriging provides confidence intervals for its predictions. For Kriging, these intervals revert to intervals based on the prior variance away from training data. In DARE, the weight on Monte Carlo data points corresponds to an implied prior variance. An initial approach to doing this would be to use the simplistic approach in §6.4. Let the implied prior variance be $1/\gamma$ – exactly the

variance implied for a Gaussian prior – and suppose a variance for measurement error to be estimated at σ^2 (presumably smaller than $1/\gamma$). Then an over-all predictive variance could be given by

$$P(D|x)\sigma^2 + \frac{1 - P(D|x)}{\gamma}.$$

This does not take into account the variability of the model fitting procedure, however. In Kriging, the variogram plays a critical role in governing the trade-off above and the measure proposed here is quite naive.

The Functional ANOVA for Measures of Extrapolation

My interest in the problem of extrapolation began with a concern over the distortion of diagnostic tools caused by that phenomenon. FAME, the final construction in this thesis, provides a generalization of the Functional ANOVA to non-product measures. In doing this it provides the same optimality and additivity properties of the standard Functional ANOVA without requiring evaluation at points of extrapolation. As with multivariate linear regression, the effects generated from FAME have a standard interpretation: they are the best fit of the prediction function, *when the other effects are taken into account.* All the functional diagnostics described in Chapters 2 and 3 can be used with FAME instead of the Functional ANOVA and in doing this the problems of extrapolation can be avoided.

FAME does present significant estimation challenges. I have proposed initial methods for meeting these based on evaluation on a grid of points. This rapidly becomes numerically difficult, although it is feasible for basic diagnostics. There are a number of sub-issues that could be further explored: the use of multiple products and a mixture-of-products representation to better target areas of large density, smoothing the effects of interest by using a finite element basis for that effect with point masses for the others. The interaction between finite element methods and the distribution

point masses also needs to be fleshed out. There are further issues in providing estimates of uncertainty and variance for FAME. These stem mostly from the (unknown) correlation of function values evaluated on a randomly generated grid. Whether reasonable estimates of these correlations can be found and used is a matter of on-going work.

Classification and Transformation

This thesis has concerned itself almost exclusively with the problem of regression – predicting a real-valued quantity. As noted in §2.4, there are standard techniques for turning regression effect estimates into classification estimates. These are often done on a logistic scale instead of a probability scale. This work has also not considered the issue of function transformations. $f(x)$ may not be close to additive, but some transformed version, $G(f(x))$, may be. It may also be that we can write

$$f(x) = h_1(g_1(x)) + \ldots + h_k(g_k(x)).$$

Projection pursuit regression [11] is a specific case of this with the g_i being linear in x. Either transformation may prove useful and the linear direction of greatest variation of a function is another diagnostic aid. In general, however, searching for such transformations is likely to over-complicate the diagnostic. A number of basic transformations might be useful - taking a log of the prediction will find multiplicative effects and multiplicative structures. Nonetheless, such transforms should be chosen *a priori* and with some care - the result of transforming a predictor effect by a function that is not additive in the set of effects shown is not a mental task that is easily carried out.

Final Thoughts

This work presents the first exposition of the connection between extrapolation and diagnostic tools of which I am aware. I have strengthened current diagnostic tools in producing a representation of interaction importance. I have provided an interpretable density estimate and used it to evaluate function behavior at points of extrapolation. I have also discussed principled methods for making prediction at points of extrapolation. Finally, I have provided a framework in which diagnostic tools can be provided without requiring the function to be evaluated at points of extrapolation.

I have carried out this work with the intention that it be generally applicable to any black box function, whatever the underlying model formulation. There are numerous machine learning methods in existence, and undoubtedly more will be produced. All the diagnostic tools that I have presented here can be employed to aid the understanding any function and are not restricted solely to the out put of a learning procedure. In CERT and FAME, I have suggested estimation schemes. In both cases, these are derived from a general quantities that are themselves original to this thesis and may be susceptible to alternative estimation techniques.

Good diagnostic tools not only provide scientific understanding of the system in question, they also allow us to create better models that rely on stronger assumptions to improve both the accuracy and the stability of predictions. The first and most obvious use of diagnostics is in feature selection. Reducing the dimensionality of the predictor space simplifies models and removes a source of variance. It is also likely to substantially reduce the amount of extrapolation in a system. There are a host of predictive procedures already developed for machine learning, all of which are improved by removing irrelevant predictors. I suspect that a large source of predictive improvement will be in methods that have structural model assumptions and in knowing which assumptions to use.

