Statistical Modeling of High-Dimensional Nonlinear Systems: A Projection Pursuit Solution

A Thesis
Presented to
The Academic Faculty

by

Michael D. Swinson

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

School of Mechanical Engineering
Georgia Institute of Technology
December 2005

Statistical Modeling of High-Dimensional Nonlinear Systems: A Projection Pursuit Solution

Approved by:

Professor Nader Sadegh, Advisor
School of Mechanical Engineering
Georgia Institute of Technology

Professor I. Charles Ume
School of Mechanical Engineering
Georgia Institute of Technology

Professor Steven Liang
School of Mechanical Engineering
Georgia Institute of Technology

Professor Brani Vidakovic
Industrial and Systems Engineering
Georgia Institute of Technology

Professor Alexander Shapiro
Industrial and Systems Engineering
Georgia Institute of Technology

Date Approved: November 22, 2005

For my parents;

for Grandpa, Robert H. Burton (1912-2004)

and for my former committee member,

Dr. Robert E. Fulton (1931-2004),

both of whom were unable to see this work completed;

and most importantly, for you, the reader,

who might be reviewing this many years from now -

I hope, in this work, there is at least the smallest

morsel of information that might prove useful

to you in your research.

ACKNOWLEDGEMENTS

I would like to take this opportunity to thank the many people who helped to make this thesis possible. My deepest gratitude goes to my advisor, Dr. Nader Sadegh. I deeply appreciate his constant inspiration, guidance, encouragement, and support. I could not have asked for a better mentor.

I would also like to thank my committee members, Dr. Steven Liang, Dr. Alexander Shapiro, Dr. I. Charles Ume, and Dr. Brani Vidakovic for all of their feedback and comments to help make this work complete. I am also grateful to the late Dr. Robert Fulton for his valuable advice and input during the process of this research. He has certainly left his imprint on this work.

I am deeply grateful to Dr. Wepfer for all of his advice, support, and encouragement over the years. Without him, this work would not have been possible. I must also acknowledge Dr. Jerry Ginsberg, Dr. Xiaoming Huo, Dr. Anthony Hayter, Dr. Liang Peng, and Dr. Ye-Hwa Chen for simply teaching their classes and presenting material to me in an invaluable way. Certainly all my teachers I over the years have contributed to getting me where I am today. Two of the most influential were my chemistry teacher, the late John E. Yanaitis, who taught me that good research and a good sense of humor go hand in hand, and my calculus teacher, Harry Kutch, who rekindled my love for mathematics and instilled in me the belief that no problem, big or small, can withstand the onlsaught of a creative mind, persistently focused. To my first boss, Rick, who reminded me to always look at things from multiple perspectives.

I would also like to express my sincere appreciation to many friends and colleagues: Yong, Rogelio, JingYing, Li Qiang, David, Susan, Justin, Kris, Lynnane, Mike R., Debao, and Patrick for making the process fun. I would like to thank Scott Billington for the use of the data from his bearing defect experiment and Dr. Joel Lander for providing intraday stock data.

To the friends who have been with me the whole time, Jing, Joel, Sharon, Shawn J., Chung, Michael Y., Matt A., Tom, Murli, Jimmy, Missy, Isaac, Bruce, Ben, Evangelos, Johnny P., and to all of those too numerous to name here, I thank you all for being a part of my journey and supporting me along the way, and for reminding me that time is measured in moments, not years.

Of course, I must also thank my family, who has been with me and stood by me my entire life. My Mom and Dad have always given me the support to make my dreams realities. My Mom, especially, has encouraged me all along this lengthy process. My brother also gets special mention for his support and for always being someone I could look up to while we were growing up.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Despite recent advances in statistics, artificial neural network theory, and machine learning, nonlinear function estimation in high-dimensional space remains a nontrivial problem. As the response surface becomes more complicated and the dimensions of the input data increase, the dreaded "curse of dimensionality" takes hold, rendering the best of function approximation methods ineffective. This thesis takes a novel approach to solving the high-dimensional function estimation problem. In this work, we propose and fully develop two distinct parametric projection pursuit learning networks with wide-ranging applicability. Included in this work is a discussion of the choice of basis functions to be used in these networks as well as a description of the optimization schemes utilized to find the parameters that enable each network to best approximate a response surface. The essence of these new modeling methodologies is to approximate functions via the superposition of a series of piecewise one-dimensional models that are fit to specific directions, called projection directions. The first of these algorithms is designed to be implemented on functions with a potential unbounded domain for the projection directions. The second is designed to be used on functions consisting of projections with limited coupling, which is often the case in most real-world applications. The key to the effectiveness of each model lies in its ability to find efficient projections for reducing the dimensionality of the input space to best fit an underlying response surface. Moreover, each method is capable of effectively selecting appropriate projections from the input data in the presence of relatively high levels of noise. This is accomplished by rigorously examining the theoretical conditions for approximating each solution space and taking full

advantage of the principles of optimization to construct a pair of algorithms, each capable of effectively modeling high-dimensional nonlinear response surfaces to a higher degree of accuracy than previously possible.

CHAPTER I

INTRODUCTION AND LITERATURE REVIEW

1.1 Problem Statement

Despite much research over the years, high-dimensional nonlinear function approximation remains a difficult problem. To be effective, an algorithm must be able to discern the relevant inputs, maintain a feasible computational complexity in the face of a large volume of data, and yet still approximate the response function accurately. A major problem facing the researcher in making predictions on high-dimensional data spaces is the aptly named "curse of dimensionality," which describes the effects on function approximation of the sparsity of sample data points in a high-dimensional function domain. As a result, a finite set of observations might not be enough to adequately describe the original response mapping. Projection methods attempt to circumvent this problem by first transforming the input space into a series of low-dimensional projections onto which the predictive model is painted. The major difficulty with such methods is how to choose the most appropriate projections: the ones that best describe the unknown response surface.

The major focus of this work is on how to best choose the optimal projection directions. In this thesis, we will extend the initial formulations of the fundamental projection pursuit regression algorithm [33] (which sequentially identifies projection directions, approximating the functions along these directions with flexible smoothing techniques), by applying to it a different iterative method for finding the optimal projections. In our approach, we define the problem parametrically to optimize projection directions simultaneously and numerically solve the function approximation problem. We will provide the foundational theory behind just such a methodology,

including a proof of its universal approximation capabilities. We also propose a second method for solving for the projections. This method first discretizes the potential projection space and then utilizes a random search technique governed by a criterion for minimal error to select the final directions to use in the model. The theory behind this approach is also presented. In the literature, similar methods are termed projection pursuit learning networks (PPLN). There are a number of papers that suggest such strategies, although their approaches differ in a few fundamental ways, including utilizing a nonparametric modeling scheme for fitting one-dimensional models along projection directions and using gradient-search techniques for solving for the optimal directions. Further, such approaches borrow from the heavily-researched neural network theory. Some of this theory, such as the proof of universal approximation, are more restrictive than is necessary for projection pursuit networks. We extend this theory and show how it can be applied parametrically with any basis functions satisfying a rather general set of criteria.

1.2 Modeling Methodologies

The goal of regression analysis is to estimate the conditional expectation of a response on the basis of the values of relevant inputs. To this end, two different general classes of modeling techniques have been employed for this estimation problem: parametric and nonparametric techniques. Multivariate adaptive regression splines and artificial neural networks are two such examples of high-dimensional parametric modeling techniques, both of which will be discussed in more depth later in the thesis. In a parametric modeling approach, the functional form of the regression surface is often assumed. The model can be quite accurate provided that the initial assumptions are correct. However, in practice, one often does not have sufficient information to assume the form of the response surface. As a consequence, nonparametric techniques are often used. These methods, which make only a few general assumptions about

the regression surface, are applicable to systems in which only information regarding the inputs and outputs are known. However, nonparametric approaches have their drawbacks as well.

Typical nonparametric methods, including splines, kernel approaches, and nearest neighbor techniques, rely on full-dimensional local averaging around the prediction point. These approaches utilize a weighted average of responses for observations with predictors in the same neighborhood as the point of interest. Prior work [76] has shown that these techniques have some favorable asymptotic properties. However, they do not perform well in high-dimensional settings, even in the presence of reasonable sample sizes. The reason is that such techniques suffer from what Bellman termed the curse of dimensionality [7]. This refers to the phenomenon in which the sample size necessary for estimating a function within a certain degree of tolerance grows exponentially with the total number of input variables.

Localized regression techniques are one of the most common form of nonparametric modeling methods. These methods each involve the use smoothing around the local region of interest. In fact, this concept is not new, as Schiaparelli, and Italian meteorologist, began investigating this method in 1866 [72], followed by De Forest (1873) [23]. Because of the effectiveness and intuitive nature of this method, it has found wide-ranging applicability and acceptance: [18], [74], and [53] in the field of economics, [55] in numerical analysis, [91] in sociology, [71] and [88] in chemometrics, [62] in computer graphics, and [2] in machine learning.

The local regression method has largely been developed on theoretical results of parametric regression methods and is based on finite sample theory of linear estimation. This theory, developed in sources such as [49], [14], and [86], trivialized problems that have proven to be major stumbling blocks for the more widely studied kernel methods.

Another theoretical treatment of the local regression approach is to view the

method as an extension of kernel methods by attempting to extend kernel theory to local regression. This treatment has become popular recently, for example in works by [87] and [27]. But, for practical purposes, some [57] have claimed that kernel theory is of limited use, basing their evidence on its often poor approximations and highly restrictive required conditions.

Despite the effectiveness of the local regression technique for a wide range of low-dimensional problems, it has yet to be practically extended to effectively model high-dimensional systems. The approach tends to fail with more than 2 or 3 dimensions, even with moderate sample sizes. This ineffectiveness directly results from the curse of dimensionality.

1.3 Data Sparsity

While we have derived several lemmas to prove the existence of the curse of dimensionality and certain specific attributes of high-dimensional space, in the interest of not getting bogged down in details, they will resurface in Chapter 2. Instead, an illustration of the effects of this phenomenon can be presented in the form of an example. If, for instance, one sets the dimensions of the local neighborhoods to cover 10% of the nearest points for each coordinate axes, we find that in 10-dimensional space only 0.1^{10}, or 0.00000001% of the total sample will be included on average. Thus, each local region will likely be empty. To counteract this, one could assign the total number of points for the local neighborhood. For instance, if the number of nearest points to be included is chosen to be 1% of the total sample, then 63% of the nearest points along each coordinate axes will have to be included. This destroys the accuracy of the model. Thus, the issue of high dimensional sparsity limits the effectiveness of traditional nonparametric modeling techniques. Hence, a new approach is needed which will be able to better handle such sparsity in high dimensions and still provide sufficient accuracy in estimating the underlying regression surface.

1.3.1 Intrinsic Dimensionality

Given a system with D independent variables, it will in practice, appear to have D_2 (where $D_2 > D$) degrees of freedom due to the presence of noise, measurement error, etc. Yet, provided the influence of these factors is not so overwhelming as to completely mask the original structure of the system, one should be able to filter such noise out and recover the original variables. We now define the intrinsic dimension of the system as the number of inputs that satisfactorily explain the system. The intrinsic dimension would be the dimension M of the projected variables that govern the operation of the system. Across varied domains like vision, speech, motor control, climatology, genetic distributions, human motor control, and a range of other physical and biological sciences, various researchers corroborated that the true intrinsic dimensionality of high dimensional data is often very low [80], [67], [85], [26]. We interpret these findings as evidence that the physical world has a significant amount of coherent structure that presents itself as being well-suited to dimension reduction techniques. For instance, in the realm of computer vision, it is quite obvious that the neighboring pixels of an image of a natural scene possess redundant information. Moreover, the probability distribution of natural scenes, in general, has been found to be highly structured. Thus, this illustrates an example application that lends itself to a sparse encoding in terms of set of basis functions [65].

So, dimension reduction techniques can be quite useful in a wide range of applications. Thus, the determination of the intrinsic dimension of a sample distribution is important in many prediction applications, as it is central to the problem of dimension reduction. Knowing the intrinsic dimension would eliminate the possibility of overfitting or underfitting the data. Of course, the problem is itself ill-posed, because given a data sample it is possible to make a manifold of any dimension pass through it (assuming no observations have duplicate inputs) with negligible error given enough parameters. Thus, it is important to take these things into account when formulating

a dimension reduction algorithm.

1.3.2 Unsupervised Dimension Reduction

One way to avoid the curse of dimensionality is to reduce the number of input dimensions. A common unsupervised learning technique used to accomplish this objective is principal component analysis (PCA). Principal component analysis (or the Karhunen-Loeve transform, as its known in signal processing) is the most commonly used dimension-reduction technique used in practice, primarily because of its simplicity and computational efficiency.

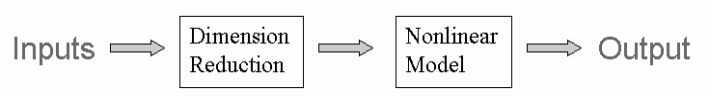

Figure 1: Model With Unsupervised Dimension Reduction

Thus, one method employed by some has been to first reduce the dimensionality of the problem with a technique like principal component analysis and then apply a nonlinear modeling technique on the reduced subspace of inputs [77], [78], and [22]. Techniques such as partial least squares (PLS) [89], [29] and principal component regression (PCR) [60], [82] use the superposition of univariate regressions onto principal component projections of the input space to make response predictions.

1.3.3 Principal Component Analysis

The basic concept behind principal component analysis is to transform the inputs into a new set of input vectors that are uncorrelated in an attempt to create the maximum separability in the input space. The dimensionality of this space may also be reduced by retaining only those components which contribute a specified proportion of the total variation in the data.

This method makes the assumption that the distribution of the data takes the form of a hyperellipsoid, such that the vector of means and the covariance matrix

define the shape and dimensionality of the distribution [73]. Let us consider a sample in $\Re^D$ with mean $\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$, covariance matrix $\Sigma = E\left[(x-\overline{x})(x-\overline{x})^T\right]$, and spectral decomposition $\Sigma = U\Lambda U^T$, with $U = [u_1 \ u_2 \ \ ... \ \ u_D]$ orthogonal and Λ diagonal. The principal component transformation: $y = U^T(x-\overline{x})$ produces a new reference with respect to the sample has zero mean and a diagonalized covariance matrix Λ containing the eigenvalues of Σ.

$$\Lambda = \begin{bmatrix} \lambda_1 & 0 & 0 & 0 \\ 0 & \lambda_2 & 0 & 0 \\ 0 & 0 & ... & 0 \\ 0 & 0 & 0 & \lambda_D \end{bmatrix}$$

Thus, the u_j's are the eigenvectors corresponding to the eigenvalues, λ_j,with the eigenvalues ordered as such: $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_D > 0$. The question then becomes: how many principal components should be to reduce the dimensionality while still capturing information of the original system? To this end, a typical rule employed is to choose $M < D$ such that:

$$\frac{\sum_{i=1}^{M}\lambda_i}{\sum_{i=1}^{D}\lambda_i} > \rho$$

where ρ is some arbitrarily selected percentage of the sum of the system's total eigenvalues.

Using this methodology of projecting the initial raw variables onto the subspace spanned by the first M principal components, the transformed variables are uncorrelated, and those variables with a small variance are discarded. An example of such a projection is illustrated [12].

The problem with this type of approach, as diagrammed in figure (2), is that the dimension reduction is performed with an unsupervised technique. Indeed, it is quite evident that the selection of the optimal dimensional subspaces will be dependent on the response surface.

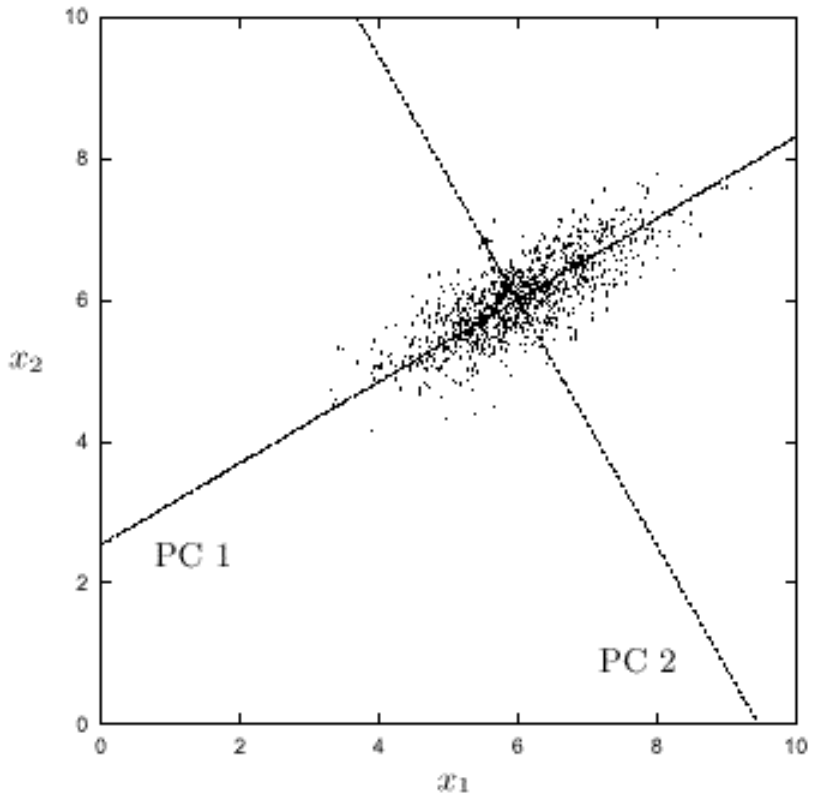

Figure 2: Two-Dimensional Normal Point Cloud with its Principal Components

1.4 Potential Solutions - supervised learning approaches

Over the recent years, some novel ways of trying to address the problem of choosing the best projections have been devised. Although they utilize an unsupervised local dimensionality reduction algorithm, Teh and Roweis [79] constrain their projections to describe a single, coherent low-dimensional coordinate system by enforcing agreement amongst principal components to fit a specific coordinate transformation. Some recent work, such as that of Vijayakumar [82] focus on finding efficient local projections to approximate functions in the neighborhood of a given query point. In one of his papers [83], Vijayakumar introduces a locally weighted projection regression (LWPR) algorithm, which uses locally linear models spanned by a series of one-dimensional regressions along selected input space directions.

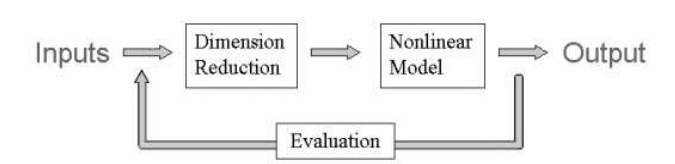

Figure 3: Model With Supervised Dimension Reduction

One method that has been proposed is covariant projection regression (CPR) [84]. This is a method that sequentially chooses its projection directions by taking into account not only the input distribution, as in the case of principal component regression, but also the covariance of the input and output data. Thus, CPR extends the technique by supervising the learning of optimal projection directions, as illustrated in figure (3). However, because this and the other algorithms mentioned above fit linear functions along individual projection directions, they are not particularly adept at capturing nonlinearities in a high-dimensional function space.

A number of methods have been developed in an attempt to address this problem. Payman Sadegh and Henrik Ojelund [70] introduce the concept of hierarchical local

regression to incorporate both local approximations in the neighborhood of a given query point and global information in the form of a set of weights from a global regression function. The concept of the multilayer nodal link perceptron network, or NLPN, is introduced in [69] and [28]. This type of model employs multi-dimensional local bases to capture nonlinearities in the response function. Among the best known of these algorithms that are designed to target nonlinearities in high-dimensional space are multivariate adaptive regression splines [37], neural network models, and projection pursuit regression [33]. The first two of these will be discussed in this chapter, while projection pursuit, central to the research of this thesis, will be discussed in much greater depth in the chapter that follows.

1.4.1 MARS

To fit a response surface, multivariate adaptive regression splines [31], or MARS models, adaptively build a set of basis functions in the original coordinate system. MARS adds basis functions by a forward selection procedure. For each input, x_i, and every possible value, t of x_i, MARS splits the data into two parts at the "knot", t. MARS then keeps the knot and the associated variable pair that provides the best fit. On these two parts, each comprising the data on one side of the knot, a pair of linear functions is fit. Each of these functions is non-zero on one side of the knot only. After one variable has been selected, further splits can be assigned via forward selection based on the previous split (splitting the input space on one side of the previous knot only), or it ignoring the previous split and splitting the entire input space on the new knot. MARS adds to the set of basis functions using a penalized residual sum of squares.

Generally, the forward selection procedure will overfit on this initial pass. Thus, MARS prunes these results, via generalized cross validation criterion, using backward elimination on the selected set of basis functions. Finally, the MARS procedure

replaces the linear basis functions with cubic splines to smooth out the approximation.

MARS is able to handle nonlinearities quite well. However, unlike projection methods, MARS operates in the original coordinate system only. De Veaux and Unger [22], provide an extension of this approach by combining the MARS procedure with the linear projection principal component method to achieve more accurate results in the face of input space multicollinearity.

1.4.2 Artificial Neural Networks

Neural networks are universal approximators. Given enough nodes, an artificial neural network can represent any well-behaved function. In general, neural nets are relatively robust to outliers and are capable of fitting highly nonlinear data quite well. When constructing models of nonlinear high-dimensional systems, neural networks are often chosen because of their of their wide-ranging applicability to such input spacings. Besides being relatively robust to outliers and noisy data, such models are suitable for efficient implementation on massively parallel computers as their hidden units only pass information to and from those units sharing a direct connection. However, interpretability of neural network model results is difficult due to the high degree of interaction and multicollinearity between the variables and basis functions. Thus, neural networks are best used as "black box" models, where interpretability of the governing model is not as important as identifying an accurate input-output relationship. Perhaps the most common form of neural networks is the single hidden-layer feedforward network with sigmoidal activation functions. This type of network uses one-dimensional activation functions to project high-dimensional spaces onto several single-dimensional spaces that are nonlinearly activated and then summed together. Such a network can be written in the form

$$y_i = \sum_{j=1}^{N} w_{ij} \sigma_j (x)$$

where

$$\sigma_j(x) = \left[1 + \exp\left(-\sum_{k=1}^{n} w_{jk} x_k\right)\right]^{-1}$$

and the weights w_{ij} and w_{jk} are selected by a nonlinear optimization method to minimize the loss function over the training set. Typically, Levenberg-Marquardt optimization is used. The Levenberg-Marquardt algorithm is a cross between gradient descent and Newton's method:

$$w_{k+1} = w_k - \left(J^T J - \mu I\right)^{-1} Je$$

where J is the Jacobian of the error criterion, μ is the gradient descent weighting, and e is the error between the target and the prediction. When the scalar μ is zero, is essentially Newton's method using the approximate Hessian matrix. When μ is large, the method converges to gradient descent with a small step size.

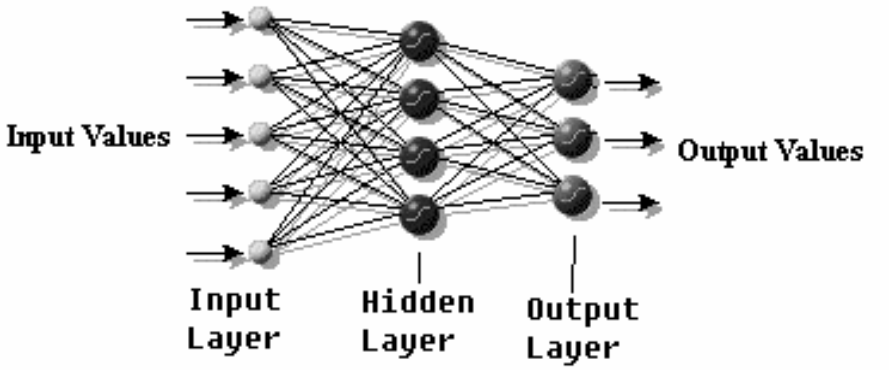

Figure 4: Schematic of the Architecture of a Single Hidden Layer Neural Network

Since neural networks are so widely used when attempting to model high- dimensional multivariate functions and have such incredible modeling flexibility, why is there a need for any other type of model? In fact, it is the large flexibility provided by neural network models that leads to some problems. First, for any given training set and any given model architecture, i.e. the number of hidden units, the weight matrix is not uniquely determined. This means that ANN models are not

identifiable. Second, the optimization problem is nonconvex and quite often unstable. While feedforward neural networks have the ability to approximate high-dimensional functions with relatively few activation functions, they use global basis functions and are difficult to train. In fact, the training of neural networks has become quite an art, of sorts. The gradient descent optimization rule, which is often used for finding the estimates, may get stuck at local minima. Thus, based on the random sequence in which the inputs are presented to the network and based on the initial values of the input parameters different solutions may be found. In fact, the non-identifiability of neural network solutions which are caused by the possible non-uniqueness of a global minima and the existence of possibly many local minima leads to a large prediction variance. Thus, the large flexibility provided by neural network models leads to predictions with a relatively small bias, but also leads to a large variance [44]. Careful methods for variance control [3], [9], [10], [66], [43] are thus required to robustify the prediction. Third, there is the problem of optimal network architecture selection (number of hidden layers, number of hidden units, weight constraints, etc.). This problem can be addressed to some degree by cross validatory choice of architecture [9], [10], or by averaging the predictors of several network with different architecture [90].

With so many problems with traditional neural networks, there is thus a need for a method that will handle some of these problems. The novel methods introduced in this work intend to address some of these problems with the use of a simpler architecture that involves fewer average parameters and a more efficient optimization scheme that will hopefully help alleviate some of the problems of optimal network architecture selection, reduce the possibility of getting caught in local minima, and achieve better overall estimation results.

1.5 Thesis Organization

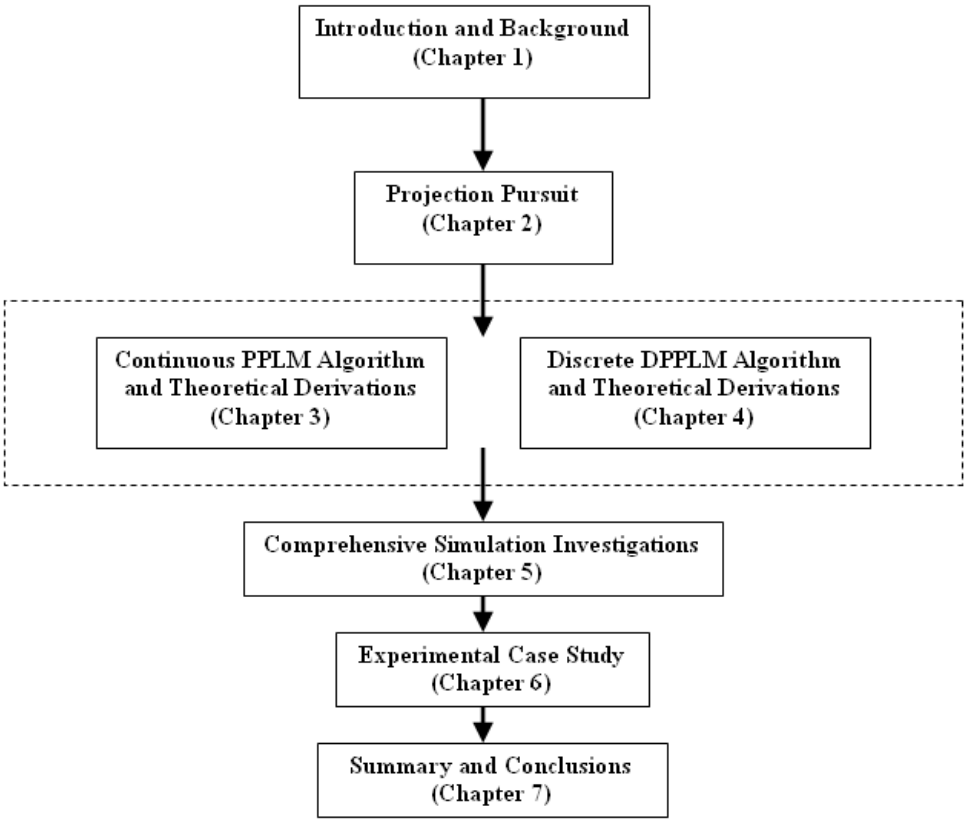

Figure 5: Flowchart of Paper Structure

In the next chapter, we focus on projection pursuit approaches. We will start by examining the "curse of dimensionality" mathematically to gain a better understanding of why a dimension reduction approach like projection pursuit might be important. Then, we will explore the state of the art in projection pursuit learning up to this point.

Chapter 3 introduces the newly-proposed parametric projection pursuit learning model of this thesis. This method solves the function approximation problem by optimizing the projection directions simultaneously using a Levenberg-Marquardt optimization technique. In this chapter, we explore the mathematical framework of the method. The major theoretical contributions of this chapter include: 1.) a theorem to

show that high-dimensional functions satisfying a rather broad set of criteria can be decomposed into an infinite number of single dimensional, mutually orthogonal functions, 2.) the derivation of the projection pursuit data dimension reduction technique from the context of Fourier Analysis, and 3.) a universal approximation theorem for the newly-proposed method. We then explore the effectiveness of the introduced technique by looking at its predictive performance on experimental results.

In Chapter 4, we introduce yet another projection-pursuit-inspired model, the discretized parametric projection pursuit learning model. This prediction methodology uses a different technique for finding optimal projections: the use of a random search on a discretized set of projection couplings. Again, we introduce the theoretical underpinnings of the approach using a similar mathematical framework. However, we will find that the theory differs a bit, as our underlying assumptions in this case are different. Next, we propose two different techniques for selecting these discretized projections, and then we proceed to provide experimental results for the method, using one of the proposed techniques.

In Chapter 5, we provide a thorough comparison of the newly proposed methods with commonly-used high-dimensional prediction techniques. An extensive set of simulations is run to examine the varying effectiveness of each of the given methods under different conditions.

Chapter 6 consists of an experimental case study on financial data. In this case study, a comparison of DPPLM simulation results with those of feedforward neural networks are provided. Also presented is an extensive statistical investigation analyzing the effectiveness of the DPPLM approach on the simulation data.

In Chapter 7, we present our conclusions, a summary of the major contributions of this work, and suggestions for the future direction of the research.

CHAPTER II

IN PURSUIT OF OPTIMAL PROJECTIONS

One major problem facing the researcher in making predictions on high-dimensional data spaces is the effect of the sparsity of sample data points in a high-dimensional function domain. As a result, a finite set of observations might not be enough to adequately describe the original response mapping. By first transforming the input space into a series of single-dimensional projections, projection methods are able to get around this problem to some extent. The major difficulty with this, however, is: how does one choose the optimal projections? This is where the true challenge of high-dimensional function approximation lies. If one can solve this problem, then prediction methodologies can be constructed that are both accurate and robust even in the presence of high levels of noise and high degrees of data sparsity. In this chapter, we will first explore the problems of high-dimensional modeling in more depth, and then we will follow this with a detailed exploration of projection pursuit methods.

2.1 The Curse of Dimensionality

In Chapter 1, we used a simple example to illustrate the aptly-named curse of dimensionality. To step beyond this simplified illustration, let us now work through some proofs to glean a better understanding of this strange phenomenon. Let us begin with a lemma:

Lemma 1 *For a hyperellipsoid in d dimensions, its equation can be written as:*

$$\frac{X_1^2}{\lambda_1^2} + \frac{X_2^2}{\lambda_2^2} + ... + \frac{X_d^2}{\lambda_d^2} = 1$$

and its volume can be written as [51]:

$$V_d = \frac{2}{d}\left(\prod_{i=1}^{d}\lambda_i\right)\frac{\pi^{d/2}}{\Gamma\left(\frac{d}{2}\right)}$$

where $\Gamma(x)$ *is the gamma function given by* $\Gamma(x) = \int_0^\infty t^{x-1}e^{-t}dt$

Using this lemma, we can proceed to make some interesting conclusions about data sparsity and the "curse of dimensionality."

Lemma 2 *The volume of a hypersphere decreases toward zero with increasing dimensionality:*

Proof. *For a hypersphere, we note that* $\lambda_i = r$ *for* $i = 1, 2, ..., d$. *Thus, the volume of the hypersphere reduces to [5]:*

$$V_d = \frac{r^d\pi^{d/2}}{\Gamma\left(\frac{d}{2}+1\right)}$$

Provided that r *is bounded by an arbitrarily large, but finite value* M*; i.e.,* $r \in (0, M]$. *Then,*

$$\lim_{d\to\infty} V_d = \lim_{d\to\infty}\frac{r^d\pi^{d/2}}{\Gamma\left(\frac{d}{2}+1\right)} = 0$$

Thus, V_d *will vanish as* d *becomes large, since* d *can always be chosen such that* $\frac{d}{2} >> \pi M^2$. ■

These results are illustrated for a unit hypersphere in figure (6). The results of this lemma can be further extended to a hyperellipsoid of a general form.

Lemma 3 *The generalized volume of a hyperellipsoid decreases toward zero with increasing dimensionality:*

Proof. For a hyperellipsoid of general form:

$$\frac{X_1^2}{\lambda_1^2} + \frac{X_2^2}{\lambda_2^2} + ... + \frac{X_d^2}{\lambda_d^2} = 1$$

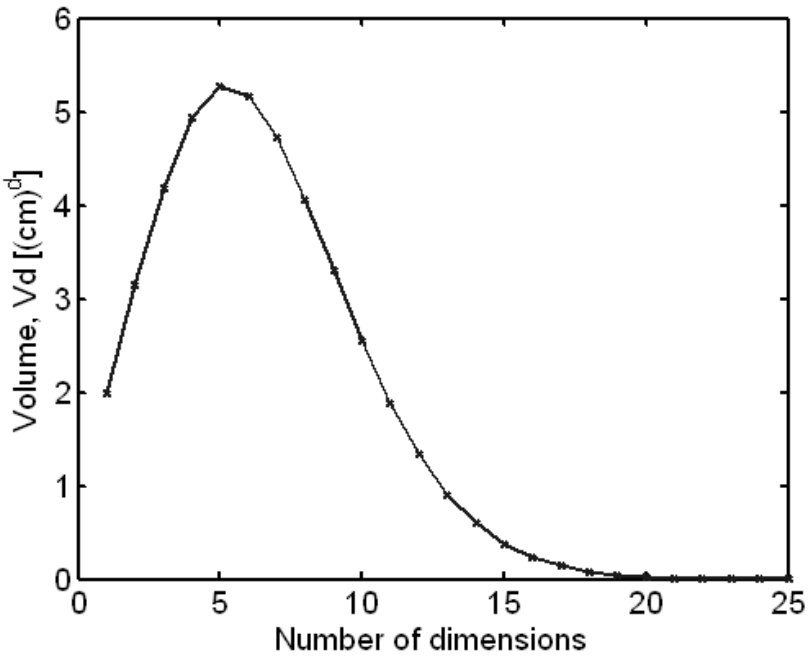

Figure 6: d-Dimensional Volume of a unit cm Hypersphere by Dimension

and provided that $\lambda_{\max}$ is bounded by an arbitrarily large, but finite value M; i.e., $\lambda_{\max} \in (0, M]$. Then,

$$0 \leq \lim_{d\to\infty} V_d = \lim_{d\to\infty} \frac{\pi^{d/2}}{\Gamma\left(\frac{d}{2}+1\right)} \prod_{i=1}^{d} \lambda_i \leq \lim_{d\to\infty} \lambda_{\max}^{d} \frac{\pi^{d/2}}{\Gamma\left(\frac{d}{2}+1\right)} = 0$$

where $i = 1, 2, ..., d$. Thus, V_d will vanish as d becomes large, since d can always be chosen such that $\frac{d}{2} >> \pi M^2$. ■

In fact, we can extend this analysis to gain further insight into the issue of data sparsity.

Lemma 4 *The volume of a general hyperellipsoid tends to concentrate in an outer shell with increasing dimensionality: [45]*

Proof. First, we note that for a hyperellipsoid of the form:

$$\frac{X_1^2}{(\lambda_1 - \varepsilon_1)^2} + \frac{X_2^2}{(\lambda_2 - \varepsilon_2)^2} + ... + \frac{X_d^2}{(\lambda_d - \varepsilon_d)^2} = 1$$

where $0 \leq \varepsilon_i < \lambda_i$ and $i = 1, 2, ..., d$, then its volume can be written as

$$V_d = \frac{\pi^{d/2}}{\Gamma\left(\frac{d}{2}+1\right)} \prod_{i=1}^{d} (\lambda_i - \varepsilon_i).$$

Now, to illustrate the lemma, we note that the volume ratio of two hyperellipsoids, one slightly smaller than the other is:

$$\frac{V_d(\lambda_i - \varepsilon_i)}{V_d(\lambda_i)} = \frac{\prod_{i=1}^{d} (\lambda_i - \varepsilon_i)}{\prod_{i=1}^{d} \lambda_i} = \prod_{i=1}^{d} (1 - \xi_i)$$

where $\xi_i = \frac{\varepsilon_i}{\lambda_i}$ for $i = 1, 2, ..., d$ and $0 < \xi_{\min} < 1$. Letting $\xi_{\min} = \min\left(\frac{\varepsilon_i}{\lambda_i}\right)$ for $i = 1, 2, ..., d$, we can see that

$$\lim \frac{V_d(\lambda_i - \varepsilon_i)}{V_d(\lambda_i)} \leq \lim \prod_{i=1}^{d} (1 - \xi_{\min}) = \lim (1 - \xi_{\min})^d = 0.$$

■

Remark 5 *Using the results of this lemma, it is evident that data in high-dimensional space is expressible in less than full dimensionality.*

To show how quickly data tends to concentrate in an outer shell for increasing dimensionality, we take the example of two hyperspheres with

$$\frac{V_d(r - \varepsilon)}{V_d(r)} = (1 - \xi)^d$$

where $\xi = \frac{\varepsilon}{\lambda} = 0.1$. Here, we find that the ratio of the two volumes tends to zero rather quickly as the number of dimensions increases.

Thus, the volume of a hypersphere in d-dimensions tends to concentrate in an outer shell as the number of dimensions grows. A general d-dimensional hyperellipsoid has similar properties.

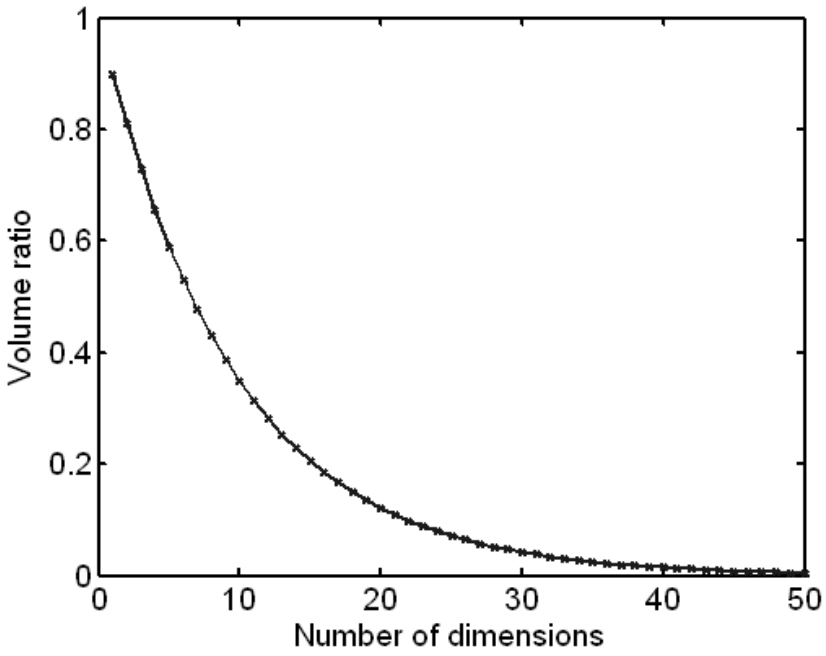

Figure 7: Percentage of d-Dimensional Hyperspherical Volume Not Concentrated in Outer Shell

Remark 6 *Note that as an extension of the prior lemma, one could show that the volume of a hypercube, or (more generally) the volume of any regular hyperparallelepiped, also tends to concentrate in an outer shell with increasing dimensionality: The proof follows similarly from the proof of the preceding lemma.*

Thus, it seems that if the volume of the n-dimensional space tends toward zero clustering in lower dimensional surfaces, then if we can find the structure of these lower dimensional clusters, we might be able to project the data onto such surface. This would help alleviate the curse of dimensionality.

2.2 Previous Work In Projection Pursuit

Many of the potential solutions to this data sparsity problem in high-dimensional space often fall under the category of projection pursuit. Projection pursuit [40] is a dimension reduction technique that identifies interesting low-dimensional linear

projections of a high-dimensional space by optimizing an objective function, called a projection index. Thus, any structure seen in a projection is but a shadow of the actual structure in original space. It is of interest to the researcher to pursue the sharpest projections: those that reveal the most information contained in the high-dimensional data distribution. With this method, the scaled components of the projection vectors that define the corresponding solution indicate the relative strength that each variable contributes to the observed effect. It is of interest to note that several methods of classical multivariate analysis are special cases of projection pursuit (for example, principal component analysis). Two disadvantages [12] of projection pursuit are: 1.) because it works with linear projections, projection pursuit has been poorly suited to deal with highly nonlinear structure, and 2.) projection pursuit methods tend to be computationally intensive.

2.2.1 What is an interesting projection?

An example of data projections is provided in figure (8) [12]. In this example, we project the original data onto a pair of arbitrary two-dimensional hyperplanes. Yet, we should hope to make such data projections only when those projections are "interesting." We consider that a projection is interesting if it contains structure [40]. Correlation between variables as detected by a linear regression is an example of easily recognizable structure in the data. Because of this, and noting the following results, an assessment of such structure can be made. For fixed variance, the normal distribution has the least information, in both the senses of Fisher information and negative entropy [17]. For most high-dimensional clouds, most low-dimensional projections are approximately normal [24]. Thus, it is considered that the normal distribution is the least structured (or least interesting) density.

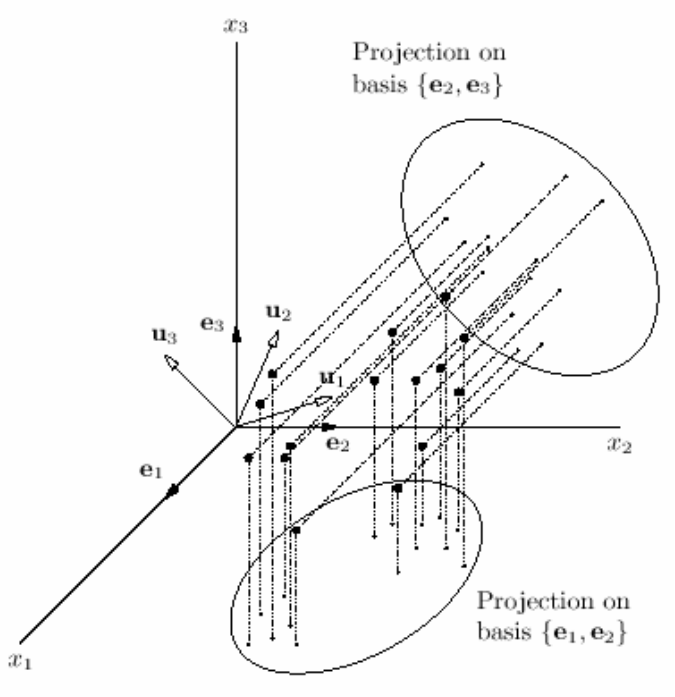

Figure 8: Illustration of Data Projections

2.2.2 The projection index

A projection index Q is a real functional on the space of distributions on R_k: [12], [40]

$$Q : f \in \mathcal{L}_2\left(\Re^k\right) \rightarrow q = Q_f \rightarrow \Re$$

Normally, $f = F_A$ will be the distribution of the projection (of matrix A) of a D-dimensional random variable X with distribution F, and will correspond to a k-dimensional random variable $Y = A^T X$, if A is $D \times k$. Projection pursuit attempts to find projection directions a_i for a given distribution F which produce local optima of Q. To make the optimization problem independent of the length of the projection vectors and to obtain uncorrelated directions, the a_i are constrained to be unit length and mutually orthogonal (i.e., the column vectors of A must be orthonormal).

2.2.3 Projection pursuit regression (PPR)

2.2.3.1 Background

Projection pursuit regression (PPR) [33] is a nonparametric regression approach for the multivariate regression problem based on projection pursuit. A remarkable feature of projection pursuit regression is that it is one of the few multivariate methods capable of bypassing the curse of the dimensionality to some extent. However, the power of any projection pursuit algorithm to find important structure will still suffer if the sample size is small and the dimension large. PPR works by additive composition, constructing an approximation to the desired response function by means of a sum of low-dimensional smooth functions, called ridge functions, each of which depend on low-dimensional projections through the data:

$$\widehat{f}(x) = \sum_{k=1}^{j} g_k\left(a_k^T x\right)$$

where each g_k is constant on hyperplanes.

2.2.3.2 Algorithm

The PPR algorithm determines a_k and g_k for k =1,..., j as follows: [12]

Initialization of algorithm: Set the projection directions a_k to some random vectors (or the first principal components in a PCA routine). The residuals are initially set to $r_{i\mathbf{0}} = y_i$. Set $j = 1$:

Repeat

1. Assuming a_k and g_k for k =1,..., $j-1$ determined, compute the residuals

$$r_{i,j-1} = y_i - \sum_{k=1}^{j-1} g_k\left(a_k^T x_i\right) = r_{i,j-2} - g_{j-1}\left(a_{j-1}^T x_i\right) \qquad i = 1, ..., n$$

2. Fit a nonparametric smooth curve g_j to the residuals $\{r_{i,j-1}\}_{i=1}^n$ as a function of $a^t x_i$ for any $a \in \Re^D$ with $\|a\| = 1$.

3. Projection pursuit step: minimize the sum of squared residuals (the L_2-norm) relative to g over a:

$$a_j = \arg\min_{\|a\|=1} \sum_{i=1}^{n} \left(r_{i,j-1} - g\left(a^T x_i\right)\right)^2 = \arg\min_{\|a\|=1} \sum_{i=1}^{n} r_{i,j}^2$$

4. Insert a_j, g_j as the next term in the equation for $\widehat{f}\,(x)$.

Until the improvement in step 3 is small.

2.2.4 A Brief History of Projection Pursuit

Projection pursuit regression was first developed by Friedman and Stuetzle [33] and expanded upon by [36]. As with other nonparametric methods, projection pursuit techniques possess certain useful properties. Nonparametric regression techniques were first devised for greater robustness and modeling capabilities when confronting estimation tasks where the functional form of the response surface is not known. Nonparametric techniques are especially useful in these situations as they make fewer assumptions about said response surface.

Projection pursuit regression (PPR) is especially useful when modeling higher dimensional data as it is capable of overcoming the curse of dimensionality as experienced in kernel and nearest-neighbor methods to some extent as all of the modeling performed is univariate. Interactions amongst predictor variables are directly modeled as PPR fits smooth curves to each univariate projection. However, because a PPR model is the superposition of low-dimensional functions, it will have trouble modeling surfaces that vary in strength equally across all possible linear combinations [52].

A comparison of PPR's prediction capabilities with that of kernel methods has been made by Donohoe and Johnstone [25]. They found that PPR works well when

the underlying function is angularly smooth, or oscillates slowly with angle, but performed poorly for harmonic analysis. Kernel models behaved well in these cases, where functions had sufficient Laplacian smoothness. They also showed that if the function to be estimated has nice tail behavior, PPR lowers the dimensionality. Chen [13] devised a PPR scheme such that the convergence rate of the estimator is independent of dimensionality.

Projection pursuit was first extended into the domain of learning networks by Barron and Barron [4]. A projection pursuit learning network (PPLN), is very similar in structure to a one hidden-layer neural network, except in place of the sigmoidal activation functions are unknown functions to be learned from the data. Thus, PPLN can be viewed as a generalization of sigmoidal feedforward neural networks.

In the original PPLN, a variable span smoother, dubbed the "supersmoother" [33], is used to generated the smooth estimated activation functions. The motivation behind the original design of the supersmoother was two-fold: a.) to have a good variable bandwidth adaptable to varying function curvature and noise levels, and b.) to be very fast, computationally. Hwang et al. [41], [42] revealed that the supersmoother, and actually such nonparametric smoothers in general, have inherent problems – such as the use of large regression tables, unstable derivative approximations, and piecewise interpolation in calculating activation values that leads to performance degradation in training and testing. They proposed using a parametric smoother that would be the constructed from a superposition of Hermite functions, where the Hermite functions are defined as

$$h_r(z) = (r!)^{-1/2} \pi^{1/4} 2^{-(r-1)/2} H_r(z) \phi(z),$$

where $-\infty < z < \infty$ and $H_r(z)$ are Hermite polynomials, expressed as

$$\begin{aligned} H_0(z) &= 1 \\ H_1(z) &= 2z \\ H_r(z) &= 2(rH_{r-1}(z) - (r-1)H_{r-2}(z)). \end{aligned}$$

The Hermite-based PPLN uses the minimum $\mathcal{L}_2$ criterion, produces smoother regression surfaces, and is inherently easy to arrive at the derivative calculations. In this thesis, we will extend upon this with another set of parametric networks that are capable of utilizing a broader, more general class of basis functions.