Appendix A

Abuses of Notation

Because of the complicated nature of some of the general calculations relating to the functional ANOVA, some notational shortcuts have been employed.

General Analysis

$\mathbb{R}^d$ d-dimensional Euclidean space.

$\mathcal{L}^2(\mathbb{R}^d)$ The set of square-integrable functions (with uniform measure) on $\mathbb{R}^d$. $\mathcal{L}^2$ will be used when the space is clear.

$\mathcal{L}^2_w$ The set of functions that are square-integrable with respect to the measure implied by w.

Extrapolation Probabilities

$P(D|x)$ Represents the probability that x was generated from distribution D when the choice is D or U with assumed prior γ. I.e.

$$\frac{\gamma P(x|D)}{\gamma P(x|D) + (1-\gamma)P(x|U)}$$

In the case of $\gamma = 1/2$, this is exactly Extrap(x).

The Functional ANOVA:

u, v Subsets of the indeces $1, \ldots, d$.

u_i The ith element of u, under some ordering.

$-u$ The complement of u.

$u \backslash v$ The elements of u not in v.

x_u Those variables whose indeces are in u.

x_i The ith example of the predictor variables in a set of examples.

Where u is a singleton giving the kth index, I write $x_{\{k\}}$.

$x_{i,u}$ The values of x_u in the ith example. The index of the example always precedes that of the variable.

$\mathbf{U}$ A collection of $u \subset \{1, \ldots, d\}$. By convention, for $u, v \in \mathbf{U}$, $u \not\subseteq v$ and $v \not\subseteq u$.

$\prec$ $\mathbf{V} \prec \mathbf{U}$, states that for all $v \in \mathbf{V}$ there is some $u \in \mathbf{U}$ for which $v \subseteq u$ and for some $v \in \mathbf{V}$ there is a $u \in \mathbf{U}$ such that $v \subset u$. $v \prec \mathbf{U}$ is equivalent to $\mathbf{V} = \{v\} \prec \mathbf{U}$.

Grids

I assume that in each coordinate k, the grid takes N_k distinct values $\{x_{i,k}\}_{i=1}^{N_k}$.

$z_{i,u}$ i is regarded as a multi-index of the size k - the cardinality of u:

$$z_{i,u} = (x_{i_1,u_1}, \ldots, x_{i_k,u_k})$$

N_u Indicates $\prod_{k \in u} N_k$.

$\{z_{i,u}\}_{i=1}^{N_u}$ Is taken to enumerate over all N_u values of the multi-index i associated with u.

Appendix B

Existence and Uniqueness of FAME Solutions

In order to be precise in the discussion of solutions to FAME, we must note that no solution will be well defined away from the support of the density $w(x)$. I therefore assume that any function discussed is a representative of an equivalence class defined by the values that the function takes on the support of w. For definiteness, I assume that all functions are zero outside that support.

B.1 Existence

Theorem B.1.1. *For any $f \in \mathcal{L}^2_w$, a FAME decomposition exists.*

Proof. The set of functions

$$\mathcal{G} = \left\{ g : g = \sum_{v \subset \{1,\ldots,d\}} g_v(x_v) \right\}$$

where $\{g_v\}_{v \subset \{1,\ldots,d\}}$ satisfies the conditions (7.2) is a closed subset of $\mathcal{L}^2_w$. This is clear: for any element $g \in \mathcal{G}$ take an ϵ ball centered on g in the $\mathcal{L}^2_w$ metric. Then $g + \epsilon t$

is within this ball when t is the independent multivariate Gaussian centered on the origin and truncated by by the support of w.

Since $f \in \mathcal{L}^2_w$, we have, by the triangle inequality

$$||g||^2_w \leq ||f||^2_w + ||f - g||^2_w$$

and the minimizer must be at least as good an approximate to f as 0, so we can bound the norm of a minimizer by 0 and $2||f||^2_w$. Call G' the restriction of G to the $||f||^2_w$ ball about zero.

Let $J_f[g]$ be the criterion (7.1) applied to g. By another triangle inequality, $J_f[g]$ is continuous in g and bounded by $||f||^2_w$ and 0.