Zhao et al. [95] investigated the use of a parametric PPR to learn the inverse dynamics of robot arms in high-dimensional space (six dimensions). They showed that PPLNs can learn this task quite well and that a parametric PPR with a direct training method can achieve better accuracy and training speed than a nonparametric PPR. Also, the parametric projection pursuit network has the advantage of achieving a higher degree of accuracy with fewer estimation parameters used than does a one hidden layer sigmoidal neural network.

Kwok and Yeung [54] improved upon the PPLN suggested by Hwang et al. [41], [42]. Recall that in those papers the parametric smoother is based on a predefined order, R, of the Hermite function. Kwok and Yeung found that a PPLN with a fixed R does not possess the universal approximation capability for any finite value of R. Thus, they suggest that is it possible to keep R fixed while still retaining the property of universal approximation by introducing a bias term to each linear combination of predictors. They also demonstrate experimentally that this change increases the rate of convergence with respect to the number of hidden units and improves the model's generalization capabilities. We will be extending upon this with the choice

of a different set of bases for the local fits along projection directions and with a new universal approximation theorem showing that our approach to approximating the response surface maintains its universal approximation capabilities for a broader class of functions than was previously demonstrated.

2.2.5 Projection Pursuit Learning Networks (PPLN)

A projection pursuit learning network (PPLN) is similar in structure to a one hidden layer sigmoidal feedforward neural network. The PPLN [42], [95] can be written as

$$\widehat{y}_i = \overline{y}_i + \sum_{k=1}^{m} \beta_{ik} f_k \left(\sum_{j=1}^{p} \alpha_{kj} x_j \right)$$

where β_{ik} are the projection strengths, f_k are the unknown smooth activation functions, and α_{kj} are the projection directions. These parameters are trained by minimizing the mean squared error loss function:

$$L \equiv \sum_{i=1}^{q} W_i E \left(y_i - \widehat{y}_i \right)^2$$

where the weights W_i describe the relative contribution of each mean squared output error to the total loss function, L, and E is the expected value function, defined as

$$E \left(y_i \right) = \frac{1}{n} \sum_{l=1}^{n} y_{li} = \overline{y}_i$$

The PPLN learning algorithm trains each hidden units sequentially instead of simultaneously, as is the case with backpropagation networks. The algorithm for the k-th hidden layer neuron can be represented as follows: [52]

1. Make initial guess for α_k, β_k, and f_k

2. Estimate $\widehat{\alpha}_k = a_k + \Delta$ using an iterative optimization method

3. Given $\widehat{\alpha}_k$, estimate f_k as the smooth 1-dimensional curve that best fits the scatterplot $\left[z_{kl}, f_k^* \left(z_{kl} \right) \right]$, where $z_{kl} = \widehat{\alpha}_k^T x_l$.

4. Repeat 2-3 for several iterations.

5. Use the most recent updates of f_k and α_k to calculate β_{ik} by setting to zero the derivatives of the loss function, L, with respect to β_{ik}.

6. Repeat steps 2-5 until the loss function is minimized with respect to all f_k, α_k, and β_{ik} associated with the k-th neuron.

This procedure is then repeated for the $(k+1)$-th hidden layer neuron. A schematic is provided in figure (9) [52].

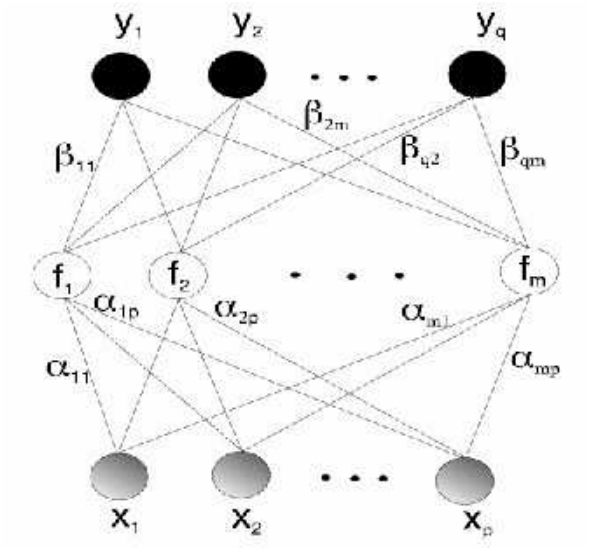

Figure 9: Schematic of PPLN architecture

Maeschler et al. [58] and Hwang et al.[41], [42] compared the performance of projection pursuit learning networks with backpropagation neural networks using two-dimensional regression problems. In each case, they used nonparametric data smoothers to optimize the activation functions in the PPLN, and found that both methods achieved similar performance on independent cross-validation samples. The work by Hwang et al. [41], [42] illuminates some problems with the nonparametric PPLN approach, namely the use of large regression tables and the inherent instability in estimating the derivatives. Thus, they propose using the superposition of parametric Hermite functions in place of the nonparametric smoother.

Hwang et al. [42] compared PPLN with cascade-correlation learning networks (CCLN), and found CCLN models unsuitable for most regression applications as their structures are prone to saturated hidden units leading to unsmooth, jumpy estimates.. Like a PPLN, CCLN grows ts hidden layer(s) during training by sequentially adding hidden units. With the CCLN, the weights on each candidate hidden unit are trained by holding constant all existing weights. Unlike as is the case with a PPLN, in which the input data form the entirety of the connections feeding into each hidden unit, each candidate unit in a CCLN will receive connections from both input units and from all other hidden units. While this attribute aids the CCLN in finding higher order features, it also makes the training more difficulty. Hwang et al. found that the maximum correlation criterion used in the network to avoid cyclic updating between layers usually produces saturated hidden units, resulting in unsmooth regression surfaces, which make CCLN unsuitable for most regression applications.

2.2.6 Remarks

In the next chapter, a new type of projection pursuit learning model is introduced. In our approach, we optimize projection directions simultaneously to solve the function approximation problem using analytical methods. On the block diagram in figure (10), we will be exploring the theory, optimization procedure, and experimental implementation results for the case of unbounded functions (note that in chapter 4, we will review the case of bounded functions). Specifically, we will provide the foundational theory behind just such a methodology, including a proof of its universal approximation capabilities. We then discuss the optimization approach employed along with providing a detailed algorithm. We conclude with a comparison of the simulation results of our projection pursuit learning model with three commonly-used high-dimensional modeling methods.

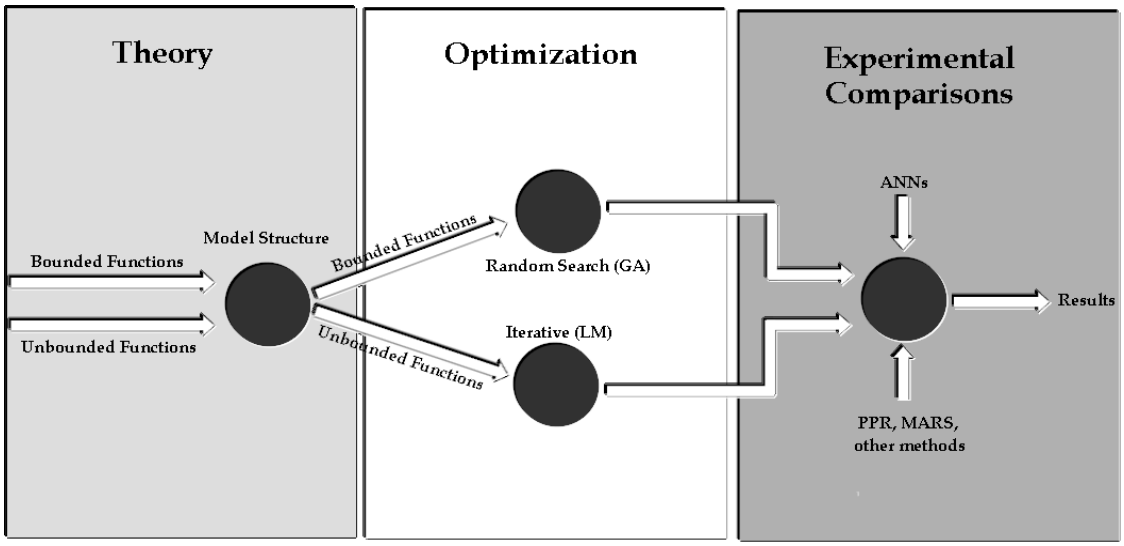

Figure 10: Block Diagram of Modeling Approaches

CHAPTER III

THE CONTINUOUS PROJECTION PURSUIT LEARNING MODEL (PPLM)

3.1 Overview

The chapter is structured in the following manner. In the next section, we present the theory to show that a response surface can be modeled by the superposition of one-dimensional functions. Next, the issue of basis functions is addressed along with the associated grid spacing. In an effort to match the input distribution, a data-driven assignment of the grid spacing is suggested, which typically results in unequally-spaced nodes. The section following this proves the universal approximation capabilities of the projection pursuit learning model. The two-stage optimization problem is then presented. The algorithm, which utilizes a Levenberg-Marquardt optimization of the projection directions, is then provided. Finally, before the concluding remarks, an experimental case study is offered comparing prediction performance with other traditional high-dimensional learning techniques. Please note that in the interest of readability, some of the theoretical derivations of this chapter have been moved to the final section of the chapter, section...

3.2 Mathematical Framework

Before beginning the analysis, it is useful to first define the mathematical framework within which we will be working. Our goal is to approximate functions belonging to a subset of Lebesgue measurable multi-dimensional functions $f : \Re^n \rightarrow \Re$. The $\mathcal{L}_p$

norm of $f : \Re^n \rightarrow \Re$ is denoted by

$$\|f\|_p = \begin{cases} \left(\int_{\Re^n} |f|^p dx\right)^{1/p} & , 1 \leq p < \infty \\ \text{ess} \sup\limits_{x \in \Re^n} |f(x)| & , \quad p = \infty \end{cases}$$

where ess denotes the essential supremum (i.e., supremum of f except for a set of measure zero). The set $\mathcal{L}_p$ refers to the class of Lebesgue measurable functions with a finite $\mathcal{L}_p$ and norm: $\|f\|_p < \infty$. Unless otherwise specified, we shall consider the class of functions f belonging to $\mathcal{L}_2$.

3.2.1 Problem Statement

Given an unknown multivariate function of dimension $n \gg 1$, $f \in \mathcal{L}_2$, our objective is to approximate f to within a prescribed degree of accuracy based on a finite set of input–output data (x_k, y_k), $k = 1, \ldots, N$ by projecting f along a finite set of directions and constructing a nonlinear model composed of the superposition of 1–dimensional functions along the projection directions that best fit the input–output data.

3.2.2 One–dimensional Decomposition

In this section, we show that a function belonging to a subset of $\mathcal{L}_2$ can be expressed as a superposition of 1–dimensional functions, which will form the basis for our projection pursuit approximation. As can be seen this decomposition is of f is closely related to its Fourier transform pair defined below:

$$\hat{f}(\omega) = \int_{\Re^n} f(x) e^{-j\boldsymbol{\omega}^T x} dx$$
$$\check{f}(x) = \frac{1}{(2\pi)^n} \int_{\Re^n} \hat{f}(\omega) e^{j\boldsymbol{\omega}^T x} d\omega$$

In general $\check{f}(x) \neq f(x)$ (pointwise). But it is true for rapidly decreasing functions [68] defined by

$$\mathcal{F}_n := \left\{ f \in \mathcal{L}_2 : \sup_{|\alpha| \leq N} \sup_{x \in \Re^n} \left(1 + \|x\|^2\right)^N \left| \frac{\partial^{|\alpha|} f(x)}{\partial x_1^{\alpha_1} \cdots \partial x_n^{\alpha_n}} \right| < \infty, \; N = 0, 1, 2, \ldots \right\} \quad (1)$$

where $\alpha = (\alpha_1, \ldots \alpha_n)$, $\alpha_i = 0, 1, 2, \ldots$, is a multi–index and $|\alpha| \leq \sum_{i=1}^{n} \alpha_i$. Where the function space $\mathcal{F}_n$ above is a Frechet space [68], that is if $f \in \mathcal{F}_n$, then $f \equiv \check{f}$.

Next we define the set of all possible projection directions. It turns out that this set can be identified by the surface of the one–half unit hypersphere in $\Re^n$ given by

$$\mathcal{U} = \left\{ \omega \in \Re^n : \|\omega\|^2 = \sum_{i=1}^{n} \omega_i^2 = 1,\ \omega_j > 0,\ j = \min_{1 \leq i \leq n} \omega_i \neq 0 \right\}$$

then it can be easily seen that $\Re^n = \cup_{r \in \Re} r\mathcal{U}$. For a function $f \in \mathcal{F}_n$ we have

$$f(x) = \frac{1}{(2\pi)^n} \int_{\Re^n} \hat{f}(\omega) e^{j\boldsymbol{\omega}^T x} d\omega$$

The following lemma can be used to determine the volume and surface area of $\mathcal{U}$.

Lemma 7 *The generalized volume and surface area of a hypersphere in n dimensions defined by* $\sum_{i=1}^{n} x_i^2 = R^2$ *is given by*

$$V_n(R) = \frac{\pi^{n/2} R^n}{\Gamma(\frac{n}{2} + 1)}$$
$$S_n(R) = \frac{dV_n}{dR} = \frac{n\pi^{n/2} R^{n-1}}{\Gamma(\frac{n}{2} + 1)}$$

where $\Gamma(x)$ *is the gamma function given by* $\Gamma(x) = \int_0^\infty t^{x-1} e^{-t} dt$ *[6].*

Using $\Re^n = \cup_{r \in \Re} r\mathcal{U}$ combined with the preceding lemma, the Fourier integral over $\Re^n$ may be expressed as

$$\int_{\Re^n} \hat{f}(\omega) e^{j\boldsymbol{\omega}^T x} d\omega = \int_{\Re} \int_{r\mathcal{U}} \hat{f}(ru) e^{jr\mathbf{u}^T x} dS_r dr$$

where dS_r is the differential of the surface area of the hyper-hemisphere of radius r, i.e.,

$$\int_{r\mathcal{U}} dS_r = \frac{n\pi^{n/2} r^{n-1}}{2\Gamma(\frac{n}{2} + 1)}$$

In particular, denoting the surface area of the unit hyper-hemisphere by dS, then $\int_{\mathcal{U}} dS = \frac{n\pi^{n/2}}{2\Gamma(\frac{n}{2}+1)}$ implying that $dS_r = r^{n-1} dS$. Thus

$$\int_{\Re^n} \hat{f}(\omega) e^{j\boldsymbol{\omega}^T x} d\omega = \int_{\mathcal{U}} \left[\int_{\Re} r^{n-1} \hat{f}(ru) e^{jr\mathbf{u}^T x} dr \right] dS$$

Defining the one–dimensional function

$$f_u(z) = \frac{1}{(2\pi)^n} \int_{\Re} r^{n-1} \hat{f}(ru) e^{jrz} dr,\ z = u^T x$$

then

$$f(x) = \int_{\mathcal{U}} f_u(z) dS$$

The next theorem shows that the one-dimensional projection functions $f_u(z)$ are mutually orthogonal with respect to the function space scalar product defined below:

Definition 8 *The scalar product of two functions $f, g \in \mathcal{L}_2$ is defined as*

$$< f, g >= \int_{\Re^n} f(x) g(x) dx$$

Remark 9 *Note that this expression is well-defined for any two functions $f, g \in \mathcal{L}_2$. By the Schwartz inequality,*

$$|< f, g >| = \left| \int_{\Re^n} f(x) g(x) dx \right| \leq \|f\|_2 \|g\|_2 < \infty$$

Theorem 10 *A function $f \in \mathcal{F}_n$ can be decomposed into an* infinite *number of single variable, mutually orthogonal functions $f_{\mathbf{u}}$, with $f_{\mathbf{u}} \in \mathcal{F}_1$ and $u \in \mathcal{U}$, i.e., $< f_{\mathbf{u}}, f_{\mathbf{u}'} >= 0$, $\mathbf{u} \neq \mathbf{u}'$:*

$$f(x) = \int_{\mathcal{U}} f_u(z) dS \tag{2}$$

Proof. First, the assertion that $f_{\mathbf{u}} \in \mathcal{F}_1$ follows from Theorem 7.7 in Rudin's book, "Functional Analysis" [68], as the Fourier transform maps $\mathcal{F}_n$ functions onto $\mathcal{F}_n$. Thus, the only thing remaining to prove is the orthogonality of these functions.

$$\begin{aligned} \langle f_u, f_v \rangle &= \frac{1}{(2\pi)^{2n}} \int_{\Re^n} \left(\int_{\Re} r^{n-1} \hat{f}(ru) e^{jru^T x} dr \right) \left(\int_{\Re} s^{n-1} \hat{f}(sv) e^{jsv^T x} ds \right) dx \\ &= \frac{1}{(2\pi)^{2n}} \int_{\Re} \int_{\Re} r^{n-1} s^{n-1} \hat{f}(ru) \hat{f}(sv) \int_{\Re^n} e^{j(ru - sv)^T x} \end{aligned}$$

The 2nd integral on the right hand side of the preceding equation is

$$\int_{\Re^n} e^{j(ru-sv)^T x} = \int_{\Re} e^{j(ru_1 - sv_1)x_1} dx_1 \cdots \int_{\Re} e^{jr(ru_n - sv_n)x_n} dx_n \neq 0$$

if and only if

$$\delta(ru_i - sv_i) = \frac{1}{2\pi}\int_{\Re} e^{j(ru_i - sv_i)x_i} dx_i \neq 0,\ i = 1, \ldots, n$$

or equivalently $ru = sv$. Since $\|u\| = \|v\|$, we must necessarily have that $r = \pm s$ or $u = \pm v$. But based on the definition of $\mathcal{U}$ it is impossible that $u = -v$ for nonzero u and v. Thus $u = v$, and

$$\langle f_u, f_v \rangle = \begin{cases} \|f_u\|^2,\ u = v \\ 0,\ u \neq v \end{cases}$$

■

Given that these functions are mutually orthogonal, deriving the prediction as a superposition of low-dimensional functions should work as these functions will be theoretically decoupled.

3.3 Bases of one-dimensional functions

Thus far, we have shown that we can deconstruct any well-behaved continuous functions into a series of mutually orthogonal one-dimensional continuous functions. Our ultimate goal is to approximate multi-dimensional functions. In this section, we shall prove that for each projection direction, we can approximate our fit with enough one-dimensional bases to reach an arbitrary degree of closeness. Then, in the section that follows we can move on to the universal approximation theorem for multi-dimensional functions. There, we shall show that the results in this section can be extended to include a reasonably large class of high-dimensional functions that can be approximated within an arbitrary degree of closeness by the sum of one-dimensional functions.

We should begin with a definition of a basis function [35].

Definition 11 *Let $\mathcal{C}$ denote a subspace of the space of real continuous functions, $f : \Re \to \Re$. Consider a countable set of linearly independent $\{\phi_i \in \mathcal{C}\}$ such that*

1. *unity can be expressed as a linear combination of finitely many* ϕ_i*'s.*

2. *the span of* $\{\phi_i\}$ *is dense in* $\mathcal{C}$*, that is, for any* $f \in \mathcal{C}$ *and* $\varepsilon > 0$*, there exists an* N *and* $w_i \in \Re$ *such that:*

$$\sup_{z \in \mathcal{R}} \left| f(z) - \overline{f}(z) \right| < \varepsilon$$

where $\overline{f}(z) = \sum_{i=0}^{N} w_i \phi_i(z)$.

There are two broad classes of basis functions: global and local.

3.3.1 Global

Examples of global bases would be:

Fourier basis: $\phi_k = e^{j\omega_k^T \mathbf{x}}$, where $\omega_k = 2\pi k$ and where $\mathcal{C}$ is the class of periodic functions.

Polynomial basis: $\phi_k = \mathbf{x}^k$, where $k = 0, \pm 1, \pm 2, ...$ and where $\mathcal{C}$ is the class of functions defined on a compact set.

A famous theorem of Weierstrass [59] states that any continuous function with very general properties may be approximated to arbitrary degree of precision.

Theorem 12 *Stone-Weierstrass Theorem:*

Let $\mathcal{X} \subset \Re^n$ *be compact and let* $\mathcal{B}$ *be a subset of continuous functions,* $f : \mathcal{X} \to \Re$ *with the following properties:*

1. $\mathcal{B}$ *is an algebra; i.e.,* $f, g \in \mathcal{B}$, $\alpha \in \Re \Longrightarrow f + g \in \mathcal{B}$, $f \cdot g \in \mathcal{B}$, *and* $\alpha f \in \mathcal{B}$*;*

2. $\mathcal{B}$ *contains a non-zero constant function;*

3. *$\mathcal{B}$ separates points; i.e., for $x, y \in \mathcal{X}$, $x \neq y$ there is an $f \in \mathcal{B}$ such that $f(x) \neq f(y)$;*

Then $\mathcal{B}$ is dense in $\mathcal{C}$; that is for each $f \in \mathcal{C}$ and $\epsilon > 0$, there exists a function $g \in \mathcal{B}$ such that $\sup_{z \in \mathcal{X}} |f(x) - g(x)| < \epsilon$.

And so we see that both polynomial and exponential functions can approximate continuous functions to an arbitrary degree of precision. What remains to be proven is the universal approximation capability of the second broad class of basis functions: the local bases.

3.3.2 Local Basis Functions

Local bases are often good choices for basis functions because they are quite adaptive to varying function surfaces [81]. In selecting a basis function, basically, we are looking for a function with good approximation capabilities within a local region, as will be described in more detail. Local basis functions have the capability to effectively model highly nonlinear data, as the function approximation in one region of the input space $S_1 \cap S_2 = \emptyset$, $x_1 \in S_1$, will not alter the approximation of the function in another region of the sample space $x_2 \in S_2$, where $S_1 \cap S_2 = \emptyset$, that may be governed by a completely different set of rules.

One possible choice of basis functions is the piecewise linear (PWL) basis. In this case, $\mathbf{x} = x$ is a scalar and the PWL basis functions reduce to simple "tent" functions. For our purposes, we shall utilize cubic bases, as the order of the error of their approximation is vastly improved.

Let us now consider the weighted basis functions on the domain,

$S \in \{x : 0 \leq x \leq h\}$.

$$W_1 \phi_1(x) = W_1 \left(1 - \frac{x}{h}\right) \tag{3}$$

$$W_2\phi_2(x) = W_2\left(1 + \frac{x-h}{h}\right) = W_2\left(\frac{x}{h}\right) \tag{4}$$

The above equation pair represents the intersection of two tent functions on the domain. Here the top of the first "tent" is affixed at $x = 0$. The top of the second tent is located at the edge $x = h$.

Utilizing the capabilities of the Taylor series, we can represent a function $f(x)$ about zero on this domain as $f(x) = f(0) + f'(0) \cdot x + O(h^2)$. Plugging in for the derivative $f'(0) = \frac{1}{h}[f(h) - f'(0)] + O(h^2)$, the function can be written as:

$$f(x) = f(0) + \frac{1}{h}[f(h) - f'(0)] + O\left(h^2\right) \tag{5}$$

for data points on x between 0 and h.

One convenient property of the PWL basis functions is that weight, or amplitude, of each "tent" at its center (the top of the "tent") is exactly equal to the approximation of the function at that point. The reason is that even though the PWL basis functions overlap, all other basis functions equal zero at any given basis function's center. In other words, at the center of the tent of any given basis function, only one basis function is turned on. We can see this to be the case from the above example, as $\phi_1(h) = 0$ and $\phi_2(0) = 0$ in the equations (3) and (4) above. Resulting from this, we can see that $f(0) = W_1\left(1 - \frac{x}{h}\right) = W_1$ and $f(h) = W_2\left(\frac{x}{h}\right) = W_2$. Substituting into equation 5 gives $f(x) = W_1\left(1 - \frac{x}{h}\right) + W_2\left(\frac{x}{h}\right) + O(x^2)$, or $f(x) = W_1\phi_1(x) + W_2\phi_2(x) + O(x^2)$, where $0 \leq x \leq h$.

The above proof yields the $O(x^2)$ order of approximation. Thus, the greater the number of basis functions that we use, packed ever more closely together on the domain $\mathcal{S}$, the better will our approximation be.