Since J_f is continuous and G' is closed and bounded, the image of G' under J_f is closed and bounded and is therefore a compact set of $\mathbb{R}$. A minimum therefore exists in this set and its inverse is in G'. □

B.2 Uniqueness

The less obvious result is the uniqueness of the FAME solution. The conditions (7.2) are designed to ensure that the solutions are unique. The challenge here is to find regularity conditions on w under which this is successful. The idea of grid closure below directly motivates the approximation scheme in §7.2.

Definition B.2.1. *A set Ω is* grid closed *if for any $x \in \Omega$, there exists $\{y^u \neq x\}_{u \subset \{1,\ldots,d\}} \subset \Omega$ such that $y^u_u = x_u,\ u \subset \{1, \ldots, d\}$.*

Definition B.2.2. *A function w is said to be* grid closed *if its support, Ω_w, is grid closed.*

I have used the name "grid closed" to suggest the existence of a grid in Ω for every point $x \in \Omega$. In fact, a full grid is not necessary and it is possible to define fractal

sets that are grid closed but which do not contain any grid. Nonetheless, for practical purposes we will use a full grid. The next lemma is obvious and given without proof.

Lemma B.2.1. *Any grid or union of grids is grid closed.*

The point of grid closure is that it ensures that functions of one variable must be constant with respect to other variables. This property provides the next lemma.

Lemma B.2.2. *Let w be grid closed. For any $\{g_u\}_{u\subset\{1,\dots,d\}} \neq 0 \in \mathcal{L}^2_w$ that satisfy the integral constraints (7.2), $\{g_u\}_{u\subset\{1,\dots,d\}}$ are linearly independent under the inner product defined by w.*

Proof. Set $g_u = \sum_{v\neq u} \beta_v g_v$. By assumption, the β_v are not all zero. By the orthogonality (7.5), $\beta_v = 0$ for $v \supset u$.

Now $g_u = \sum_{v\not\supset u} \beta_v g_v$. Consider any point $x = (x_u, x_{-u}) \in \Omega_w$. By grid closure, there is some other point $y = (x_u, y_{-u}) \in \Omega_w$. Now for any such $y \in \Omega_w$

$$g_u(x) = \sum_{v\not\supset u} \beta_v g_v(x_u, x_{-u}) = \sum_{v\not\supset u} \beta_v g_v(x_u, y_{-u}) = g_u(y)$$

and therefore for any given z, $g_u(x_u) = \sum_{v\not\supset u} \beta_v g_v(x_u, z)$ can be written as

$$\sum_{v\not\supset u} \beta_v g_v(x_u, z) = \sum_{v\subset u} f_v(x_v). \tag{B.1}$$

for some functions $\{f_v\}_{v\subset u}$ Now, however, observe that (7.5) implies

$$\int g_u(x_u) f_v(x_v) w(x) dx = 0$$

for any f_v by the condition (7.2). g_u is therefore orthogonal to $\sum_{v\subset u} f_v(x_v)$ and we must have $g_u = 0$. Contradiction. □

Grid closure is necessary here to ensure (B.1) holds. To demonstrate it's necessity, consider a bivariate w with support only on the line $y = x$. On the support of w,

$f(x) = f(y)$ for any f and we cannot insist on being able to write (B.1). This linear independence will now be used with a calculus of multiple variations to derive the main result.

Theorem B.2.1. *For w grid closed and $f \in \mathcal{L}^2_w$, the FAME criterion for f has a unique minimizer.*

Proof. Call the FAME for criterion for f evaluated at G $J_f[G]$. Let $F = \{f_u\}_{u\subset\{1,\ldots,d\}}$ be a minimizer of J_f and let $G = \{g_u\}_{u\subset\{1,\ldots,d\}} \neq 0$ satisfy (7.2). Let $H = \{h_u\}_{u\subset\{1,\ldots,d\}} = \{f_u + \epsilon_u g_u\}_{u\subset\{1,\ldots,d\}}$ for $\epsilon = \{\epsilon_u\}_{u\subset\{1,\ldots,d\}}$. Then $J[H]$ has a minimum at $\epsilon = 0$.

Continuing the calculations from Theorem 7.1.1, the Hessian of $J[H]$ with respect to ϵ is given by the inner product matrix of G under w:

$$\left[\nabla^2_\epsilon J[H]\right]_{u,v} = \int g_u(x) g_v(x) w(x) dx$$

By the Lemma B.2.2, the elements of G are linearly independent under w, and this matrix is positive definite. $J[H]$ is therefore convex in ϵ and $\epsilon = 0$ is the unique solution.
Since this is true for any $G \neq 0$ satisfying (7.2), F is the unique minimizer of J. □

I used grid closure as a motivation for the approximation scheme in §7.2. Below I present stronger and more common regularity conditions.