Note that the set of one-dimensional basis functions can easily be generalized for any one-dimensional domain with the following framework. Assume that $x \in A$ with

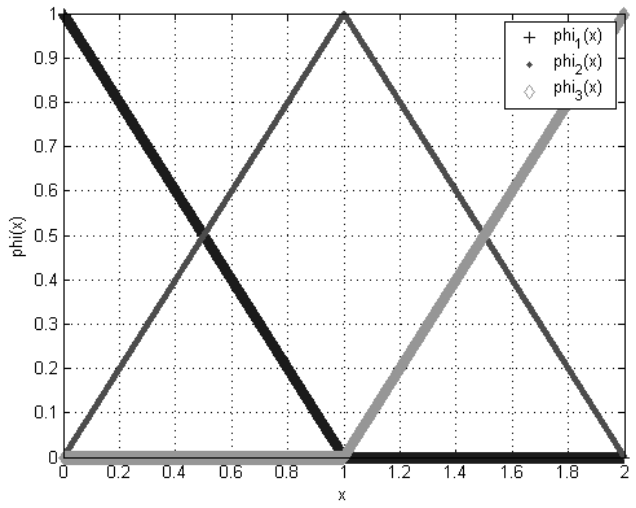

Figure 11: Local Piecewise Linear Basis Functions

the input space shaped such that $A = [\alpha, \beta]$. Allowing this interval to be equally divided into N subintervals, we find that the center of each triangular basis function described by these subintervals can be denoted as $\eta_i = \alpha + i\left(\frac{\beta-\alpha}{N}\right)$, while the base of each equal-sized interval can be written as: $b = \frac{\beta-\alpha}{N}$.

Provided in figure (11) is an illustration of the local piecewise linear bases. Note that, for this example where $h = 1$, $\alpha = 0$, and $\beta = 1$, only the bases, ϕ_1 and ϕ_2, are active in the region where $0 < x < 1$.

Thus, if we define $\overline{x}_i = \frac{x-\eta_i}{b}$ where each $\overline{x}_i$ is a real number between -1 and 1, then $\phi_i = \left\{ \begin{array}{cc} 1 - |\overline{x}_i| & \text{if } |\overline{x}_i| \leq 1 \\ 0 & \text{otherwise} \end{array} \right\}$.

Another example of local basis functions would be local piecewise cubic bases.

3.3.2.1 Local Piecewise Cubic Basis Functions

From the Taylor Series approximation, we can see that the order of approximation improves considerably:

$$f(x) = w_1\phi_r^1 + w_2\phi_l^1 + v_1\phi_r^2 + v_2\phi_l^2 + O\left(x^4\right) \tag{6}$$

for data points on x between 0 and h.

Working out the computation of these local bases yields:

$\phi_l^1 = 3\overline{x}_r^2 - 2\overline{x}_r^3$

$\phi_r^1 = 1 - 3\left|\overline{x}_r\right|^2 + 2\left|\overline{x}_r\right|^3$

$\phi_l^2 = x\left(\overline{x}_r^2 - \overline{x}_r\right)$

$\phi_r^2 = h\overline{x}_r\left(1 - 2\left|\overline{x}_r\right| + \left|\overline{x}_r\right|^2\right)$

Where $\overline{x} = \frac{x-c}{b}$, with c as the position of the node in question and b is the base.

We see that:

$$\overline{x} = \left\{ \begin{array}{ll} \frac{x}{h} = \overline{x}_r & \text{, for } x > c \\ \frac{x-h}{h} = \overline{x}_l & \text{, for } x \leq c \end{array} \right\} \text{ and } |\overline{x}| = \left\{ \begin{array}{ll} \frac{x}{h} = |\overline{x}_r| & \text{, for } x > c \\ 1 - \frac{x}{h} = |\overline{x}_l| & \text{, for } x \leq c \end{array} \right\}$$

So, $\overline{x}_l = \overline{x}_r - 1$, and $|\overline{x}_l| = 1 - |\overline{x}_r| = 1 - \overline{x}_r$, and $|\overline{x}_r| = \overline{x}_r$

A proof of this is provided in the next section.

An example illustration of these bases is provided in figures (12) and (13).

Note that figure (13) actually represents the derivative portion of the local cubic bases.

3.3.3 Derivation of Local Cubic Basis Functions

To derive the local cubic bases, we turn to the Taylor Series approximation:

$f(x) = f(0) + f'(0)\,x + f''(0)\,\frac{x^2}{2} + f'''(0)\,\frac{x^3}{3!} + O\left(x^4\right)$

Assigning w_2 and $\overline{v}_2$ to be:

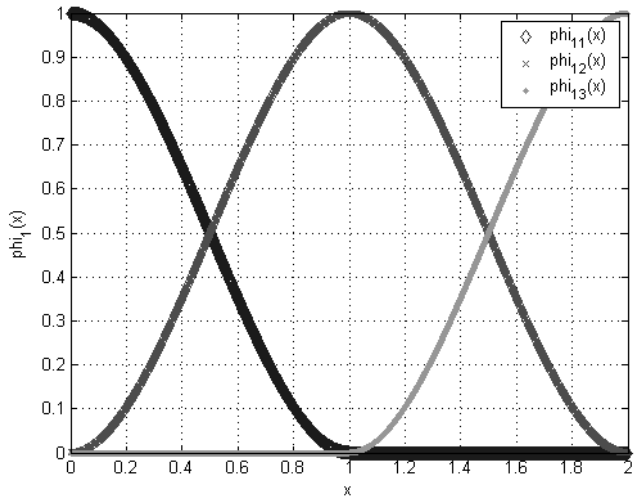

Figure 12: Local Piecewise Cubic Basis Functions

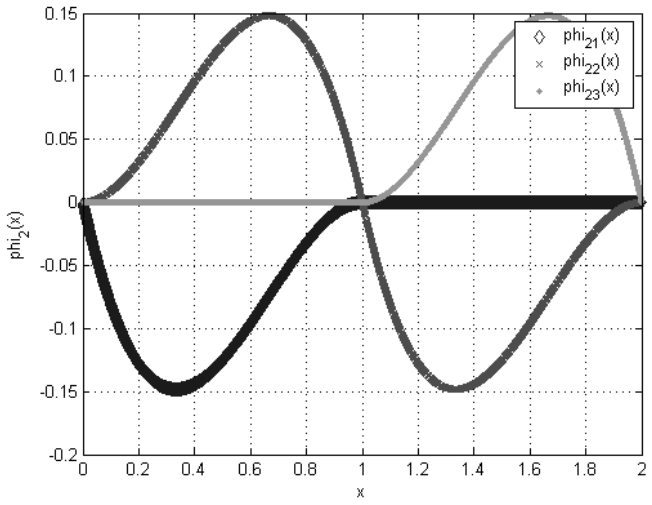

Figure 13: Local Piecewise Cubic Basis Functions (Derivative Portion)

$$w_2 = f(h) = w_1 + \overline{v}_1 h + ah^2 + bh^3 + O(h^4)$$

$$\overline{v}_2 = f'(h) = w_1 + \overline{v}_1 + 2ah + 3bh^2 + O(h^3)$$

We can then solve for a and b :

$$\begin{bmatrix} h^2 & h^3 \\ 2h & 3h^2 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} w_2 - w_1 - \overline{v}_1 h + O(h^4) \\ \overline{v}_2 - \overline{v}_1 + O(h^3) \end{bmatrix}$$

$$\begin{bmatrix} a \\ b \end{bmatrix} = \frac{1}{h^4} \begin{bmatrix} 3h^2 & -h^3 \\ -2h & h^2 \end{bmatrix} \begin{bmatrix} w_2 - w_1 - \overline{v}_1 h + O(h^4) \\ \overline{v}_2 - \overline{v}_1 + O(h^3) \end{bmatrix}$$

$$\begin{bmatrix} a \\ b \end{bmatrix} = \frac{1}{h^4} \begin{bmatrix} 3h^2(w_2 - w_1) - 2\overline{v}_1 h^3 - \overline{v}_2 h^3 + O(h^6) \\ -2h(w_2 - w_1) + h^2(\overline{v}_2 + \overline{v}_1) + O(h^5) \end{bmatrix}$$

Plugging these into the Taylor Series formulation:

$$\begin{aligned} f(x) &= w_1 + \overline{v}_1 x + \left[\frac{3}{h^2}(w_2 - w_1) - \frac{1}{h}(2\overline{v}_1 + \overline{v}_2) + O\left(h^2\right) \right] x^2 \\ &\quad + \left[\frac{-2}{h^3}(w_2 - w_1) - \frac{1}{h^2}(\overline{v}_1 + \overline{v}_2) + O(h) \right] x^3 + O\left(x^4\right) \end{aligned}$$

Collecting terms yields the following expression:

$$\begin{aligned} f(x) &= w_1 \left(1 - 3\left(\frac{x}{h}\right)^2 + 2\left(\frac{x}{h}\right)^3 \right) + w_2 \left(3\left(\frac{x}{h}\right)^2 - 2\left(\frac{x}{h}\right)^3 \right) \\ &\quad + \overline{v}_1 x \left(1 - 2\left(\frac{x}{h}\right) + \left(\frac{x}{h}\right)^2 \right) + \overline{v}_2 x \left(-\left(\frac{x}{h}\right) + \left(\frac{x}{h}\right)^2 \right) + O\left(x^4\right) \end{aligned}$$

This can be concisely written as

$$f(x) = w_1 \phi_r^1 + w_2 \phi_l^1 + \overline{v}_1 x \phi_r^2 + \overline{v}_2 x \phi_l^2 + O\left(x^4\right)$$

Substituting $v_1 = \overline{v}_1$ and $v_2 = \overline{v}_2$ yields the local cubic approximation, which is seen to be a fourth-order approximation:

$$f(x) = w_1 \phi_r^1 + w_2 \phi_l^1 + v_1 \phi_r^2 + v_2 \phi_l^2 + O\left(x^4\right)$$

Assigning $\overline{x} = \frac{x-c}{b}$, where c is the position of the node in question and b is the base. We see that:

$$\begin{aligned} \overline{x} &= \left\{ \begin{array}{lll} \frac{x}{h} = \overline{x}_r & , & \text{for } x > c \\ \frac{x-h}{h} = \overline{x}_l & , & \text{for } x \leq c \end{array} \right\} \\ |\overline{x}| &= \left\{ \begin{array}{lll} \frac{x}{h} = |\overline{x}_r| & , & \text{for } x > c \\ 1 - \frac{x}{h} = |\overline{x}_l| & , & \text{for } x \leq c \end{array} \right\} \end{aligned}$$

So, $\overline{x}_l = \overline{x}_r - 1$, and $|\overline{x}_l| = 1 - |\overline{x}_r| = 1 - \overline{x}_r$, and $|\overline{x}_r| = \overline{x}_r$. From the equations that preceded, we note that: $\phi_l^1 = 3\overline{x}_r^2 - 2\overline{x}_r^3 = 1 - 3\left|\overline{x}_r\right|^2 + 2\left|\overline{x}_r\right|^3$. Plugging in for $|\overline{x}_l|$ and rewriting yields:

$$\begin{aligned} \phi_r^1(x_l) &= 1 - 3(1 - \overline{x}_r)^2 + 2(1 - \overline{x}_r)^3 \\ &= 1 - 3\left(1 - 2\overline{x}_r + \overline{x}_r^2\right) + 2(1 - \overline{x}_r)\left(1 - 2\overline{x}_r + \overline{x}_r^2\right) \\ &= 1 - 3 - 6\overline{x}_r - 3\overline{x}_r^2 + 2\left(1 - 3\overline{x}_r + 3\overline{x}_r^2 - \overline{x}_r^3\right) \\ &= 3\overline{x}_r^2 - 2\overline{x}_r^3 = \phi_l^1 \end{aligned}$$

Thus, we really have just one ϕ^1 basis function for the left and right sides. The only difference is that we can just plug in $\overline{x}_r$ and $\overline{x}_l$ to distinguish between its values on the two sides of the center of the node. With a little more algebra, we can see that the same thing is true for the ϕ^2 basis function. Again, we note that:

$$\begin{aligned} \phi_l^2 &= x\left(\overline{x}_r^2 - \overline{x}_r\right) \\ \phi_r^2 &= h\overline{x}_r\left(1 - 2\left|\overline{x}_r\right| + \left|\overline{x}_r\right|^2\right) \end{aligned}$$

Plugging in for $|\overline{x}_l|$ and rewriting:

$$\begin{aligned}
\phi_r^2(x_l) &= h(\overline{x}_r - 1)\left(1 - 2(1 - \overline{x}_r) + (1 - \overline{x}_r)^2\right) \\
&= h(\overline{x}_r - 1)\left(1 - 2\overline{x}_r + 1 - 2\overline{x}_r + \overline{x}_r^2\right) = h(\overline{x}_r - 1)\overline{x}_r^2 \\
&= x\left(\overline{x}_r^2 - \overline{x}_r\right) = \phi_l^2
\end{aligned}$$

Please note that there also are other choices for localized bases, such as wavelets, which are particularly effectively at modeling time-scale problems and behave like the Fourier bases but also include frequency location information [81]. The researcher is certainly welcome to and is, indeed, encouraged to utilize the local basis function of his or her choice when applying the projection pursuit learning methods developed in this thesis.

3.3.4 Grid Spacing

Thus far, we have yet to address how one determines the spacing of the nodes. Indeed, one potential problem with a local piecewise fit rests with the data distribution. If the data is not uniformly distributed, the grid spacing may be adjusted to match this input distribution such that a certain fraction of data points fall between each node. This enables the local models to generate very accurate fits in dense regions of the input space without overfitting the sparse regions. Thus, we suggest creating the order statistics [30] from the data for each observation. The nodes can be shifted to enforce an equal number of observations per segment. Note that we are utilizing an unsupervised approach to the grid-spacing adjustment; it is based solely on the data distribution. A supervised approach could potentially be used for optimal effect. However, in the simulations that follow later in the paper, we have found that the improvement is marginal in most cases, but the computational cost is quite substantive.

3.4 Universal Approximation Capability

In the prior section, we discussed approximating one-dimensional functions. Now, we shall move onto multi-dimensional functions. In this section, we shall attempt to show that a fairly large class of high-dimensional functions can be approximated by the sum of one-dimensional functions as discussed in the previous section. Within the context of our algorithmic implementation, we would like to numerically approximate the integral of the projection directions over the surface of an m-dimensional hypersphere. Thus, we must show that this approach of numerical integration over a finite set can indeed approximate the continuous integral of a compact set within an arbitrary degree of closeness. Or, in other words, we must prove the universal approximation capabilities of our discretized approach.

Theorem 13 *Let f be an arbitrary function, with $f \in \mathcal{F}_n$, where $\mathcal{F}_n$ is a class of Frechet functions as defined in equation (1). Let $\mathcal{X}$ be a compact subset of $\Re^n$. For any $\varepsilon > 0$, we can find a finite number of directions $u_1, \ldots, u_M$ and basis functions per direction ϕ_{ij}, $i = 1, \ldots M$, $j = 1, \ldots n_i$, such that the resulting approximation will estimate f to within ε, that is $\sup_{x \in \mathcal{X}} \left| f(\mathbf{x}) - \sum_{i=1}^{M} \sum_{j=1}^{n_i} w_{ij} \phi_{ij}(\mathbf{z}) \right| < \varepsilon$, where $\mathbf{z} = x^T u$, ϕ is the matrix of basis functions, and w is the corresponding weights as defined in section* (3.3).

The proof of the universal approximation theorem is based on lemma (19), which is important in its own right. The lemma whose proof is given in at the end of this chapter, in section (3.8.1), shows that

$$f(x) = \int_{S^{m-1}} f_u(z) dS \tag{7}$$

in theorem (10) can be approximated by a finite sum of one-dimensional functions. But, now we continue on to the proof of theorem (13).

Proof. From lemma (19) it follows that

$$f = \int_{\mathcal{S}^{m-1}} f_u(z_1, ..., z_m)\, dS = \sum_{i=1}^{M} \widehat{f}_{u_i}(z) + \widetilde{f}_u(z)$$

where $\left|\widetilde{f}_u(z)\right| < \left|\frac{\epsilon}{2}\right|$. For each direction, u_i, we shall choose enough basis functions, n_i, such that

$$\widehat{f}_{u_i}(z) = \sum_{j=1}^{n_i} \widehat{f}_{u_{ij}}(z) + \widetilde{f}_{u_i}(z)$$

where $\widehat{f}_{u_{ij}}(z) = w_{ij}\phi_{ij}(z)$ with w_{ij} as the weights and $\phi_{ij}(z)$ as the basis functions, and $\left|\widetilde{f}_{u_{ij}}(z)\right| < \left|\frac{\epsilon}{2M}\right|$. So, our approximation can be written as

$$f = \sum_{i=1}^{M}\sum_{j=1}^{n_i} \widehat{f}_{u_{ij}}(z) + \sum_{i=1}^{M} \widetilde{f}_{u_i}(z) + \widetilde{f}_u(z)$$

Thus, the error associated with the approximation is

$$e = \sum_{i=1}^{M} \widetilde{f}_{u_i}(z) + \widetilde{f}_u(z)$$

and its error is bounded by

$$|e| \leq \sum_{i=1}^{M} \left|\widetilde{f}_{u_i}(z)\right| + \left|\widetilde{f}_u(z)\right| \leq M \cdot \left|\frac{\epsilon}{2M}\right| + \left|\frac{\epsilon}{2}\right| = \epsilon.$$

Hence, we have shown the universal approximation capability for a fixed x. ■

As an extension of theorem (13), we can make this theorem more powerful. The theorem, along with its accompanying proof is now provided.

A More Powerful Theorem As was mentioned, there is an extension of theorem (13) by which we can make this theorem more powerful. Below, we show that we can approximate any arbitrary $\mathcal{L}_p$ function such that its mean error falls within an arbitrary ε.

Theorem 14 *Letting f be an arbitrary function in $\mathcal{L}_p$ where $1 \leq p < \infty$. For any $\varepsilon > 0$, we can find a finite number of directions $u_1, \ldots, u_M$ and basis functions per direction ϕ_{ij}, $i = 1, \ldots M$, $j = 1, \ldots n_i$, such that the resulting approximation will estimate f to within ε, that is $\left\| f(\mathbf{x}) - \sum_{i=1}^{M} \sum_{j=1}^{n_i} w_{ij} \phi_{ij}(\mathbf{z}) \right\|_p < \varepsilon$, where the individual terms are as defined in theorem* (13).

Proof. Letting f be an arbitrary function in $\mathcal{L}_p$ because $\mathcal{D}(\Re^n)$, or the space of Frechet functions with compact domains, is dense in $\mathcal{L}_p$ [68], then we can find an $\overline{f}$ in $\mathcal{D}(\Re^n)$ such that $\left\| f(\mathbf{x}) - \overline{f}(\mathbf{x}) \right\|_p < \varepsilon'$ From theorem (13), our approximation $f \approx \sum_{i=1}^{M} \sum_{j=1}^{n_i} w_{ij} \phi_{ij}(\mathbf{z})$ was uniform with respect to the sup *norm*. Thus, integrating over the entire domain yields

$$\left\| \overline{f}(\mathbf{x}) - \widehat{f}(\mathbf{x}) \right\|_p < V(\kappa)\, \varepsilon'$$

where $V(\kappa)$ is the volume and κ is the domain of f which has compact support, $f \in \mathcal{D}(\Re^n)$. Letting R_κ be the radius of the hypersphere enclosed by the domain κ, then the volume of this n-dimensional hypersphere can be calculated by the expression

$$V(\kappa) = \frac{\pi^{n/2} R_\kappa^n}{\Gamma\left(\frac{n}{2}+1\right)}.$$

Thus, utilizing the Schwartz Inequaility, it can be seen that

$$\begin{aligned} \left\| f(\mathbf{x}) - \widehat{f}(\mathbf{x}) \right\|_p &= \left\| \left(f(\mathbf{x}) - \overline{f}(\mathbf{x}) \right) + \left(\overline{f}(\mathbf{x}) - \widehat{f}(\mathbf{x}) \right) \right\|_p \\ &\leq \left\| f(\mathbf{x}) - \overline{f}(\mathbf{x}) \right\|_p + \left\| \overline{f}(\mathbf{x}) - \widehat{f}(\mathbf{x}) \right\|_p \\ &\leq \varepsilon' + \frac{\pi^{n/2} R_\kappa^n}{\Gamma\left(\frac{n}{2}+1\right)} \varepsilon' = \varepsilon \end{aligned}$$

where $\varepsilon = \left(1 + \frac{\pi^{n/2} R_\kappa^n}{\Gamma\left(\frac{n}{2}+1\right)} \right) \varepsilon'$. ■

Remark 15 *The above theorem approximated the function with respect to the $\mathcal{L}_p$ norm, which expands upon the class of functions for which such a universal approximation holds in the literature [54], [47], [46]. But, as was noted in theorem* (13),

the approximation held pointwise for rapidly decreasing functions, $f \in \mathcal{F}_n$. Thus, we see that functions, $f \in \mathcal{F}_n$, can also be approximated pointwise as well as in the $\mathcal{L}_p$ sense.

3.5 Optimization

To motivate the optimization problem, we recall lemma(19), which claimed that we can approximate the response surface to within an arbitrary degree of precision, ϵ, with an appropriately chosen set of M directions. So how does one go about finding these optimal directions? We propose selecting the directions that minimize the nonlinear least-squares cost function (provided below) given a specified set of input-output data:

Optimization Objective: Given a high-dimensional, nonlinear dataset, we are seeking to accurately approximate the underlying response function, $f(x)$ governing the sample space, S.

Specifically, we are seeking the directions, $u_1, ..., u_M$, and the 1-dimensional functions, $\widehat{f}_{u_i}$, along those directions that minimize

$$L = \sum_{k=1}^{N} \left| y_k - \sum_{i=1}^{M} \widehat{f}_{u_i}(z_{ik}) \right|^2$$

where $\widehat{f}_{u_i}(z_{ik}) = \sum_{j=1}^{n_i} w_{ij}\phi_{ij}(z_{ik})$, $z_{ik} = u_i^T x_k$, and w is a vector of weights on the basis functions. Each function, $\widehat{f}_{u_i}$, is an approximation of the underlying response surface along that direction and is formulated by using a suitable set of bases, as described in section (3.3). The grid spacing methodology upon which these bases operate is as outlined in section (3.3.4).

Because we are dealing with nonlinear, high-dimensional datasets, the task of finding the optimal directions, u_i, is difficult. Our approach will be to solve for them

individually, fitting the best single dimensional function along each, and then at the end, put all of these directions together to find the optimal model.

Thus, we solve our estimation problem with a two-stage optimization. The first stage involves searching for the best fit given fixed directions, $u_1, ..., u_M$. This is a standard linear least squares problem to minimize the cost function $\|\Phi W - Y\|^2$, with the solution given by $W_{LS} = \Phi^+ Y$, where Y is the response, $\Phi^+ = \left(\Phi^T \Phi\right)^{-1} \Phi^T$ and Φ is an $N \times \left(\sum_{i=1}^{M} n_i\right)$ matrix defined as $\left[\Phi_{ij}\left(z_{ik}\right)\right]^{ST}$. Note that each $\Phi_{ij}\left(z_{ik}\right)$ is a suitable basis function as described in section (3.3), and the matrix W contains the individual weights for each of those bases. Details for the matrix stacking operation are provided in the appendix. The second level of the optimization is a nonlinear problem that involves finding the optimal projection directions.

$$e_{LS} = \min_{u_1,...,u_m \in \Re, \|u_i\|=1} \left(\Phi W_{LS} - Y\right)^T \left(\Phi W_{LS} - Y\right) \tag{8}$$

The next theorem shows that the two-stage optimization problem can be reduced to the following nonlinear optimization.

Theorem 16 *Assuming there exists a function belonging to the class of universal approximators as stated in theorem* (13)*, the optimization problem for finding the best function estimator associated with seeking the optimal projection pursuit directions can be formulated as maximizing the explanatory power of the estimate, succinctly stated as*

$$c_{ep} = \max_{u_1,...,u_m \in \Re, \|u_i\|=1} Y^T \Phi \Phi^+ Y \tag{9}$$

Proof. The argument of equation (8) can be written as

$$\left(\Phi W_{LS} - Y\right)^T \left(\Phi W_{LS} - Y\right) = \left(\Phi \Phi^+ Y - Y\right)^T \left(\Phi \Phi^+ Y - Y\right)$$

which leads to

$$\left(\Phi \Phi^+ Y - Y\right)^T \left(\Phi \Phi^+ Y - Y\right) = Y^T \left(I - \Phi \Phi^+\right)^T \left(I - \Phi \Phi^+\right) Y$$

by plugging in for the least-squares solution. Noting that $\left(\Phi\Phi^{+}\right)^{T} = \left(\Phi\Phi^{+}\right)$, we have

$$Y^{T}\left(I-\Phi\Phi^{+}\right)^{T}\left(I-\Phi\Phi^{+}\right)Y = Y^{T}\left(I-\Phi\Phi^{+}-\Phi\Phi^{+}+\Phi\Phi^{+}\Phi\Phi^{+}\right)Y$$

Given that $\Phi^{+}\Phi = I$, this leaves us with

$$Y^{T}\left(I-\Phi\Phi^{+}-\Phi\Phi^{+}+\Phi\Phi^{+}\Phi\Phi^{+}\right)Y = Y^{T}\left(I-\Phi\Phi^{+}\right)Y$$

Thus, the optimization problem of eq. (8) reduces to

$$c_{ep} = \max_{u_1,\ldots,u_m\in\Re,\|u_i\|=1} Y^{T}\Phi\Phi^{+}Y$$

given that $Y^{T}Y$ is constant. ■

Corollary 17 *If the bases, ϕ, that form the basis function matrix Φ are orthogonal, then $\Phi^{+} = \left(\Phi^{T}\Phi\right)^{-1}\Phi^{T} = \Phi^{T}$, and the optimization problem reduces to maximizing the 2-norm of the coefficients:*

$$c_{ep} = \max_{u_1,\ldots,u_m\in\Re,\|u_i\|=1} Y^{T}\Phi\Phi^{T}Y = \left\|\Phi^{T}Y\right\|^{2}$$

where again, Φ is our matrix of basis functions along the given M directions and Y is the response vector.

Corollary 18 *If the basis function matrix Φ is one-dimensional then*

$$c_{ep} = \max_{u}\left\|\overline{\Phi}^{T}Y\right\|^{2} \tag{10}$$

where $\overline{\Phi} = \frac{\Phi}{\|\Phi\|}$.

Proof. Noting that $\Phi^{+} = \frac{\Phi^{T}}{\Phi\Phi^{T}}$ in the 1-dimensional case, then it follows from equation (9) that

$$c_{ep} = \max_{u}\frac{\left\|\Phi^{T}Y\right\|^{2}}{\|\Phi\Phi^{T}\|}$$

which can be rewritten as equation (10). ■

The algorithm used to implement this modeling technique transforms the raw input data into a matrix of projection directions and assigns the grid spacing along these directions. One-dimensional local basis function fits along these projections are then modeled simultaneously. The performance of this model is evaluated along with its Jacobian and Hessian. The algorithm is solved iteratively to find the optimal projection direction matrix, R, by adjusting it via the following Levenberg-Marquardt optimization: $R_{k+1} = R_k - (H + \mu I)^{-1} J^T e$. It is not particularly straight-forward to arrive at the specific expressions used in this optimization, but rather requires extensive calculations. Thus, the derivations of the Jacobian and Hessian formulations are provided at the end of this chapter, in section (3.8.2). The interested reader is encouraged to explore these derivations, however, the section could be skipped without loss of the general flow of the material.

3.6 PPLM Design

3.6.1 Network Structure

A schematic of the PPLM architecture is provided in figure (14).

As illustrated, the inputs, x_D, are to be reconstructed into a series of projection directions, g_j, that are nonlinearly activated and then combined to form the projected output(s), y_q, of the model. A detailed sketch of the algorithm used is provided in the section that follows:

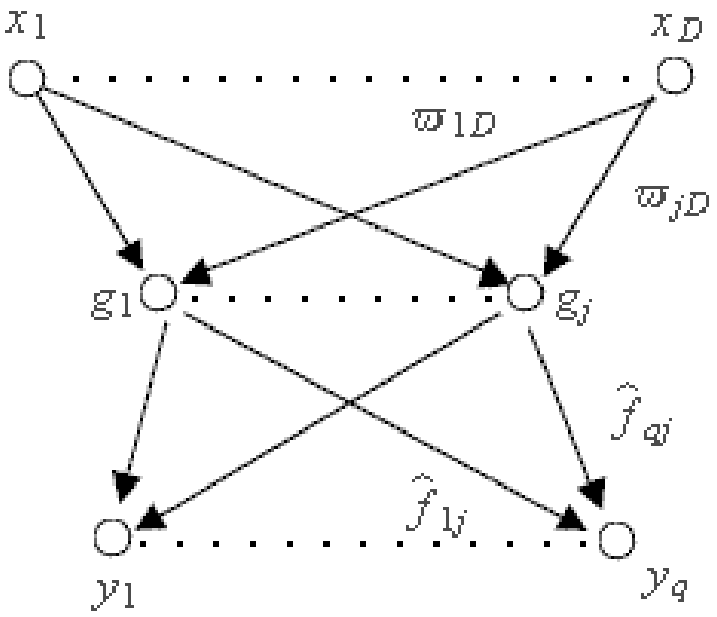

Figure 14: Schematic of PPLM network structure

3.6.2 Algorithm

Note: m = total number of projection directions

$\mathbf{R}$ = projection matrix

d = grid point subintervals per direction

The performance criterion is set to the MSE of the validation sample.

1. Initialization

    a. Set the number of projection directions, m, and loop through

        while performance criterion improves, $m = m + 1$

    b. Initialize R randomly

    c. For $d = d_{\min}$ to $d_{\max}$ (good values of d might range from 3 to 10)

    d. Set the stopping criteria for the outer loop

    e. Set parameters for adaptive μ optimization algorithm, loop through while criteria are valid

        i. Set μ_{inc} and μ_{dec} (increments and decrements of 10 are good values)

        ii. Set $\mu_{\max}$

iii. Set $iter_{\max}$

iv. Set e_{goal}

2. While $(\mu < \mu_{\max})$ & $(iter < iter_{\max})$ & $(e > e_{goal})$

3. Determine grid space values

a. $\mathbf{Z} = \mathbf{X} * \mathbf{R}$ (setting transformed inputs)

b. For each transformed input vector, $\mathbf{Z}_k$, create a vector of Order statistics for the observations, $\mathbf{O}_k$, $\forall$ $\mathbf{Z}_k$ and set the nodes for that input

4. Next, the basis functions can be set

a. For each projection direction, k, compute the distance of each data point to each node (dropping subscript, k, for clarity):

i. $\overline{z}_{ij,l} = \frac{z_{ij} - \eta_j}{b_{j,l}}$

ii. $\overline{z}_{ij,r} = \frac{z_{ij} - \eta_j}{b_{j,r}}$

b. Create local piecewise cubic basis functions for each projection (Note: dropping observation number subscript, i, to avoid confusion):

i. For each projection direction, compute the first portion of the basis function (to be designated as $\{\phi\}^1$), accounting for potential unequal base widths along dimensions:

$$\{\phi_j\}^1 = \left\{ \begin{array}{ccc} (|\overline{z}_{j,l}| - 1)^2 (2 \cdot |\overline{z}_{j,l}| + 1) & & \text{if } -1 < \overline{z}_{j,l} \leq 0 \\ (|\overline{z}_{j,r}| - 1)^2 (2 \cdot |\overline{z}_{j,r}| + 1) & , & \text{if } 0 < \overline{z}_{j,r} \leq 1 \\ 0 & & \text{otherwise} \end{array} \right\}$$

ii. For each projection direction, compute the derivative portion of the basis function (to be designated as $\{\phi\}^2$):

$$\{\phi_j\}^2 = \left\{ \begin{array}{ccc} (|\overline{z}_{j,l}| - 1)^2 \cdot \overline{z}_{j,l} & & \text{if } -1 < \overline{z}_{j,l} \leq 0 \\ (|\overline{z}_{j,r}| - 1)^2 \cdot \overline{z}_{j,r} & , & \text{if } 0 < \overline{z}_{j,r} \leq 1 \\ 0 & & \text{otherwise} \end{array} \right\}$$

iii. Assign basis function for each projection direction:

$$\phi = \left[\begin{array}{cc} \{\phi\}^1 & \{\phi\}^2 \end{array} \right]$$

iv. Append to full basis function matrix: $\Phi = \begin{bmatrix} \Phi & \phi \end{bmatrix}$

v. Fit using least squares to find the weighting matrix, W: $\Phi W = Y$, where Y is the response vector

(a). $\Phi^+ = \left(\Phi^T\Phi + \gamma I\right)^{-1}\Phi^T$, where γ is some small, positive constant near zero

(b). $W = \Phi^+ Y$

(c). Compute the error: $E = \Phi W - Y$

5. Optimization: Compute Jacobian and Hessian as per the Optimization section, equation (17) for the Jacobian and equations (19) and (20) for the Hessian.

a. $R_1 = R - (H + \mu I)^{-1} J^T e$

b. $\mathbf{Z}_1 = \mathbf{X} \cdot R_1$

c. Repeat steps 3 & 4 with this new R_1 and $\mathbf{Z}_1$ matrix to set the grid space, assign the basis functions, and fit the appropriate weights

6. Evaluation

a. Compare E_1^2 with E_0^2

b. If $E_1^2 < E_0^2$, then $R_0 = R_1$, $\mathbf{Z_0} = \mathbf{Z}_1$, $E_0 = E_1$

c. Update μ if necessary

7. Return to step 2, repeating steps 3 through 6 provided the criteria in the while loop of step 2 still holds

3.7 Experimental Results

As a further test of the method's effectiveness, the model's predictive prowess was tested on data from a bearing defect experiment [94]. Details for this test, including the experimental setup and an explanation of the sensor data are provided below, followed by a comparison of the predictive results. Note that for this test, a comparison of prediction accuracy of the PPLM model along with that of a feedforward neural network approach is provided.

3.7.1 Background

The failure of rolling element bearings is one of the primary causes of breakdown in rotating machinery. In certain applications, this failure can produce catastrophic consequences. Unexpected machine breakdown can often lead to high maintenance costs and lengthy downtime. It is important to monitor and diagnose bearing condition online, because detecting bearing defects early can lead to optimal maintenance scheduling. Details for the experiment, as run by Georgia Tech research fellow, Scott Billington, are now provided.

3.7.1.1 Experimental Setup

A Timken LM50130 cup (outer race) and LM501349 cone (inner race) bearing is used for this experiment. All bearing defects are artificially inscribed axially in the center of the outer race with a diamond scribe. The size of the damage is controlled by the pressure and number of passes of the scribe. A Form Talysurf Profilometer was used to measure both the width and the height of the defects on the outer race. A table of the widths and heights of the bearing defects is provided in Appendix (A.2). It should be noted that the majority of these defect areas are well below bearing failure industry standards. Such industry standard has defined a bearing defect to be one that has a total area of at least $6.25mm^2$. So, a predictive model capable of identifying defects at such an early stage as the defects presented in this experiment

would provide an early warning system that would be useful in online monitoring and diagnosis.

A Triaxial Kistler 8792A50 high frequency accelerometer and a Physical Acoustics Corporation acoustic emission sensor R15 were mounted on the housing directly above the defective bearing. All signals were sampled at 50kXHz with $(2^{18} + 10)$ scans per file. The data acquisition system utilizes a National Instruments DAQ-1200 PCMCIA data acquisition card and a Pentium computer. Experiments are performed at different cyclic speeds and radial loads for each defect. These different input conditions are provided in Appendix (A.2).

3.7.1.2 Signal Processing

Bearing defect signals must be extracted and isolated from a variety of noise that is present in a real-world operating environment. From these extracted signals, the signal features can be processed. The predictive models used for this analysis are trained to predict the different levels of bearing defect using such signal features as inputs.

The output signals from the accelerometer and acoustic emission were digitized from analog voltage signals. Noise cancellation was employed by HFRT, or high frequency resonance technique. This amplitude demodulation makes use of modulated high frequency vibration signatures of the defect frequency in a series of three steps: bandpass filtering, signal rectification, and low-pass filtering. Spectrum analysis can then be conducted on the resulting signal.

The signal features used in this work are RMS, Kurtosis, Crest factor, Max_FFT, Peak Value of the amplitude spectrum of the HFRT signal, Peak Value of the Cepstrum analysis of the accelerometer and RMS, and the 1^{st} FFT Peak Value of Acoustic Emission. RMS is the root-mean-square of the signal,

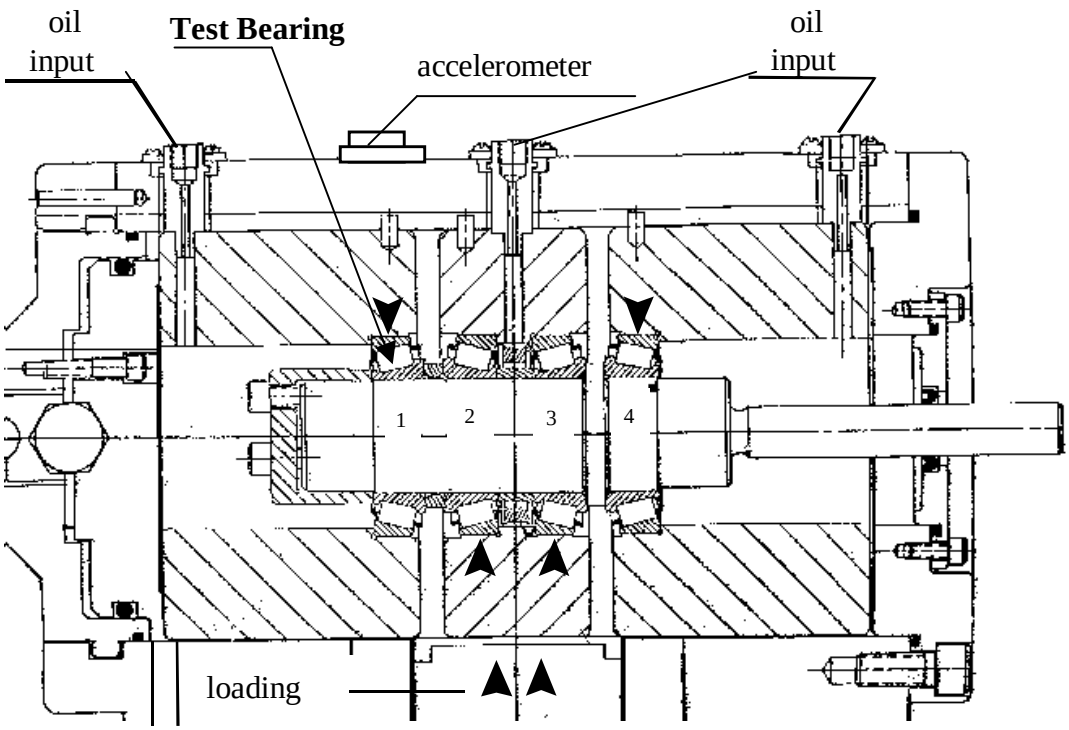

Figure 15: Schematic of Test Housing

$$RMS = \sqrt{\frac{\sum_{i=1}^{N} X_i^2}{N}}$$

where N is the total number of sampling points and X_i is the signal at each sampling point, i. The kurtosis of a signal is a measure of the degree to which the data are peaked or flat relative to a normal distribution.

$$Kurtosis = \frac{\sqrt{\frac{\sum_{i=1}^{N} X_i^4}{N}}}{RMS^4}$$

The Crest Factor is equal to the peak amplitude of a waveform divided by its RMS value.

$$Crest\ Factor = \frac{\max(abs(X))}{RMS}$$

The purpose of the Crest Factor is to provide an indication for how much impacting is occurring in a waveform. Peak value is the maximum value of the amplitude spectrum at a particular defect frequency. Max_FFT is the maximum amplitude of the FFT (Fast Fourier Transform) of a time-domain signal.

$$x(t) = rawdata$$
$$x(t) = bandpass(x(t))$$
$$X(f) = FFT(x(t))$$
$$Max\_FFT = \max(abs(X(f)))$$

1^{st} FFT peak value is the first peak value reading from the FFT signal. Peak Value of the amplitude spectrum of the HFRT signal is constructed as follows:

$$x(t) = rawdata$$
$$x(t) = bandpass(x(t))$$
$$x(t) = abs(x(t))$$
$$x(t) = lowpass(x(t))$$
$$X(f) = FFT(x(t))$$
$$Peak\_envelope = \max(X(f))$$

Cepstrum analysis treats the spectrum as if it were a waveform:

$$Cepstrum = \frac{FFT\left(\log_{10}\left[\left(\frac{FFT(x(t))}{N}\right)\left(conjugate\left(\frac{FFT(x(t))}{N}\right)\right)\right]\right)}{N}$$

The usefulness of cepstrum is as a pattern recognition scheme that is sensitive to patterns of sidebands and harmonics. The full listing of signal feature inputs is provided in Appendix (A.3).

3.7.2 Modeling Procedures

For the series of bearing defect experiments, signal features are provided for different levels of defect attribute versus varying load levels and speeds. A total of 143 observations were investigated. The input data were a set of 29 different signal features based on accelerometer and acoustic emissions data, as described previously. For all of the experiments run on this data, 75% of the data was used for training, while 25% was randomly chosen as the cross-validation sample. As each of the models have a number of varying parameters to be set, the use of this validation sample was devised as a data-driven selection method for choosing the appropriate parameters. Each time the experiment was run, the training and cross-validation samples are held constant for all of the methods tested in order to give a fair comparison of the results. In order to maximize the use of testing data, an $n-1$ set produced for each observation. Thus, for each one of the 143 observations, that one data point was removed from the sample space. The remaining 142 observations were then split into a 75% and a 25% randomly-chosen training and validation sample. The best neural network and the best PPLM models were thus chosen and then applied to the holdout (test) observation to develop a predicted defect size. This process was repeated for each of the 143 observations. The results displayed are cumulative (average across all observations) or, in some cases, have been split up into cumulative within groups (average across all defects of a certain width). In this manner, we can glean an

accurate reflection of the prediction capabilities of the compared modeling methods.

3.7.3 Experiment Set #1

For this experiment, the defect width was chosen as the response variable. The Levenberg-Marquardt implementation of the projection pursuit network was compared on the cross-validation sample with the best feedforward artificial neural network (ANN) model. The neural net which performed the best on the cross-validation sample is presented in the graph, figure (16). As a comparison of the results, we will illustrate the differences in performances by simply comparing results across defect grouping. The defect groupings were defined based on the experimental data provided. The data consisted of 8 distinct levels of defect, or in this case defect width. Thus, performance across each of those groupings is what is illustrated in the graph. The true value of the defect width for each group is labeled actual in the chart, whilst the other two bars per grouping are average prediction results along each set of test conditions run for that particular defect group.

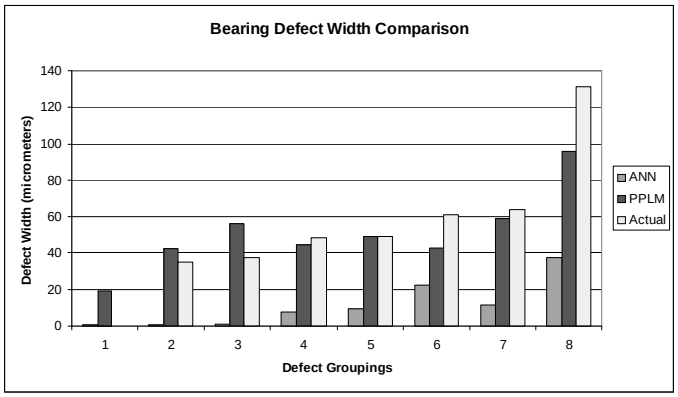

Figure 16: Bearing Defect Width: Predictions vs. Actuals

An error comparison is provided below. For this comparison, errors were grouped by defect grouping and calculated for the ith group as follows:

$$error_i = \frac{abs\left(E\left[\widehat{y}_i\right] - E\left[y_i\right]\right)}{E\left[y\right]}$$

where $E[\widehat{y}_i]$ is the mean value of the prediction within defect group i, $E[y_i]$ is the expected value of the actual defect size within the group, and $E[y]$ is the overall average defect size across the entire sample population. Note that we are computing error per grouping in this manner so as not to weigh more heavily the lower groupings (those with small defects) and similarly, so as not to produce a division-by-zero error for group 1.

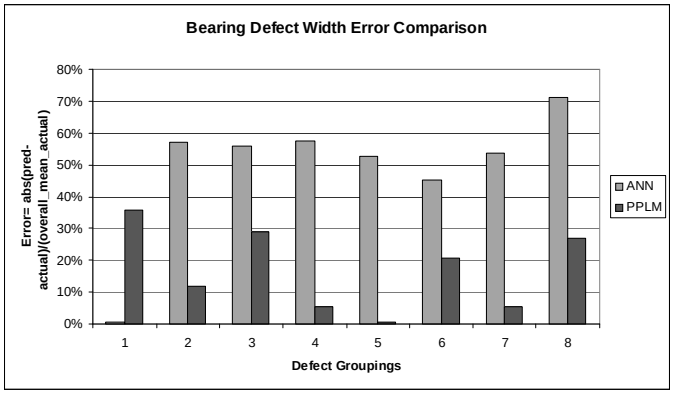

Figure 17: % Errors of Bearing Defect Width Predictions

Computing overall percent error in a more standard manner yields the following:

Table 1: Overall defect width percentage error for Experiment 1

Model	**% Error**
ANN	32.1%
PPLM	16.5%

In this table, the percentage error is calculated as

$$\% \; error = \frac{abs\left(E\left[\widehat{y}\right] - E\left[y\right]\right)}{E\left[y\right]}$$

3.7.4 Experiment Set #2

Using the same data, another potential response variable is available: defect height. Thus, the same set of analysis was run on the data, but this time making a prediction

of the defect height. Results are shown in figure (18) grouped by the 8 discrete levels of this new response. An error comparison is provided in figure (19). Again, for this comparison, errors were grouped by defect grouping and calculated for the ith group as follows:

$$error_i = \frac{abs\left(E\left[\widehat{y}_i\right] - E\left[y_i\right]\right)}{E\left[y\right]}$$

where the terms are as defined previously. Similarly, taking the overall percentage error, we see that the PPLM approach performs quite well.

Table 2: Overall defect height percentage error for Experiment 1

Model	**% Error**
ANN	3.1%
PPLM	1.7%

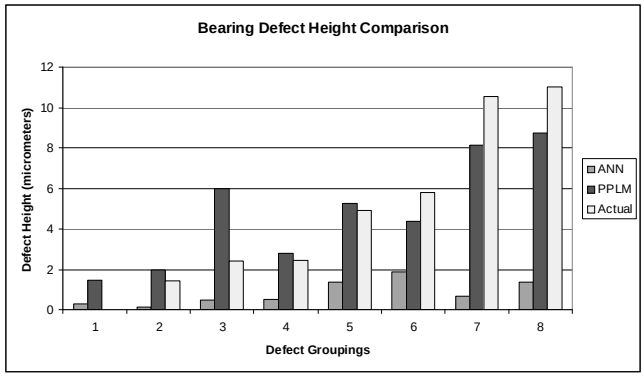

Figure 18: Bearing Defect Height: Predictions vs. Actuals

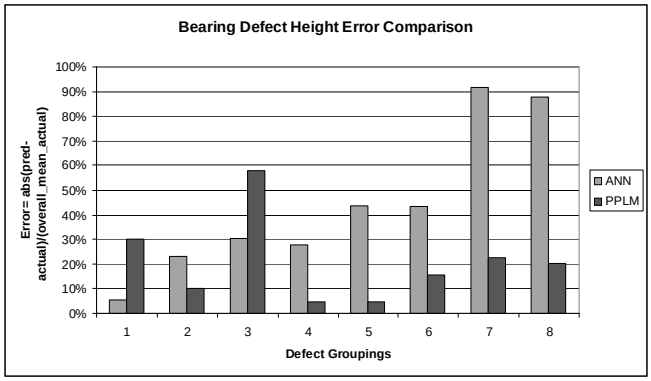

Figure 19: % Errors of Bearing Defect Height Predictions

3.8 Additional Theory and Derivations for Chapter 3

Because of the extensive mathematical derivations, the casual reader is invited to skip the next subsection without loss of the general flow of the material. The next section contains the bulk of the proofs for the universal approximation capabilities of the method. In the interest of readability, the derivations of the Error Jacobian and the Hessian for use in the Levenberg-Marquardt optimization technique have also been moved to this final section of Chapter 3. They are included after the universal approximation theory details. Again, while the reader is encouraged to explore this section, the casual reader may skip directly to Chapter 4 without loss of continuity.

3.8.1 Universal Approximation Theory and Derivation Details

Let us begin with the derivation of a key lemma.

Lemma 19 *For any arbitrary function $f \in \mathcal{F}_n$, with $\epsilon > 0$, there exists an integer M set of directions $u_1, ..., u_{M,}$ where $M > 0$, and a set of continuous functions $\widehat{f}_{u_k} : \Re \rightarrow \Re$, such that*

$$\left| \int_{\mathcal{U}} f_u\left(x^T u\right) dS - \sum_{k=1}^{M} \widehat{f}_{u_k}\left(x^T u_k\right) \right| < \epsilon$$

uniformly in $\mathcal{X} \subset \Re^n$ for all $x \in \mathcal{X}$.