Lemma B.2.3. *An open set of $\mathbb{R}^d$ is grid closed.*

Proof. Let x be in the support of w. Since the support of w is open, $\exists \epsilon > 0$ such that $B_\epsilon(x)$ is also in the support of w. Now take the grid of values defined by adding $\epsilon/2n$ to any set of coordinates of x. This grid is wholly contained in $B_\epsilon(x)$. □

Theorem B.2.1 and Lemma B.2.3 immediately provide the following corollary, giving a more common form of regularity.

Corollary B.2.1. *For any w that has support on an open set of $\mathbb{R}^d$, and for any $F \in \mathcal{L}^2_w$, there exists a unique FAME decomposition for F.*

In a the same vein, we can demonstrate the conditions for (7.8) to be well defined.

Corollary B.2.2. *If the support of w is a union of grids, then for any F in $\mathcal{L}^2_w$ there exists a unique FAME decomposition.*

Appendix C

Sparse Matrix Solutions for FAME

This appendix provides a description of the computational complexity associated with solving (7.8) using conjugate gradient methods. The structure of the equations allows the linear system to be solved with much lower computational and memory cost than a naive implementation would imply. I will abuse notation further here and assume we are estimating an effect f_u. $-u$ will be taken as a single variable which will become the $|u|+1$st element of u.

C.1 FAME Estimation in Matrix Form

Using matrix notation defined below, I will re-write (7.8) as

$$\sum_{i=1}^{N_u} w(z_i)\left(X_{i,}f - F(x_i)\right)^2 = (Xf-F)^T W(Xf-F)$$

under the constraints, $\forall v \subset u, \forall x_{k,v}$:

$$\sum_{i=1}^{N_{v\setminus\{j\}}} w_u(x_{i,v\setminus\{j\}}, x_{k,j}) f_v(x_{i,v\setminus\{j\}}, x_{k,j}) = 0 \Leftrightarrow Cf = 0$$

$X_{i,}$ indicates the ith row of X.

Here the following notation is used, with examples for estimating a univariate effect $f_{\{1\}}$:

f a vector indexed by $\{i, u\}, i \in 1, \ldots, N_u$ giving $f_{i,u} = f_u(z_i)$.

For estimating $f_{\{1\}}$ this is (in transpose):

$$[f_1(z_{1,1}), \ldots, f_1(z_{N_1,1}), f_{-1}(z_{1,-1}), \ldots, f_{-1}(z_{N_{-1},-1})]$$

F, W are vectors giving $F(z_i)$ and $w(z_i)$ respectively for $i \in 1, \ldots, N_u$.

X A matrix indexed by $i \in 1, \ldots, N_u$ rows and $\{j, v\}$, $v \subset u$,$j \in 1, \ldots, N_v$. It's entries are given by

$$[X]_{i,\{j,v\}} = I(z_{j,v} = z_{i,v})$$

and for the univariate problem it can be written as

$$\begin{bmatrix}
1 & 0 & \cdots & 0 & 1 & 0 & \cdots & 0 \\
0 & 1 & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
\vdots & \ddots & \ddots & 0 & \vdots & \vdots & \ddots & \vdots \\
0 & \cdots & 0 & 1 & 1 & 0 & \cdots & 0 \\
 & \vdots & \vdots & & & \vdots & \vdots & \\
1 & 0 & \cdots & 0 & 0 & \cdots & 0 & 1 \\
0 & 1 & \ddots & \vdots & \vdots & \ddots & \vdots & 1 \\
\vdots & \ddots & \ddots & 0 & \vdots & \ddots & \vdots & \vdots \\
0 & \cdots & 0 & 1 & 0 & \cdots & 0 & 1
\end{bmatrix}$$

C Is indexed by $\{k, \{v, j\}\}$, $v \subset u$, $j \in v$ $k \in 1, \ldots, N_{v \setminus j}$and $\{i, v'\}$, $v' \subset u$, $i \in 1, \ldots, N_{v'}$ with entries given by

$$[C]_{\{i,v_j\},\{k,v'\}} = I(v = v')I(z_{i,v\setminus j} = z_{k,v\setminus j})w_v(z_{k,v\setminus j})$$

which translates, for the example, to

$$\begin{bmatrix} w_1(z_{1,1}) & \cdots & \cdots & w_1(z_{1,N_1}) & 0 & \cdots & \cdots & 0 \\ 0 & \cdots & \cdots & 0 & w_{-1}(z_{1,-1}) & \cdots & \cdots & w_{-1}(z_{N_{-1},-1}) \end{bmatrix}$$

Note that both X and C are very sparse. I show in §C.2 how these objects can be coded efficiently and that X never needs to be stored.

The desired result f, then solves

$$\begin{bmatrix} X^TWX & C^T \\ C & 0 \end{bmatrix} \begin{bmatrix} f \\ \lambda \end{bmatrix} = \begin{bmatrix} X^TWF \\ 0 \end{bmatrix}$$

where λ is a Lagrange multiplier vector indexed by $\{k, \{v, j\}\}$, $v \subset u$, $j \in v$, $k \in 1, \ldots, N_{v\setminus j}$. This system is $\sum_{v\subset u}\left(N_v + \sum_{j\in v} N_{v\setminus j}\right)$ square. Moreover, the $\sum_{v\subset u} N_v$-square matrix X^TWX is no longer sparse and requires $N_u^2 \sum_{v\subset u} N_v$ calculations to create.

For the purposes of the demonstrations in this thesis, I have solved this system naively on problems using up to 100 points to generate grids. This will not be feasible for larger numbers of points or more than three dimensions. However, good iterative methods can be applied to solve this system at much lower cost.

C.2 Iterative Methods and Storage

By Corollary B.2.2, so long as w has sufficient regularity on the grid, the linear system is invertible. Using an iterative method like the conjugate gradient method

[23] only requires left multiplication of a vector by the matrix on the left hand side. With reasonable weights w, we can expect such methods to converge quickly.

As observed, X^TWX is not sparse and even this multiplication is cumbersome. However, left multiplication can be performed in steps. To make this efficient, I will re-represent each of the quantities in the manner suggested by their indexing:

F, W can be placed into $|u|$-dimensional arrays, with entries exactly corresponding to the grid points where they are evaluated. Requires storing N_u real numbers.

f is a list of $2^{|u|}$ arrays indexed by $v \subset u$. Requires $\sum_{v \subset u} N_v$ reals.

C We make use of sparsity to store C as in f: a list of sub-arrays, one for each $v \subset u$, containing the marginal of w on v calculated on the grid points. This can be created by calculating X^TW. We need $\sum_{v \subset u} N_v$ reals.

λ Is a list of lists. For $v \subset u$, store $|v|$, $|v|-1$-dimensional arrays, the jth of which has the elements in $\lambda_{v \setminus j}$. Stores $\sum_{v \subset u} \sum_{j \in v} N_{v \setminus j}$ reals.

I will not store or calculate X. The required left multiplication can then be achieved using the following operations:

- $X : f \to F$: The resulting component at multi-index i is given by $\sum_{u \subset v} f_{i_v, v}$. Requires $2^{|u|} N_u$ additions.
- $W : F \to F$: N_u component-wise multiplications.
- $X^T : F \to f$: At v and multi-index i_v, this is given by $\sum_{j_{-v}} F_{\{i_v, j_{-v}\}}$ - a summation over the remaining dimensions. Total requirements are $2^{|u|} N_u$ additions.
- $C^T : \lambda \to f$: At $f_{i,v}$ calculate $C_{i,v} \sum_{j \in v} \lambda_{j,i}$. Requires $2^{|u|-1} \sum_{j \in u} N_j$ additions.
- $f + C^T\lambda$: $\sum_{u \subset v} N_v$ component-wise additions.

- $C : f \to \lambda$: At component $\{i, v_j\}$ calculate $\sum_{k=1}^{N_i} f_{\{i_j, k_{v\setminus j}\}, v} C_{\{i_j, k_{v\setminus j}\}, v}$. Takes $2^{|u|-1} \sum_{j \in u} N_j$ additions and multiplications.

So the total storage cost and total computational cost of this iteration are $O(N_u)$. Given a bounded set of iterations, the total approximation cost is on the order of the size of the grid. In general terms, $N_u \approx N^{|u|}$ which quickly becomes large with the dimension of $|u|$. However, since we can visualize and most second-order interactions, this is sufficient for practical purposes.