Proof of Lemma 19 The function $f_u(x)$, $f_u(x) : \mathcal{S}^{m-1} \times \mathcal{X} \rightarrow \Re$, is continuous on the compact space in $\mathcal{S}^{m-1} \times \mathcal{X}$, and thus can be considered to be uniformly continuous on $\mathcal{S}^{m-1} \times \mathcal{X}$. For any $\varepsilon' > 0$, there exists a $\delta > 0$ such that if $y \in B_\delta(x)$ then $\left| f_u\left(x^T u\right) - f_u\left(y^T u\right) \right| < \varepsilon'$ for any $x \in \mathcal{X}$ and $u \in \mathcal{S}^{m-1}$, and where $B_\delta(x)$ is a ball of radius δ centered at x.

Since $\underset{x \in \mathcal{X}}{\cup} B_\delta(x)$ is an open cover for $\mathcal{X}$ and due to the compactness of $\mathcal{X}$, there exists a finite subcover. Let $x_1, x_2, ..., x_N$ be the centers of the corresponding open balls,

$B_\delta(x)$, that cover space $\mathcal{S}^{m-1}$. For each x_k, we wish to approximate $\int\limits_{\mathcal{S}^{m-1}} f_u\left(x_k^T u\right) dS$ numerically. Fortunately, the integral over the surface of an m-dimensional hypersphere on a compact set may be approximated within an arbitrary degree of precision by a finite sum [19] and [75]. For each x_k, $k = 1, 2, ..., N$, there exists an $m_k > 0$, a set of weights $\psi_{1,}....\psi_{m_k}$, and a set of directions $u_1, ...u_{m_k}$ such that

$$\left| \int_{\mathcal{U}} f_u\left(x_k^T u\right) dS - \sum_{j=1}^{m_k} \psi_{j_k} f_{u_{j_k}}\left(x_k^T u_{j_k}\right) \right| < \varepsilon' \tag{11}$$

for $k = 1, 2, ..., N$ [19], [75].

Now define

$$\widehat{F}(x) = \frac{\sum\limits_{k=1}^{N} \left(\alpha_k \sum\limits_{j=1}^{m_k} \psi_{j_k} f_{u_{j_k}}\left(x_k^T u_{j_k}\right) \right)}{\sum\limits_{k=1}^{N} \alpha_k}$$

where $\alpha_k = 1 - \min\left(\frac{|x - x_k|}{\delta}, 1\right)$, thus $0 \leq \alpha_k \leq 1$ for each k.

$$\widehat{F}(x) = \sum_{k=1}^{N} \sum_{j=1}^{m_k} \left[\frac{\alpha_k \psi_{j_k} f_{u_{j_k}}\left(x_k^T u_{j_k}\right)}{\sum_{k=1}^{N} \alpha_k} \right] = \sum_{i=1}^{N\prime} \widehat{f}_{u_i}(z_i)$$

where $z_i = x_k^T u_i$, $\widehat{f}_{u_i} = \dfrac{\alpha_k \psi_{j_k} f_{u_{j_k}}\left(x_k^T u_{j_k}\right)}{\sum_{k=1}^{N} \alpha_k(x_k)}$,

$$j_1 + j_2 + ... + j_k < i < j_1 + j_2 + ... + j_{k+1}$$

$0 < k < N$, $M = \sum\limits_{k=1}^{N} j_k$, $1 \leq i \leq M$.

Using equation (11) and letting $E_r = \left| \int_{\mathcal{U}} f_u\left(x_k^T u\right) dS - \sum_{i=1}^{N\prime} \widehat{f}_{u_i}\left(z_i\right) \right|$ we find that

$$\begin{aligned} E_r &\equiv \left| \frac{\sum_{k=1}^{N} \alpha_k(x) \int_{\mathcal{U}} f_u\left(x^T u\right) dS - \sum_{k=1}^{N} \alpha_k(x) \sum_{j=1}^{m_k} \psi_{j_k} f_{u_{j_k}}\left(x_k^T u_{j_k}\right)}{\sum_{k=1}^{N} \alpha_k(x)} \right| \\ &= \left| \left[\frac{\sum_{k=1}^{N} \alpha_k}{\sum_{k=1}^{N} \alpha_k} \right] \left[\int_{\mathcal{U}} f_u\left(x_k^T u\right) dS - \sum_{j=1}^{m_k} \psi_{j_k} f_{u_{j_k}}\left(x_k^T u_{j_k}\right) \right] \right| \\ &= \left| \int_{\mathcal{U}} f_u\left(x_k^T u\right) dS - \sum_{j=1}^{m_k} \psi_{j_k} f_{u_{j_k}}\left(x_k^T u_{j_k}\right) \right| < \varepsilon' \end{aligned}$$

and thus we have that

$$\left| \int_{\mathcal{U}} f_u\left(x_k^T u\right) dS - \sum_{i=1}^{N\prime} \widehat{f}_{u_i}\left(z_i\right) \right| < \varepsilon' \tag{12}$$

Let $x \in \mathcal{X}$ be arbitrary. Since $\bigcup_{k=1}^{N} B_\delta(x_k)$ covers $\mathcal{X}$, there exists k such that $x \in B_\delta(x_k)$. Then, $\left[\sum_{i=1}^{M} \widehat{f}_{u_i}\left(x_k^T u_i\right) - \sum_{i=1}^{M} \widehat{f}_{u_i}\left(x^T u_i\right) \right] \leq \varsigma$

where $\varsigma = \sum_{k=1}^{N} \sum_{j=1}^{m_k} \left[\frac{\alpha_k \psi_{j_k} \left| f_{u_{j_k}}\left(x_k^T u_{j_k}\right) - f_{u_{j_k}}\left(x^T u\right) \right|}{\sum_{k=1}^{N} \alpha_k} \right]$.

It follows then that

$$\sum_{k=1}^{N} \sum_{j=1}^{m_k} \left[\frac{\alpha_k \psi_{j_k} \left| f_{u_{j_k}}\left(x_k^T u_{j_k}\right) - f_{u_{j_k}}\left(x^T u\right) \right|}{\sum_{k=1}^{N} \alpha_k} \right] \leq \frac{\varepsilon' \sum_{k=1}^{N} \alpha_k \sum_{j_k=1}^{m_k} \left| \psi_{j_k} \right|}{\sum_{k=1}^{N} \alpha_k} \leq c\varepsilon'$$

where $c_k = \sum_{j_k=1}^{m_k} \left| \psi_{j_k} \right|$ and $c = \max_k \; c_k$.

Then, recalling equation (12) and since

$$\left| \int_{\mathcal{U}} f_u\left(x_k^T u\right) dS - \int_{\mathcal{U}} f_u\left(x^T u\right) dS \right| \leq \left| \int_{\mathcal{U}} \left| f_u\left(x_k^T u\right) dS - f_u\left(x^T u\right) \right| dS \right| \leq \varepsilon' S,$$

we can write:

$$\left| \int_{\mathcal{U}} f_u\left(x^T u\right) dS - \sum_{i=1}^{M} \widehat{f}_{u_i}\left(x^T u_i\right) \right| \leq \left| \int_{\mathcal{U}} f_u\left(x^T u\right) dS - \int_{\mathcal{U}} f_u\left(x_k^T u\right) dS \right| + \xi$$

where

$$\xi = \left| \int_{\mathcal{U}} f_u\left(x_k^T u\right) dS - \sum_{i=1}^{M} \widehat{f}_{u_i}\left(x_k^T u_i\right) \right| + \left| \sum_{i=1}^{M} \widehat{f}_{u_i}\left(x_k^T u_i\right) - \sum_{i=1}^{M} \widehat{f}_{u_i}\left(x^T u_i\right) \right|.$$

It can then be noted that $\xi \leq (1 + S + c)\,\varepsilon'$ Now, by letting $\varepsilon' = \frac{\epsilon}{1+S+c}$, the theorem is proven:

$$\left| \int_{\mathcal{U}} f_u\left(x^T u\right) dS - \sum_{i=1}^{M} \widehat{f}_{u_i}\left(x^T u_i\right) \right| < \epsilon,$$

and thus the universal approximation capability holds across x.

3.8.2 Additional Optimization Theoretical Derivations

To implement a Levenberg-Marquardt optimization of the PPLM approach, we must first derive the Error Jacobian and Hessian specific to our optimization problem. These derivations are painstakingly detailed in the pages that follow.

3.8.2.1 Derivation of the Error Jacobian: Case where $\mu = 0$:

Defining Y_s to be the response variable and letting g be the weighting matrix, $g \equiv W$. Let

$$f = Y_s^T g \left(g^T g\right)^{-1} g^T Y_s.$$

Then the Error Jacobian can be written as $d\left(E^T E\right) = -df$ since $E^T E = Y_s^T Y_s - f$. Letting $z = g^T Y_s$, we are left with $f = z^T \left(g^T g\right)^{-1} z$. And letting Δf and $\Delta\left(g^T g\right)$ each denote a small perturbation in f and $\left(g^T g\right)$, respectively, we arrive at the following expression, neglecting for higher-order terms:

$$\Delta f = z^T \Delta \left(g^T g\right)^{-1} z + \Delta z^T \left(g^T g\right)^{-1} z + z^T \left(g^T g\right)^{-1} \Delta z.$$

And since each of these collections of terms is a scalar, it follows that:

$$\Delta f = z^T \Delta \left(g^T g\right)^{-1} z + 2 z^T \left(g^T g\right)^{-1} \Delta z. \tag{13}$$

But, we can now note that

$$\Delta \left(g^T g\right)^{-1} = \left[g^T g + \Delta\left(g^T g\right)\right]^{-1} - \left(g^T g\right)^{-1}.$$

Using this, we can write

$$\left[g^T g + \Delta\left(g^T g\right)\right]^{-1} = \left(g^T g\right)^{-1} + \left(g^T g\right)^{-1} \Delta\left(g^T g\right)\left(g^T g\right)^{-1} + O\left(\left\|\Delta\left(g^T g\right)\right\|^2\right).$$

Thus, it follows that

$$\Delta \left(g^T g\right)^{-1} = -\left(g^T g\right)^{-1} \Delta\left(g^T g\right)\left(g^T g\right) + O\left(\left\|\Delta\left(g^T g\right)\right\|^2\right).$$

And then equation (13) can be written (dropping higher-order terms) as:

$$\Delta f = -z^T \left(g^T g\right)^{-1} \Delta \left(g^T g\right) \left(g^T g\right)^{-1} z + 2z^T \left(g^T g\right)^{-1} \Delta z. \tag{14}$$

Now, letting Δg denote a perturbation in g, so that,

$$\begin{aligned} \Delta \left(g^T g\right) &= \left(g^T + \Delta g^T\right)(g + \Delta g) - g^T g \\ &= g^T g + g^T \Delta g + \Delta g^T g + \Delta g^T \Delta g - g^T g. \end{aligned}$$

We arrive at

$$\Delta \left(g^T g\right) = g^T \Delta g + \Delta g^T g + O\left(\|\Delta g\|^2\right).$$

Plugging this into equation (14) and dropping higher-order terms yields

$$\begin{aligned} \Delta f \;\; = \;\; & -z^T \left(g^T g\right)^{-1} g^T \Delta g \left(g^T g\right)^{-1} z & (15) \\ & -z^T \left(g^T g\right)^{-1} \Delta g^T g \left(g^T g\right)^{-1} z + 2z^T \left(g^T g\right)^{-1} \Delta z. & (16) \end{aligned}$$

Therefore, it follows due to the symmetry of the matrix $\left(g^T g\right)$, as

$$\left(g^T g\right)^{-T} = \left(g^T g\right)^{-1}$$

that we arrive at the following reduction of equation (15):

$$\Delta f = -Y_d^T g^T \Delta g Y_d - Y_d^T \Delta g^T g Y_d + 2z^T \left(g^T g\right)^{-1} \Delta z.$$

Again, using the property that the collections of these terms are scalars

$$\begin{aligned} \Delta f &= -2Y_d^T g^T \Delta g Y_d + 2\Delta z^T \left(g^T g\right)^{-1} z \\ &= -2Y_d^T g^T \Delta g Y_d + 2Y_s^T \Delta g Y_d \\ &= 2\left(Y_s - g Y_d\right)^T \Delta g Y_d. \end{aligned}$$

Thus,

$$\{\Delta f\}^s = 2\left(Y_d^T \otimes E^T\right)\{\Delta g\}^s.$$

where the specified matrices have been stacked. Note that the Kronecker product is defined in the Appendix $(A.1)$. And so as $\Delta g \to 0$, we have

$$Df = 2\left(Y_d^T \otimes E^T\right)(Dg).$$

However, in our case, we add the constant μ to reduce the chance of singularity. The derivation of the Jacobian for this is now provided in the subsection that follows.

3.8.2.2 Derivation of the Error Jacobian: Case where $\mu \neq 0$:

Defining Y_s to be the response variable, assigning $Y_d = g^+ Y_s$ to be the basis functions, and letting g be the weighting matrix, $g \equiv W$, our error vector may be written as: $E = Y_s - gY_d = Y_s - gg^+ Y_s = \left[I - g\left(g^T g + \mu I\right)^{-1} g^T\right] Y_s$. Note that we have added the constant μ to the least-squares computation to reduce the chance of singularity. So,

$$E^T E = Y_s^T \left[I - g\left(g^T g + \mu I\right)^{-1} g^T\right]^T \left[I - g\left(g^T g + \mu I\right)^{-1} g^T\right] Y_s.$$

Rearranging terms, we find that

$$E^T E = Y_s^T \left[I - g\left(g^T g + \mu I\right)^{-T} g^T\right] \left[I - g\left(g^T g + \mu I\right)^{-1} g^T\right] Y_s.$$

Expanding yields the following expression:

$$E^T E = \begin{array}{c} Y_s^T \left[I - g\left(g^T g + \mu I\right)^{-T} g^T - g\left(g^T g + \mu I\right)^{-1} g^T\right] Y_s \\ + Y_s^T \left[g\left(g^T g + \mu I\right)^{-T} g^T g \left(g^T g + \mu I\right)^{-1} g^T\right] Y_s. \end{array}$$

Now add

$$\psi = Y_s^T g\left(g^T g + \mu I\right)^{-T} \mu I \left(g^T g + \mu I\right)^{-1} g^T Y_s^T$$

to both sides of the equation, and since $\left(g^T g + \mu I\right)\left(g^T g + \mu I\right)^{-1} = I$, then we are left with:

$$E^T E + \psi = Y_s^T \left[I - g\left(g^T g + \mu I\right)^{-T} g^T - g\left(g^T g + \mu I\right)^{-1} g^T + g\left(g^T g + \mu I\right)^{-T} g^T\right] Y_s.$$

Canceling out another set of terms, leaves us with

$$E^T E + \psi = Y_s^T \left[I - g\left(g^T g + \mu I\right)^{-1} g^T\right] Y_s$$

or, it may be restated as

$$E^T E = Y_s^T \left[I - g\left(g^T g + \mu I\right)^{-1} g^T\right] Y_s - Y_s^T g\left(g^T g + \mu I\right)^{-T} \mu I \left(g^T g + \mu I\right)^{-1} g^T Y_s^T.$$

Recalling that $Y_d = \left(\mu I + g^T g\right)^{-1} g^T Y_s$, yields:

$$E^T E = Y_s^T \left[I - g \left(g^T g + \mu I \right)^{-1} g^T \right] Y_s - Y_d^T \mu I Y_d = Y_s^T \left[I - g g^+ \right] Y_s - \mu Y_d^T Y_d$$

where $g^+ \equiv \left(g^T g + \mu I \right)^{-1} g^T$. Defining

$$f = - \left[Y_s^T Y_s - z^T \left(g^T g + \mu I \right)^{-1} z \right]$$

with $z = g^T Y_s$, then

$$E^T E = - \left(f + \mu Y_d^T Y_d \right)$$

and

$$D(E^T E) = -Df - \mu D(Y_d^T Y_d).$$

Therefore the computation of the Jacobian of $E^T E$ can be separated into computation of Df and $D(Y_d^T Y_d)$. So, dropping higher-order terms:

$$\Delta f = \Delta z^T \left(g^T g + \mu I \right)^{-1} z + z^T \Delta \left(g^T g + \mu I \right)^{-1} z + z^T \left(g^T g + \mu I \right)^{-1} \Delta z.$$

Collecting algebraic expressions, we can rewrite as

$$\Delta f = z^T \Delta \left(g^T g + \mu I \right)^{-1} z + 2 z^T \left(g^T g + \mu I \right)^{-1} \Delta z.$$

Noting that

$$\Delta \left(g^T g + \mu I \right)^{-1} = \left[g^T g + \Delta \left(g^T g \right) + \mu I \right]^{-1} - \left(g^T g + \mu I \right)^{-1}$$

And that

$$\begin{aligned} \left[g^T g + \Delta \left(g^T g \right) + \mu I \right]^{-1} &= \left(g^T g + \mu I \right)^{-1} - \left(g^T g + \mu I \right)^{-1} \Delta \left(g^T g \right) \left(g^T g + \mu I \right)^{-1} \\ &\quad + O \left(\left\| \Delta \left(g^T g \right)^2 \right\| \right). \end{aligned}$$

Then, dropping higher-order terms yields:

$$\Delta \left(g^T g + \mu I \right)^{-1} = - \left(g^T g + \mu I \right)^{-1} \Delta \left(g^T g \right) \left(g^T g + \mu I \right)^{-1}$$

And we are left with

$$\Delta f = 2z^T \left(g^T g + \mu I\right)^{-1} \Delta z - z^T \left(g^T g + \mu I\right)^{-1} \Delta \left(g^T g\right) \left(g^T g + \mu I\right)^{-1} z.$$

Noting that

$$\Delta \left(g^T g\right) = \left(g^T + \Delta g^T\right)(g + \Delta g) - g^T g = g^T g + g^T \Delta g + \Delta g^T g + \Delta g^T \Delta g - g^T g,$$

Dropping higher order terms, yields

$$\Delta \left(g^T g\right) = g^T \Delta g + \Delta g^T g.$$

So,

$$\begin{aligned} \Delta f &= 2z^T \left(g^T g + \mu I\right)^{-1} \Delta z - z^T \left(g^T g + \mu I\right)^{-1} \Delta g^T g \left(g^T g + \mu I\right)^{-1} z \\ &\quad -z^T \left(g^T g + \mu I\right)^{-1} g^T \Delta g \left(g^T g + \mu I\right)^{-1} z. \end{aligned}$$

Since $g^+ \equiv \left(g^T g + \mu I\right)^{-1} g^T$, $Y_d = g^+ Y_s$, and $z = g^T Y_s$ then

$$Y_d = \left(\mu I + g^T g\right)^{-1} z.$$

Thus,

$$\Delta f = 2Y_d^T \Delta z - Y_d^T \Delta g^T g Y_d - Y_d^T g^T \Delta g Y_d = 2\Delta z^T Y_d - 2Y_d^T g^T \Delta g Y_d.$$

This can be rewritten as:

$$\Delta f = 2\left(\Delta g Y_s\right)^T Y_d - 2Y_d^T g^T \Delta g Y_d = 2\left(Y_s - g Y_d\right)^T \Delta g Y_d = 2E^T \Delta g Y_d.$$

Because we are dealing with matrices, we will rewrite this in stack form as $\{\Delta f\}^s = 2\left(Y_d^T \otimes E^T\right)\{\Delta g\}^s$. As $\Delta g \to 0$, we are left with

$$Df = 2\left(Y_d^T \otimes E^T\right)(Dg).$$

Since $E^T E = -\left(f + \mu Y_d^T Y_d\right)$, then it follows that $D\left(E^T E\right) = -Df - \mu D\left(Y_d^T Y_d\right)$. So, now we must compute $D\left(Y_d^T Y_d\right)$. Recall that $Y_d = g^+ Y_s$, where

$$g^+ \equiv \left(g^T g + \mu I\right)^{-1} g^T.$$

So,

$$\Delta\left(Y_d^T Y_d\right) = \Delta Y_d^T Y_d + Y_d^T \Delta Y_d + O\left(\|\Delta Y_d\|^2\right) \cong 2Y_d^T \Delta Y_d,$$

dropping higher order terms. Noting that $\Delta Y_d = \Delta g^+ Y_s$, and (dropping higher order terms) that

$$\Delta g^+ = \Delta\left(g^T g + \mu I\right)^{-1} g^T + \left(g^T g + \mu I\right)^{-1} \Delta g^T,$$

it follows that

$$\Delta g^+ = \left(\mu I + g^T g\right)^{-1} \Delta g^T - \left(\mu I + g^T g\right)^{-1} \Delta\left(g^T g\right)\left(\mu I + g^T g\right)^{-1} g^T.$$

If we then recall that $\Delta\left(g^T g\right) = \Delta g^T g + g^T \Delta g$, we can write the following expression:

$$\begin{aligned}\Delta g^+ &= \left(\mu I + g^T g\right)^{-1} \Delta g^T \\ &\quad - \left[\left(\mu I + g^T g\right)^{-1} \Delta g^T g + \left(\mu I + g^T g\right)^{-1} g^T \Delta g\right]\left(\mu I + g^T g\right)^{-1} g^T\end{aligned}$$

and we have that

$$\Delta g^+ = \left(\mu I + g^T g\right)^{-1} \Delta g^T - \left(\mu I + g^T g\right)^{-1} \Delta g^T g g^+ - g^+ \Delta g g^+.$$

This leads us to rewriting the expression $\Delta\left(Y_d^T Y_d\right) = 2Y_d^T \Delta Y_d$, as follows:

$$\begin{aligned}\Delta\left(Y_d^T Y_d\right) = 2Y_d^T \Delta Y_d &= 2Y_d^T\left(\mu I + g^T g\right)^{-1} \Delta g^T Y_s \\ &\quad - 2Y_d^T\left(\mu I + g^T g\right)^{-1} \Delta g^T g g^+ Y_s - 2Y_d^T g^+ \Delta g g^+ Y_s \\ &= 2Y_s^T \Delta g\left(\mu I + g^T g\right)^{-1} Y_d - 2Y_d^T g^T \Delta g\left(\mu I + g^T g\right)^{-1} Y_d - 2Y_d^T g^+ \Delta g Y_d \\ &= 2\left(Y_s - g Y_d\right) \Delta g\left(\mu I + g^T g\right)^{-1} Y_d - 2Y_d^T g^+ \Delta g Y_d.\end{aligned}$$

Which leads us to a final expression:

$$\Delta\left(Y_d^T Y_d\right) = 2\left[\left(\left(\mu I + g^T g\right)^{-1} Y_d\right)^T \otimes E^T - Y_d^T \otimes Y_d^T g^+\right]\{\Delta g\}^s.$$

Note that the Kronecker product is defined in the Appendix ($A.1$). Thus, we arrive at our stated objective with an expression for $D\left(Y_d^T Y_d\right)$ as follows:

$$D\left(Y_d^T Y_d\right) = 2\left\{\left[\left(\mu I + g^T g\right)^{-1} Y_d \otimes E\right]^T - \left(Y_d \otimes {g^+}^T Y_d\right)^T\right\} Dg$$

where $g^{+^T} = g\left(\mu I + g^T g\right)^{-1}$. So finally, recalling that $D\left(E^T E\right) = -Df - \mu D\left(Y_d^T Y_d\right)$, we arrive at:

$$D\left(E^T E\right) = -2\left(Y_d^T \otimes E^T\right)(Dg) - \vartheta \tag{17}$$

where

$$\vartheta = 2\mu\left\{\left[\left(\mu I + g^T g\right)^{-1} Y_d \otimes E\right]^T - \left(Y_d \otimes g^{+^T} Y_d\right)^T\right\} Dg.$$

Since $J^T e = D\left(E^T E\right)$, the value we compute for this $D\left(E^T E\right)$ will be used in our optimization algorithm: $R_{k+1} = R_k - \left(H + \mu I\right)^{-1} J^T e$. At this point, we have only to compute the Hessian, H, for use in the algorithm.