For interaction importance scores, we have already commented that the conditions C are not necessary, nor are all the effects. This leaves an under-determined system. However, techniques for iterative solutions to these also exist [5]. The main cost involved in such a solution remain left multiplication by X and X^T which have computational cost $|u|N_u$ for this setting.

Bibliography

[1] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proceedings of the 20th International Conference on Very Large Databases*, 1994.

[2] L. Breiman. Bagging predictors. *Mach. Learn.*, 24(2):123–140, 1996.

[3] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth, 1984.

[4] A. Buja, D. F. Swayne, M. L. Littman, N. Dean, and H. Hofmann. Xgvis: Interactive data visualization with multidimensional scaling, 2001. http://www.research.att.com/areas/stat/xgobi/index.html.

[5] D. Calvetti, L. Reichel, and Q. Zhang. Iterative solution methods for large linear discrete ill-posed problems, 1998.

[6] J.-P. Chilés and P. Delfiner. *Geostatistics: Modeling Spatial Uncertainty*. Wiley Series in Probability and Statistics, New York, 1999.

[7] S. E. Feinberg. *The Analysis of Cross-Classified Categorical Data*. MIT Press, 1980.

[8] Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *International Conference on Machine Learning*, pages 148–156, 1996.

[9] J. H. Friedman. Multivariate adaptive regression splines. *Annals of Statistics*, 19(1):1 – 141, 1991.

[10] J. H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29(5):1189–1232, 2001.

[11] J. H. Friedman and W. Stuetzle. Projection pursuit regression. *J. Amer. Statist. Assoc.*, 76:817, 1981.

[12] D. Harrison and D. L. Rubinfeld. Hedonic prices and the demand for clean air. *Journal of Environmental Economics and Management*, 5:81–102, 1978.

[13] T. Hastie and R. Tibshirani. *Generalized Additive Models*. Chapman and Hall, London, 1990.

[14] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Machine Learning: Data Mining, Inference and Prediction*. Springer, New York, 2001.

[15] M. Hegland. Adaptive sparse grids. In *Proceedings of the 10th Biennial Computational Techniques and Applications Conference*, 2002.

[16] W. Hoeffding. A class of statistics with asymptotically normal distributions. *Annals of Mathematical Statistics*, 19:293–325, 1948.

[17] T. Jiang and A. B. Owen. Quasi-regression with shrinkage. *Math. Comput. Simul.*, 62(3-6):231–241, 2003.

[18] O. Linton and J. P. Nielsen. A kernel method of estimating structured nonparametric regression based on marginal integration. *Biometrika*, 82(1):93–100, 1995.

[19] R. Liu and A. B. Owen. Estimating mean dimensionality. Technical report, Stanford University, 2003.

[20] P. McCullagh and J. A. Nelder. *Generalized Linear Models*. Chapman and Hall, London, 1989.

[21] P. Niyogi, F. Girosi, and T. Poggio. Incorporating prior information in machine learning by creating virtual examples. *Proceedings of the IEEE.*, 86(11):2196–2209, 1998.

[22] A. B. Owen. The dimension distribution and quadrature test functions. *Statistica Sinica*, 13(1), 2003.

[23] W. H. Press, S. A. Teukolski, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 1997.

[24] R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaurmann, San Mateo, 1993.

[25] R-project. http://www.r-project.org/.

[26] J. O. Ramsay. Differential equation models for statistical functions. *Canadian Journal of Statistics*, 28:225–240, 2000.

[27] J. O. Ramsay and B. W. Silverman. *Functional Data Analysis*. Springer, New York, 1997.

[28] C. Roosen. *Visualization and Exploration of High-Dimensional Functions Using the Functional Anova Decomposition*. PhD thesis, Stanford University, 1995.

[29] S. Rosset and E. Segal. Boosting density estimation. In S. T. S. Becker and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 641–648. MIT Press, Cambridge, MA, 2003.

[30] I. M. Sobol. Global sensitivity indices for nonlinear mathematical models and their Monte Carlo estimates. *Math. and Computs in Sim.*, 5:271–280, 2001.

[31] G. Wahba. Spline models for observational data. *SIAM CBMS-NSF Regional Conference Series in Applied Mathematics*, 59, 1990.

[32] C. K. I. Williams and C. E. Rasmussen. Gaussian processes for regression. In D. Touretzky, M. Mozer, and M. Hasselmo, editors, *Advances in Neural Information Processing Systems 8*, Cambridge, MA, 1996. MIT Press.