3.8.2.3 Derivation of the Approximate Hessian:

Recalling that $E = Y_s - gY_d$, then

$$\Delta E = \Delta Y_s - \Delta g Y_d - g\Delta Y_d.$$

But since $\Delta Y_s = 0$, then

$$\Delta E = -\Delta g Y_d - g\Delta Y_d \tag{18}$$

Now we must solve for an expression for ΔY_d. As one recalls: $\Delta Y_d = \Delta g^+ Y_s$ where

$$\begin{aligned}\Delta g^+ &= \Delta\left(\mu I + g^T g\right)^{-1} g^T + \left(\mu I + g^T g\right)^{-1} \Delta g^T \\ &= \left(\mu I + g^T g\right)^{-1} \Delta g^T - \left(\mu I + g^T g\right)^{-1} \Delta\left(g^T g\right)\left(\mu I + g^T g\right)^{-1} g^T.\end{aligned}$$

Since $\Delta\left(g^T g\right) = \Delta g^T g + g^T \Delta g$, then,

$$\begin{aligned}\Delta g^+ \;=\; & \left(\mu I + g^T g\right)^{-1} \Delta g^T \\ & - \left[\left(\mu I + g^T g\right)^{-1} \Delta g^T g + \left(\mu I + g^T g\right)^{-1} g^T \Delta g\right]\left(\mu I + g^T g\right)^{-1} g^T.\end{aligned}$$

Thus, an expression for ΔY_d can be written as

$$\Delta Y_d = \left(\mu I + g^T g\right)^{-1} \Delta g^T Y_s - \left[\left(\mu I + g^T g\right)^{-1} \Delta g^T g + \left(\mu I + g^T g\right)^{-1} g^T \Delta g\right] Y_d.$$

Cleaning this up a bit by collecting appropriate terms, yields the following expression for ΔY_d :

$$\Delta Y_d = \left(\mu I + g^T g\right)^{-1} \Delta g^T \left[Y_s - g Y_d\right] - \left(\mu I + g^T g\right)^{-1} g^T \Delta g Y_d.$$

This can, in turn, be substituted into the equation (18) to show that

$$\Delta E = -\Delta g Y_d - g\left(\mu I + g^T g\right)^{-1} \Delta g^T E + g\left(\mu I + g^T g\right)^{-1} g^T \Delta g Y_d.$$

Upon further inspection, the expression reduces to:

$$\Delta E = -\Delta g Y_d - g^{+T} \Delta g^T E + g^{+T} g^T \Delta g Y_d.$$

And so we have $-\Delta E = \left(I - g^{+T} g^T\right) \Delta g Y_d + g^{+T} \Delta g^T E$. Or written in stack form:

$$-\Delta E^s = Y_d^T \otimes \left(I - g^{+T} g^T\right) \Delta g^s + \left(g^{+T} \otimes E^T\right) \Delta g^s.$$

Finally, we arrive at our expression for the Hessian:

$$H \simeq (DE)\,(DE)^T \tag{19}$$

where

$$DE^s = -Y_d^T \otimes \left(I - g^{+T} g^T\right) Dg^s - \left(g^{+T} \otimes E^T\right) Dg^s. \tag{20}$$

CHAPTER IV

THE DISCRETE PROJECTION PURSUIT LEARNING MODEL (DPPLM)

4.1 Mathematical Framework

4.1.1 Problem Statement

Given an unknown multivariate function of dimension $n \gg 1$, $f \in \mathcal{L}_2$, our objective is to approximate f to within a prescribed degree of accuracy based on a finite set of input–output data (x_k, y_k), $k = 1, \ldots, N$ by projecting f along a finite set of directions and constructing a nonlinear model composed of the superposition of 1–dimensional functions along the projection directions that best fit the input–output data.

4.1.2 Theory

Before beginning the analysis, it is useful to first define the mathematical framework within which we will be working. For approximation purposes, we consider a subset of Lebesgue measurable multi-dimensional functions $f : \Re^n \rightarrow \Re$ with rectangular domain $\mathcal{D} = [\alpha_1, \beta_1] \times \cdots \times [\alpha_n, \beta_n]$. Without loss of generality, by scaling and/or shifting the coordinate axes if necessary, we shall assume that $\mathcal{D} = [0, 1]^n$. The $\mathcal{L}_2$ norm of $f : \Re^n \rightarrow \Re$ on the domain $\mathcal{D}$ is defined as, $\|f\|_2 = \left(\int_{\mathcal{D}} |f|^2 dx\right)^{1/2}$. The notation $\mathcal{L}_2(\mathcal{D})$ denotes the space of functions where each function f is (Lebesgue) measurable and $\|f\|_2 < \infty$.

The scalar product of two functions $f, g \in \mathcal{L}_2(\mathcal{D})$ is defined as

$$< f, g >= \int_{\mathcal{D}} f(x)g(x)dx$$

Motivated by the fact that we will be approximating multivariate functions on a bounded compact domain, our analysis will be carried through by representing our function in terms of its Fourier series. This seems well justified when considering that the Fourier series representation of a function is intended for functions defined on a bounded compact domain space. Analysis using the Fourier transform would be equally valid but its ability to represent a function defined on all of $\Re^n$ would be largely unnecessary for purposes here. Letting and $\mathcal{Z}^n = \underbrace{\mathcal{Z} \times \cdots \times \mathcal{Z}}_{n\text{–times}}$ then the Fourier series of $f \in \mathcal{L}_2(\mathcal{D})$ can be expressed by the series,

$$\check{f}(x) = \sum_{\mathbf{k} \in \mathcal{Z}^n} \hat{f}_{\mathbf{k}} e^{j\boldsymbol{\omega}_{\mathbf{k}}^T x} \tag{21}$$

where $\boldsymbol{\omega}_{\mathbf{k}} = 2\pi\mathbf{k}$

$$\hat{f}_{\mathbf{k}} = \int_{\mathcal{D}} f(x) e^{-j\boldsymbol{\omega}_{\mathbf{k}}^T x} dx$$

From the well known $\mathcal{L}_2$ theory, the Fourier series of an $\mathcal{L}_2$ function converges in the $\mathcal{L}_2$ sense so that $\left\| f - \check{f} \right\|_2 = 0$. Also by Parseval's theorem the $\mathcal{L}_2$ of f and the 2–norm of its coefficients coincide: $\|f\|_2^2 = \sum_{\mathbf{k} \in \mathcal{Z}^n} \left| \hat{f}_{\mathbf{k}} \right|^2$.

To develop the projection pursuit formulation let $\mathcal{K}$ be the following subset of $\mathcal{Z}^n$

$$\mathcal{K} = \left\{ \mathbf{k} \in \mathcal{Z}^n : \gcd(k_1, \ldots, k_n) = 1,\ k_j > 0,\ j = \min_{1 \le i \le n} k_i \neq 0 \right\}$$

where $\gcd(k_1, \ldots, k_n)$ denotes the greatest common divisor of $k_1, \ldots, k_n$.

It can be easily seen that $\mathcal{Z}^n = \cup_{r \in \mathcal{Z}} r\mathcal{K}$ and consequently the Fourier series of f, given that f is continuous, can be expressed as

$$f(x) = \hat{f}_{\mathbf{0}} + \sum_{\mathbf{k} \in \mathcal{K} \setminus \mathbf{0}} \sum_{r \in \mathcal{Z}} \hat{f}_{\mathbf{k}r} e^{j2\pi rz},\ z = \mathbf{k}^T x$$

Defining

$$f_{\mathbf{k}}(z) := \sum_{r \in \mathcal{Z}} \hat{f}_{\mathbf{k}r} e^{j2\pi rz}$$

for $\mathbf{k} \neq \mathbf{0}$ and $f_{\mathbf{0}}(z) = \hat{f}_{\mathbf{0}}$ (constant), then

$$f(x) = \sum_{\mathbf{k} \in \mathcal{K}} f_{\mathbf{k}}(z),\ z = \mathbf{k}^T x$$

We can summarize the results obtained so far in form of the following theorem:

Theorem 20 *A continuous function $f \in \mathcal{L}_2(\mathcal{D})$ can be decomposed into an* infinite *number of single variable, mutually orthogonal functions $f_{\mathbf{k}}$, i.e., $< f_{\mathbf{k}}, f_{\mathbf{k}'} >= 0$, $\mathbf{k} \neq \mathbf{k}'$:*

$$f(x) = \sum_{\mathbf{k} \in \mathcal{K}} f_{\mathbf{k}}(z), \; z = \mathbf{k}^T x$$

PROOF The only thing we need to prove is the orthogonality of these functions.

$$\begin{aligned}
\langle f_{\mathbf{k}}, f_{\mathbf{m}} \rangle &= \int_{\mathcal{D}} \left(\sum_{r \in \mathcal{Z}} \hat{f}_{\mathbf{k}r} e^{j2\pi r \mathbf{k}^T x} \right) \left(\sum_{s \in \mathcal{Z}} \hat{f}_{\mathbf{m}s} e^{-j2\pi s \mathbf{m}^T x} \right) dx \\
&= \sum_{r \in \mathcal{Z}} \sum_{s \in \mathcal{Z}} \hat{f}_{\mathbf{k}r} \hat{f}_{\mathbf{m}s} \int_{\mathcal{D}} e^{j2\pi (r\mathbf{k} - s\mathbf{m})^T x} \\
&= \sum_{r \in \mathcal{Z}} \sum_{s \in \mathcal{Z}} \hat{f}_{\mathbf{k}r} \hat{f}_{\mathbf{m}s} \int_0^1 e^{j2\pi (rk_1 - sm_1)x_1} dx_1 \cdots \int_0^1 e^{j2\pi r (rk_n - sm_n)x_n} dx_n
\end{aligned}$$

The right hand side of the above expression is nonzero if only if

$$\int_0^1 e^{j2\pi (rk_i - sm_i)x_i} dx_i \neq 0, \; i = 1, \ldots, n$$

or equivalently $rk_i = sm_i$, $i = 1, \ldots, n$ by the orthogonality of $e^{j2\pi kx}$ functions. We claim that $\mathbf{k} = \mathbf{m}$. If this is not the case, let $d = \gcd(r, s)$, and put $r' = r/d$, and $s' = s/d$. Then $r'k_i = s'm_i$ and since r' and s' are relatively prime, they must divide m_i and k_i respectively. But this is a contradiction since elements of $\mathbf{k}$ and $\mathbf{m}$ are relatively prime. By the definition of set $\mathcal{K}$ it is also impossible for $\mathbf{k} = -\mathbf{m}$ unless they are both zero (and consequently equal). Thus $r = s$ and $\mathbf{k} = \mathbf{m}$ implying that

$$\langle f_{\mathbf{k}}, f_{\mathbf{m}} \rangle = \begin{cases} \sum_{r \in \mathcal{Z}} \left\| \hat{f}_{\mathbf{k}r} \right\|^2 = \|f_{\mathbf{k}}\|^2, & \mathbf{k} = \mathbf{m} \\ 0, & \mathbf{k} \neq \mathbf{m} \end{cases}$$

4.2 Universal Approximation Capability

Within the context of our algorithmic implementation, we would like to show that numerical integration over a selected set of directions can approximate the infinite set

to within an arbitrary degree of closeness. Or, in other words, it must be proven that this discretized approach possesses universal approximation capabilities. Drawing upon the concept of basis functions as described in section (3.3), the derivation of universal approximation follows.

Theorem 21 *For any function $f \in \mathcal{L}_2(\mathcal{D})$ and $\varepsilon > 0$, we can find a finite number of directions and basis functions per direction such that the resulting approximation will estimate f to within ε, such that $\left\| f(x) - \sum_{i,j} \hat{f}_{ij}(x) \right\| < \varepsilon$.*

PROOF

By Theorem (20), we recall that $f(x) = \sum_{\mathbf{k} \in \mathcal{K}} f_{\mathbf{k}}(z)$, or redefining indices, $f(x) = \sum_{i=1}^{\infty} f_i\left(u_i^T x\right)$. Since f_i's are mutually orthogonal $\|f\|^2 = \sum_{i=1}^{\infty} \|f_i\|^2 < \infty$. Thus there exists $M > 0$ such that $\left\| f - \sum_{i=1}^{M} f_i \right\| = \sqrt{\sum_{i=M+1}^{\infty} \|f_i\|^2} < \frac{\epsilon}{2}$. For each direction u_i , we shall choose enough basis functions $\{\phi_{ij}\}$, $1 \leq j \leq n_i$, such that $\widetilde{f}_i = f_i - \sum_{j=1}^{n_i} \theta_{ij}\phi_{ij}$ satisfies $\left\| \widetilde{f}_i \right\| < \frac{\epsilon}{2M}$. So, our approximation can be written as:

$$f = \sum_{i=1}^{M} \sum_{j=1}^{n_i} \theta_{ij}\phi_{ij} + \sum_{i=1}^{M} \widetilde{f}_i(z) + \widetilde{f}(z)$$

where $\tilde{f} = f - \sum_{i=1}^{M} f_i$. Thus, the error associated with the approximation is: $e = \sum_{i=1}^{M} \widetilde{f}_i(z) + \widetilde{f}(z)$. And we have this error bounded by:

$$\|e\| \leq \sum_{i=1}^{M} \left\| \widetilde{f}_i \right\| + \left\| \widetilde{f} \right\| \leq M \cdot \frac{\epsilon}{2M} + \frac{\epsilon}{2} = \epsilon.$$

The theorem is now proven by setting $\widehat{f_{ij}} = \theta_{ij}\phi_{ij}$.

4.3 Optimization

There are alternative approaches for the optimization procedure that would be particularly useful for identifying functions with bounded domains. For instance, instead

of using a multidimensional gradient search method to find the optimal directions, an exhaustive search of the entire domain of projection directions could be employed. Or if an exhaustive search would prove too computationally intensive for the domain of interest, an efficient search scheme could be attempted after the linear trends are removed to target the remaining effects. In this case, a genetic algorithm could be used to identify the projection directions. Together with the use of limited coupling, the genetic algorithm approach could determine the desired projection directions while avoiding getting caught in local minima.

Recalling theorem(20) , the function to be modeled can be written simply as:

$$f = f_0 + f_1 + f_2 + ... = \sum_{k=1}^{\infty} f_k$$

We note a partial sum of these low-dimensional functions:

$$\widehat{f_r} = f_0 + f_1 + f_2 + ... + f_M = \sum_{k=1}^{M} f_k$$

Thus the error given the orthogonality of these functions, $\langle f_i,\ f_j \rangle = 0$, can be expressed as:

$$\left\| f - \widehat{f_r} \right\|_2^2 = \sum_{k=M+1}^{\infty} \|f_k\|_2^2 \tag{22}$$

This expression for the 2-norm of the error can then be used as the evaluation criterion of our approximation. This metric can be used in the development of a random search routine, such as a genetic algorithm, for the optimal projection directions.

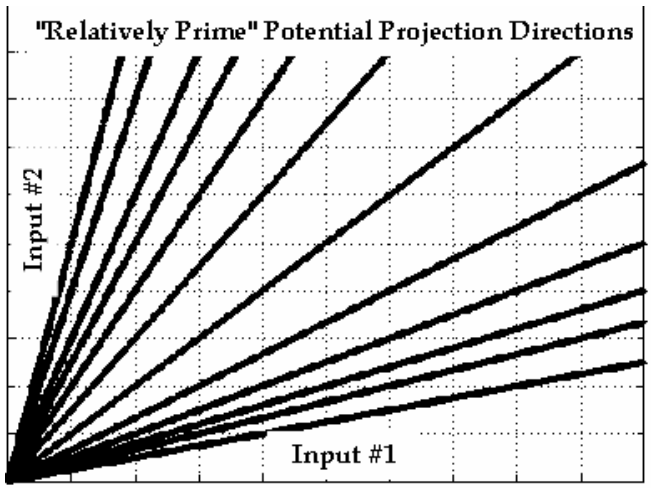

Figure 20: Relatively Prime Projection Directions

Such a random search approach could be coded to identify the contributions of each projection direction to the overall governing function. As such, the norm of this contribution for each attempted projection direction would be the fitness function for evaluating the performance of a given direction. Further, by reducing coupling and assuming the function is bandlimited, the search space of possible projection directions is finite. Thus, the problem of finding the optimal projection directions seems well-suited for a such a search technique. And, of course, once the directions have been chosen, these can be fed back into the local basis function network to accurately model the given system. In the example simulations to follow later in this dissertation, an exhaustive search will be utilized with low-degree of coupling assumed, as the dimensionality of the input space is small enough to do so. However, we will briefly explore the potential implementation of a GA search routine given that such a method would likely be necessary if the dimensionality of the problem space truly quite high. Indeed, a brief simulation investigation of this GA adapted method will also be presented following the comprehensive simulations.

4.3.1 Genetic Algorithm

A genetic algorithm is an optimization procedure that searches a function's solution space via a simulated version of Darwinian evolution, i.e., the survival of the fittest strategy. In fact, the algorithm works in a manner that is quite similar to the well-known biological process upon which it is based. In general, the "fittest" individuals of a population tend to reproduce and survive to the next generation, thus improving the overall fitness of successive generations. However, there is still a chance that "inferior" individuals might also survive and reproduce.

One advantage of genetic algorithms over some other potential optimization choices is that genetic algorithms have been shown to solve linear and nonlinear problems by exploring all regions of a search space and exponentially exploiting promising areas through the procedures of mutation, crossover, and selection operations [63]. The basic steps of a genetic algorithm (GA) are summarized in the algorithm that follows. We will touch on each of the major components in the paragraphs that follow.

4.3.1.1 Algorithm

Basic Steps of the Genetic Algorithm Scheme

1. Supply a population P_0 of N individuals and respective function values.
2. $i \longleftarrow 1$
3. $P_i' \longleftarrow selection\_function(P_i - 1)$
4. $P_i \longleftarrow reproduction\_function(P_i')$
5. $evaluate(P_i')$
6. $i \longleftarrow i + 1$
7. Repeat step 3 until termination
8. Output rank-ordered population of best solutions found

The use of a genetic algorithm requires the determination of six fundamental facets: chromosome representation, the selection function, the genetic operators making up the reproduction function, the creation of the initial population, the termination criteria, and the evaluation function. The rest of this section describes each of these issues.

4.3.1.2 Solution Representation

For any GA, a chromosome representation is needed to describe each individual in the population. The representation scheme determines how the GA problem is structured and also determines the genetic operators that are to be used. Each individual, or chromosome, is made up of a sequence of genes from a certain alphabet. In this case, we use bounded integer values. Thus, each chromosome is bounded by the infinity norm of its composite genes.

4.3.1.3 Selection Function

The selection of individuals to produce successive generations plays an extremely important role in the genetic algorithm. Often, a probabilistic selection is performed based upon the individual's fitness such that the better individuals have an increased chance of being selected. An individual in the population can be selected more than once with all individuals in the population having a chance of being selected to reproduce into the next generation. There are several potential schemes for the selection process: roulette wheel selection and its extensions, scaling techniques, tournament, elitist models, and ranking methods [34], [63].

In our case, we enlist the services of the tournament selection method. Tournament selection only requires the evaluation function to map solutions to an ordered set and does not assign probabilities. Tournament selection works by selecting j individuals randomly, with replacement, from the population, and inserts the best of the j into the new population. This procedure is repeated until N individuals have been selected.

4.3.1.4 Genetic Operators

Genetic Operators provide the basic search mechanism of the genetic algorithm. The operators are used to create new solutions based on existing solutions in the population. There are two basic types of operators: crossover and mutation. Crossover takes two individuals and produces two new individuals while mutation alters one individual to produce a single new solution.

Crossover For our purposes the crossover function works as demonstrated in the accompanying figure, Figure (21). In the figure, we assume an initial input space consisting of 5 dimensions. Thus, the chromosomes, or individuals, listed are the vector representations of projection directions that we are mating to achieve a new set of vector representations. For this example, the crossover point was randomly

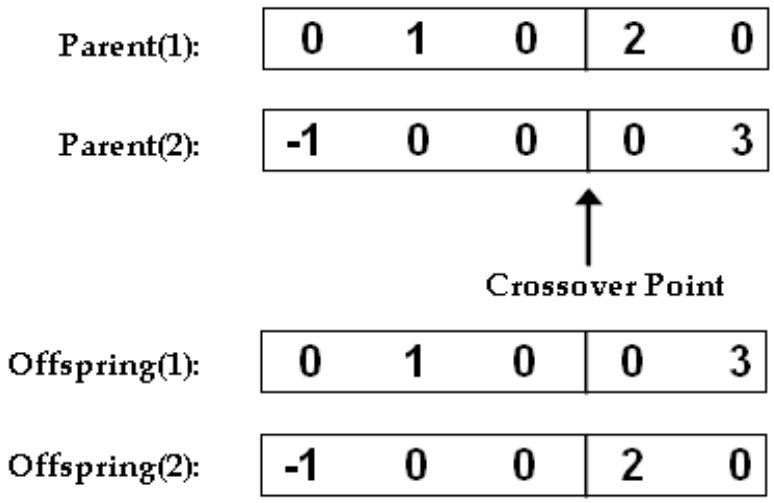

Figure 21: Demonstration of Crossover Procedure

selected to be between the third and fourth components of the vectors. Note that there will be some instances in which the crossover will naturally produce unchanged offspring.

Mutation: Because of their inherent randomness, mutations have the unique ability to keep the optimization procedure from getting stuck in a local minimum: This provides one clear-cut advantage of the evolutionary approach to optimization over its gradient descent and Newtonian cousins. The mutation scheme adopted for use in our algorithm was governed by the type of chromosomes we used. Because of the complexity of the projection direction chromosomes, in which each direction is comprised of component genes with the non-zero components representing the projection hyperplane and the magnitude of those components identifying the specific direction within the hyperplane, a multi-attribute mutation scheme was employed. Thus, mutations can occur in any one of 4 different ways, which are assigned randomly and with equal probability. The first type of mutation is the simple switching of vector components within a projection direction. This is illustrated in figure (22).

Another potential mutation is given in figure (23). With this flip-and-switch

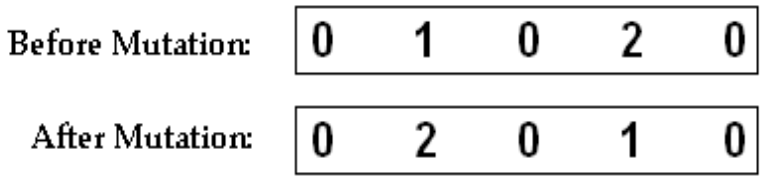

Figure 22: Illustration of a "Switch" Mutation (Type I)

mutation, components are switched and signs are flipped.

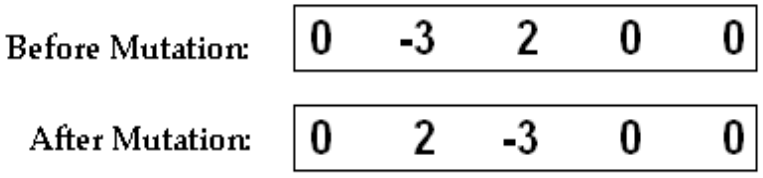

Figure 23: Illustration of a "Flip-and-Switch" Mutation (Type II)

A third mutation type involves shifting one non-zero direction component to an unoccupied slot, where an unoccupied slot is defined as a component position having a zero value. This is portrayed in figure (24).

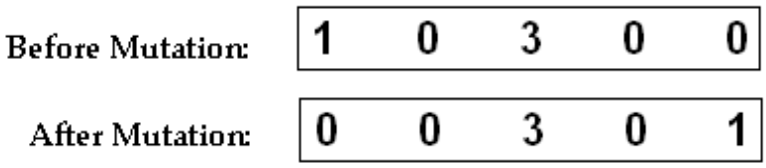

Figure 24: Illustration of a "Shift" Mutation (Type III)

The fourth type of mutation employed is the flip-and-shift. An example of this is provided in figure (25).

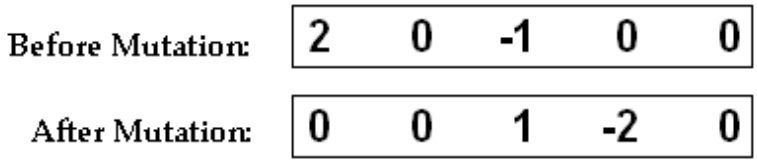

Figure 25: Illustration of a "Flip-and-Shift" Mutation (Type IV)

4.3.1.5 Initial Population Generation:

The GA must be provided an initial population as indicated in step 1 of the algorithm provided in section 4.3.1.1. The most common method is to randomly generate solutions for the entire population. This is the technique we employ in our version of the genetic algorithm.

4.3.1.6 Stopping Criteria:

The GA moves to each successive generation by selecting and reproducing parents until the termination criterion is met. In our case, this stopping criterion is a specified maximum number of 500 generations.

4.3.1.7 Evaluation function:

The evaluation function was presented in equation (22) and is the same one used in the other PPLM algorithms we have implemented. It is provided again below, for the reader's convenience.

$$\left\| f - \widehat{f}_r \right\|_2^2 = \sum_{k=M+1}^{\infty} \|f_k\|_2^2$$

Once again, the terms were as originally defined earlier in this chapter.

4.3.2 Projection Direction Optimization:

Evaluating each individual direction separately, we choose the top m_{sub} directions (a subset of our total number of directions) based on our selection criterion of the

minimization of the approximation error, from equation (22). From this subset of selected directions, the forward selection procedure is performed to choose the projection directions, adding sequentially those with the biggest impact on the error. In this way, the model space is built up until the stopping criterion is reached.

4.3.3 Function Approximation

Problem Statement:

Given a high-dimensional, nonlinear dataset, we are seeking to accurately approximate the underlying response function, $f(x)$ governing the sample space, S. Specifically, we are seeking the directions, $u_1, u_2, ..., u_M$, and the one-dimensional functions, $\widehat{f}_{u_k}$, along those directions that minimize $\left(\sum_{j=1}^{N} \left| y_j - \sum_{k=1}^{M} \widehat{f}_{u_k}(z_{kj}) \right|^2\right)$, where

$$\widehat{f}_{u_k}(z_{kj}) = \sum_{j=1}^{n_k} w_j \phi_{ij}(z_{ki}),$$

$z_{ki} = u_k^T x_i$, and w is a vector of weights on the basis functions. Each function, $\widehat{f}_{u_k}$, is an approximation of the underlying response surface along that direction and is formulated by using a suitable set of bases, as described in section (3.3). The grid spacing methodology upon which these bases will operate is as outlined in section (3.3.4).

Because we are dealing with nonlinear, high-dimensional datasets, the task of finding the optimal directions, u_k, is difficult. Our approach will be to solve for them individually, fitting the best 1-dimensional function along each, and then at the end, put all of these directions together to find the optimal model.

Thus, we will attempt to solve our estimation problem with a two-stage optimization. The first stage involves solving for the best fits given fixed directions, $u_1, ..., u_M$. This is a standard linear least squares problem to minimize the cost function $\|\Phi W - Y\|^2$, with the solution given by $W_{LS} = \Phi^+ Y$, where Y is the response, $\Phi^+ = \left(\Phi^T \Phi\right)^{-1} \Phi^T$ and Φ is an $N \times \left(\sum_{k=1}^{M} n_k\right)$ matrix defined as $\left[\Phi_{kj}(z_{ki})\right]^{ST}$.

Note that each $\Phi_{kj}(z_{ki})$ is a suitable basis function as described in section (3.3), and the matrix W contains the individual weights for each of those bases. Details for the stacking operation for matrices is provided in the appendix. The 2nd level of the optimization is a nonlinear problem that involves finding the optimal projection directions.

$$e_{LS} = \min_{u_1,\ldots,u_m \in \Re, \|u_i\|=1} (\Phi W_{LS} - Y)^T (\Phi W_{LS} - Y) \tag{23}$$

4.3.4 Exact Function Approximation – an infinite sample

Before moving on, one should be careful to note the approximations being made when characterizing the function estimation in the manner described above. To begin, the least-squares error shall be defined as:

$$e_{LS} = \min \int_{\mathcal{X}} e^2 dx$$

where $e = \Phi W - \widehat{f}$, and $\widehat{f} = \Phi W$. The terms Φ and W are as described previously. However, the basis function matrix for exact function approximation would consist of infinite rows (in the continuous case) as Φ contains as many elements as W. So, we take $\Phi(x)$. By, basing our function estimation only on the available information, we can then solve only for the row we need based on the data.

Rewriting our least squares error based on this information, we have:

$$e_{LS} = \min_{u,W} \int [\Phi(x) W - Y(x)]^T [\Phi(x) W - Y(x)]$$

For a given fixed u, the least-squares solution is

$$W_{LS} = \Phi^+(x) Y(x) = \left(\int \Phi(x) \Phi^T(x) dx \right)^{-1} \int_{\mathcal{X}} \Phi^T(x) Y(x) dx$$

So, the least-squares error can then be written as

$$e_{LS} = \int Y^2(x)\,dx - P_y \left(\int_S \Phi(x)\,\Phi^T(x)\,dx \right)^{-1} P_Y$$

where $P_y = \int_S \Phi^T(x)\,Y(x)\,dx$. If we knew the values of the function, this is how we would solve it for an exact function representation. However, in practice, we do not know the exact values of the function for all possible inputs; hence, the need for our function approximation in the first place. The discrete solution of $\max_u \left\| \int_{\mathcal{X}} \Phi^T(x)\,Y(x)\,dx \right\|$ depends on the distribution of the data. We cannot simply take a summation of the terms, as, in this case, the summation is not a good approximation for the integral we are trying to solve. Thus, unless the distribution of the bases is exactly uniform or unless we have infinite data, then we do not have strict orthogonality. So, as an approximation, we discretize the solution. Note that this was the reason for the nonparametric technique of arranging the data into Order statistics as described in section (3.3.4) and in the algorithm details. Such Order statistics were employed to transform the bases into a uniform distribution that could then be utilized to approximate the integrals in the equations above.

As a result of this, we must proceed with a finite approximation to the exact solution.

$$e_{LS} = \min_{u_1,\ldots,u_m \in \Re, \|u_i\|=1} (\Phi W_{LS} - Y)^T (\Phi W_{LS} - Y) \tag{24}$$

4.3.5 Nonlinear Optimization

The next theorem shows that the two-stage optimization problem can be reduced to the following nonlinear optimization.

Theorem 22 *Assuming there exists a function belonging to the class of universal approximators as stated in theorem* (21)*, the optimization problem for finding the best*

function estimator associated with seeking the optimal projection pursuit directions can be formulated as maximizing the explanatory power of the estimate, succinctly stated as

$$c_{ep} = \max_{u_1,\ldots,u_m \in \Re, \|u_i\|=1} Y^T \Phi \Phi^+ Y \tag{25}$$

PROOF

The argument of equation (23) can be written as

$$\begin{aligned} \left(\Phi W_{LS} - Y\right)^T \left(\Phi W_{LS} - Y\right) &= \left(\Phi\Phi^+ Y - Y\right)^T \left(\Phi\Phi^+ Y - Y\right) \\ &= Y^T \left(I - \Phi\Phi^+\right)^T \left(I - \Phi\Phi^+\right) Y \end{aligned}$$

by plugging in for the least-squares solution. Noting that $\left(\Phi\Phi^+\right)^T = \left(\Phi\Phi^+\right)$, we have

$$\begin{aligned} Y^T \left(I - \Phi\Phi^+\right)^T \left(I - \Phi\Phi^+\right) Y &= Y^T \left(I - \Phi\Phi^+\right) \left(I - \Phi\Phi^+\right) Y \\ &= Y^T \left(I - \Phi\Phi^+ - \Phi\Phi^+ + \Phi\Phi^+\Phi\Phi^+\right) Y \end{aligned}$$

Given that $\Phi^+\Phi = I$, this leaves us with

$$Y^T \left(I - \Phi\Phi^+ - \Phi\Phi^+ + \Phi\Phi^+\Phi\Phi^+\right) Y = Y^T \left(I - \Phi\Phi^+\right) Y.$$

Thus, the optimization problem of eq. (23) reduces to

$$c_{ep} = \max_{u_1,\ldots,u_m \in \Re, \|u_i\|=1} Y^T \Phi \Phi^+ Y$$

given that $Y^T Y$ is constant.

Corollary 23 *If the bases, ϕ, that form the basis function matrix Φ are orthogonal, then $\Phi^+ = \left(\Phi^T\Phi\right)^{-1}\Phi^T = \Phi^T$, and the optimization problem reduces to maximizing the 2-norm of the coefficients:*

$$c_{ep} = \max_{u_1,\ldots,u_m \in \Re, \|u_i\|=1} Y^T \Phi \Phi^T Y = \left\|\Phi^T Y\right\|^2$$

where again, Φ is our matrix of basis functions along the given M directions and Y is the response vector.

Corollary 24 *If the basis function matrix* Φ *is 1-dimensional then*

$$c_{ep} = \max_{u} \left\| \overline{\Phi}^T Y \right\|^2 \tag{26}$$

where $\overline{\Phi} = \frac{\Phi}{\|\Phi\|}$.

Proof. Noting that $\Phi^T = \frac{\Phi^T}{\Phi\Phi^T}$ in the 1-dimensional case, then it follows that

$$c_{ep} = \max_{u} \frac{\left\| \Phi^T Y \right\|^2}{\|\Phi\Phi^T\|}$$

which can be rewritten as equation (26) . ■

The algorithm used to implement this modeling technique transforms the raw input data into a finite set of projection directions and assigns the grid spacing along these directions. Sequentially, one-dimensional local basis function fits are generated along these projections. The performance along these directions is evaluated and the direction set is then winnowed down to a more manageable number for simultaneous direction modeling. The final directions are chosen via a forward selection routine.

4.4 Algorithm

Note: m = number of projection directions

R = projection matrix

d = grid point subintervals per direction

The performance criterion is set to the MSE of the validation sample.

1. Initialization

a. Set the number of couplings (for systems with low degrees of coupling, 2 is a good choice)

b. Set the relatively-prime projection directions per hyperplane (integer directions with a maximum infinity-norm of 3 are what we use in our simulations)

c. Assign the number of projection directions on which to run the selection procedure: m_{sub}

d. Set the maximum number of projection directions to include, $m_{\max}$, and the maximum number of consecutive iterations without improvement, $iter_{\max}$, as the stopping criteria and loop through while performance criterion improves, $m = m + 1$. Initialize R to be an $n \times m$ uniform random variable.

e. For $d = d_{\min}$ to $d_{\max}$ (good values of d might range from 3 to 10)

2. Assign grid spacing for each direction and store

a. $\mathbf{Z} = \mathbf{X} * R$ (setting transformed inputs and storing these for repeated use)

b. For each transformed input vector, $\mathbf{Z}_i$, create Order statistics for the observations, O_k , $\forall$ $\mathbf{Z}_k$ where $k \in [1, N]$ and set the nodes for that input.

3. Whittle down the total set of projection directions

a. Loop through each direction, reassigning the transformed inputs and nodes

b. Next, the basis functions can be set

(a). Compute distance of each data point to each node:

i. $\overline{z}_{ij,l} = \frac{z_{ij} - \eta_{ij}}{b_{ij,l}}$

ii. $\overline{z}_{ij,r} = \frac{z_{ij} - \eta_{ij}}{b_{ij,r}}$

(b). Create local piecewise cubic basis functions for each projection (Note: dropping projected input subscript, j, to avoid confusion):

i. For each projection direction, compute the first portion of the basis function (to be designated as $\{\phi\}^1$), accounting for potential unequal base widths along dimensions:

$$\{\phi_i\}^1 = \left\{ \begin{array}{ccc} \left(|\overline{z}_{i,l}| - 1\right)^2 \left(2 \cdot |\overline{z}_{i,l}| + 1\right) & & \text{if } -1 < \overline{z}_{i,l} \leq 0 \\ \left(|\overline{z}_{i,r}| - 1\right)^2 \left(2 \cdot |\overline{z}_{i,r}| + 1\right) & , & \text{if } 0 < \overline{z}_{i,r} \leq 1 \\ 0 & & \text{otherwise} \end{array} \right\}$$

ii. For each projection direction, compute the derivative portion of the basis function (to be designated as $\{\phi\}^2$):

$$\{\phi_i\}^2 = \left\{ \begin{array}{ccc} (|\overline{z}_{i,l}| - 1)^2 \cdot \overline{z}_{i,l} & & \text{if } -1 < \overline{z}_{i,l} \leq 0 \\ (|\overline{z}_{i,r}| - 1)^2 \cdot \overline{z}_{i,r} & , & \text{if } 0 < \overline{z}_{i,r} \leq 1 \\ 0 & & \text{otherwise} \end{array} \right\}$$

iii. Assign basis function for each projection direction:

$$\phi = \left[\begin{array}{cc} \{\phi\}^1 & \{\phi\}^2 \end{array} \right]$$

iv. Append to full basis function matrix: $\Phi = \left[\begin{array}{cc} \Phi & \phi \end{array} \right]$

c. Fit least-squares derived weighting matrix, W: $\Phi W = Y$, where Y is the response vector

(a). $\Phi^+ = \left(\Phi^T \Phi + \gamma I\right)^{-1} \Phi^T$, where γ is some small, positive constant near zero

(b). $W = \Phi^+ Y$

(c). Compute the error: $E = \Phi W - Y$

(d). Evaluate the strength of the fit along the direction:

(Note 1.) Since, $f = \Phi W$,

$\|f\|_2 = \int W^T \Phi^T \Phi W \; dz = W^T \left(\int_0^1 \Phi^T \Phi dz \right) W$

(Note 2.) For cubic basis functions

$$\left(\int_0^1 \Phi^T \Phi dz \right) = \left[\begin{array}{cccc} 13/35 & 9/70 & 11/210 & -13/420 \\ 9/70 & 13/35 & 13/420 & -11/210 \\ 11/210 & 13/420 & 1/105 & -1/140 \\ -13/420 & -11/210 & -1/140 & 1/105 \end{array} \right]$$

(e). Select top m_{sub} directions based on this strength criterion

4. Optimization (Build model from m_{sub} projection population, starting with "best" individual direction, adding directions with the forward selection method), $m = 1$

a. While $(m < m_{\max})$ & $(iter < iter_{\max})$,

b. Loop through m_{sub} directions not yet in model direction population (initially assigned to null space)

i. Repeat Step 3a-3c for $m-1$ total model directions, adding one m_{sub} direction not yet in model direction population

ii. Evaluate e_{LS}

iii. Remove this added direction from the total model direction population

c. Choose the direction that had the lowest e_{LS} in step 4b to be added to the model direction population

d. Fit this same model to the test sample and evaluate $e_{LS,test}$

e. If $e_{LS,test}(m) > e_{LS,test}(m-1)$

Then $iter = iter + 1$

Else $iter = 0$

f. $m = m + 1$

4.5 Experimental Results

As another test of the method's effectiveness, we return to the bearing defect experiment. In this case, the prediction accuracy of the DPPLM model will be compared with that of a continuous PPLM methodology and a feedforward neural network approach. Results for the same three response variables will be presented.

The test procedure is the same as in the continuous case. Once again, the input data are a set of 29 different signal features based on accelerometer and acoustic emissions data, with an $n-1$ set of data being split into a 75% training sample and 25% validation sample for constructing the predictions on each individual observation. The results are again displayed as cumulative (average across all observations) or, in some cases, have been split up into cumulative within groups (average across all defects of a certain defect size).

4.5.1 Experiment Set #1

For this experiment, the defect width was chosen as the response variable. As a comparison of the results, we will illustrate the differences in performances by simply comparing results across the 8 distinct levels of defect. It is clear from figure (26) that the PPLM and DPPLM models estimate the defect widths more accurately than does the best of the artificial neural networks.

Computing overall percent error yields the following:

Table 3: Overall defect width percentage error for Experiment 2

Model	%**Error**	(%**Error**)^2
ANN	32.1%	10.3%
PPLM	16.5%	2.7%
DPPLM	13.1%	1.7%

For comparison purposes, [94] used $(\%\ error)^2$ as the calculated error and produced a prediction error of 11.8% (albeit on a somewhat different sample) using a specialized back-propagation prediction network with parameters attuned to this bearing defect problem. Thus, the results of the similar feedforward networks used (ANN) in this analysis seem consistent with results produced by previous researchers. Therefore, the improved prediction accuracy of the two new methods introduced in this thesis are notable

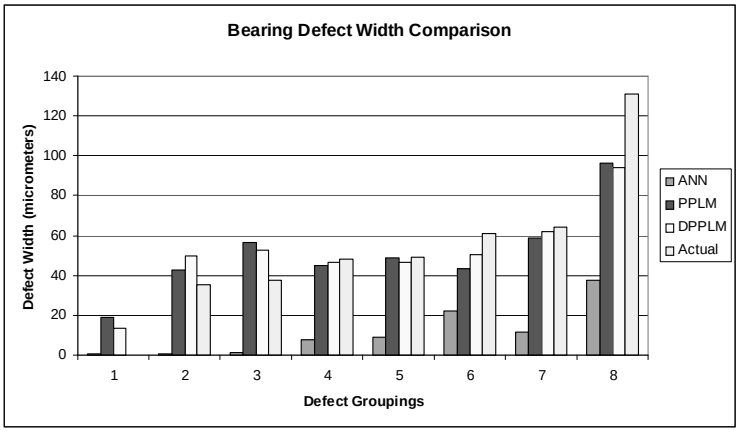

Figure 26: A Comparison of Actual Defect Widths and Predicted Defect Widths

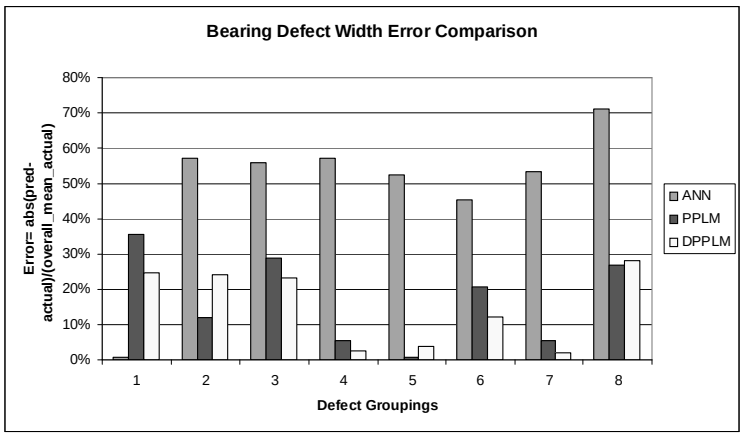

Figure 27: Bearing Defect Width Percentage Error Comparison

4.5.2 Experiment Set #2

Using the same data and defect height as the response, the comparison group predictions for the various models are shown in figure (28). The errors by defect height

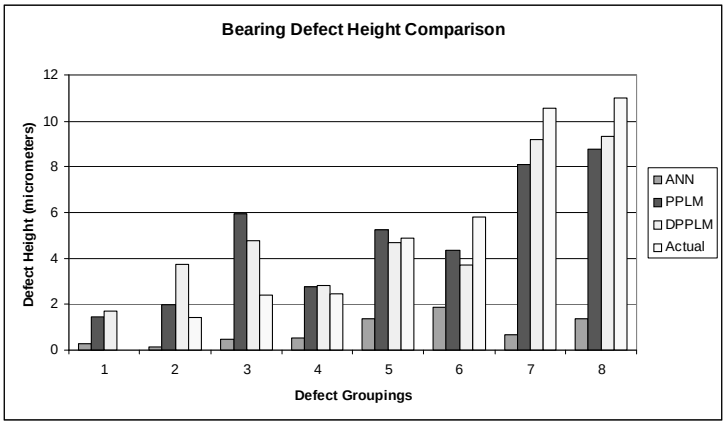

Figure 28: Comparison of Actual Defect Heights and Predicted Defect Heights

size are displaying in figure (29).

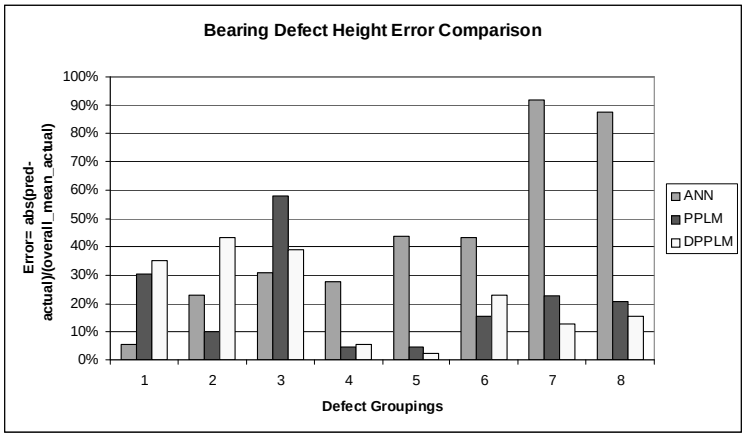

Figure 29: Bearing Defect Height Percentage Error Comparison

Table shows the overall percentage error is lowest for the DPPLM approach.

Table 4: Overall defect height percentage error for Experiment 2

Model	**%Error**	$(\%\mathbf{Error})^2$
ANN	3.1%	0.095%
PPLM	1.7%	0.029%
DPPLM	1.5%	0.023%

Overall, the results show that both the continuous PPLM and discrete projection pursuit learning network perform quite well on the experimental data relative to the best other available methods. The DPPLM even provides a slight improvement in prediction accuracy over the continuous PPLM method described earlier.

CHAPTER V

COMPARISON OF METHODS - SIMULATION RESULTS

Thus far, we have only examined the effectiveness of the newly-devised models with a very limited set of test cases. It is in this chapter that the bulk of the simulation results, a total of 96 separate test cases, will be presented. In the simulations that follow this section, comparisons will be made of the performance of the two projection pursuit learning methods presented in this thesis with various other methods. These other methods used for comparison have been described in great detail earlier. They are projection pursuit regression, MARS, and feedforward neural networks. But, before getting into the simulation results, the issue of overfitting and the steps taken to handle it, must first be addressed.

5.1 Bias-Variance Trade-off / Overfitting

"If you torture the data long enough, they will confess. " – Thomas Mayer [61]

5.1.1 Overfitting

For most classes of models, we can reduce the prediction error of the estimation on the model sample by increasing the complexity of the model structure. While this sharpening of our predictive blade at first may seem advantageous in all circumstances, it is in fact dangerous as the complexity blade is a double-edged sword. We can get a model that is as accurate as we'd like on the model sample by simply making its structure more complicated; but this increased accuracy is gained at a price. While our very flexible model produces a relatively small bias (yields an average prediction

for each snapshot of input values that is close to the true response), this increased model complexity result in an increase in the number of parameters to be estimated. This results in a higher variance of each parameter estimation generated from different datasets.

5.1.2 Bias-Variance Trade-off

The response can be written as $y = f(\mathbf{x};\theta)+\epsilon$. So, $\mu_y = E[y \mid \mathbf{x}]$ represents the actual expected value of the response for a given input state space. Similarly, $\widehat{y} = f\left(\mathbf{x};\widehat{\theta}\right)$ is the estimate provided by our model and its corresponding fitted parameters. Thus, the mean squared error at $\mathbf{x}$ is defined as:

$$MSE(\mathbf{x}) = E\left[\widehat{y} - \mu_y\right]^2$$

which can be rewritten as

$$MSE(\mathbf{x}) = E\left[\widehat{y} - E(\widehat{y})\right]^2 + E\left[E(\widehat{y}) - \mu_y\right]^2 \tag{27}$$

where the expectation, E, is taken over the probability distribution $p(D)$ of all potential datasets of size n. In this way, $\widehat{y}$ is a random variable allowing for the random sampling responsible for generating the particular set of training data, D, from amongst all possible choices from within the theoretical population. Note that different datasets, D, would have led to different models with different estimation parameters and a different set of predictions, $\widehat{y}$. Thus, the expectation, E, in the equation for the mean squared error above represents the expected value of the given random variable over different potential datasets of equivalent size n, each chosen randomly from the source population.

Note that this relationship is written in such a form as to provide us with insights into the bias variance trade-off. Essentially, what we have written is:

$$MSE(\mathbf{x}) = variance + bias^2$$

So a closer inspection of equation (27) now unveils insights into each of its major constituents. The variance term, $E\left[\widehat{y} - E\left(\widehat{y}\right)\right]^2$, provides an expectation for the deviation of our estimate, $\widehat{y}$, across different potential datasets of size n. It measures the sensitivity of $\widehat{y}$ to the specific dataset being used to train the model. To glean a better understanding of this term, let us choose a couple of examples. For instance, if the constant, y_c, was always chosen as our predicted response, without consideration for the data, this variance would be zero. Choosing the other extreme, if we select a very complicated model with many parameters, our predictions, $\widehat{y}$, will tend to vary greatly given a different choice of training dataset.

The bias term, $E\left[E\left(\widehat{y}\right) - \mu_y\right]^2$, reflects the systemic error in our prediction: the deviation of our average predicted response, $E\left(\widehat{y}\right)$, from the true population mean,.μ_y. If we again choose a constant, y_c, as our predicted response, irrespective of the data, we might expect this model to have a large bias term. On the other hand, if a more complex model is employed, our bias (or average prediction) may be substantively lower. This battle of the countervailing forces of bias and variance quantifies the tension between the choice of a simple model (one with low variance and high bias) and a more complicated one (with low bias and high variance).

From a practical standpoint, the average mean squared error over the entire domain of the function being estimated is of interest to us. Thus, we might define the expected MSE with respect to the input distribution, $p(\mathbf{x})$, as $\int MSE(\mathbf{x})\, p(\mathbf{x})\, d\mathbf{x}$. While this quantification of the bias-variance trade-off is of interest theoretically, it is not possible to calculate it in practice because we cannot measure the bias term. Nevertheless, the theoretical bias-variance formulation is quite instructive, as it illuminates the need to choose a model flexible enough to handle variations within the input data, yet not too complex so as to result in the overfitting of the variations of

the noise present in the dataset.

5.1.3 How to Handle Overfitting

While we have measures in place to try to improve the overall fit of the model, what can be done to reduce the possibility of overfitting the training set? One method is choosing a score function with two components: one to measure the goodness-of-fit of the predictions, and the other component that penalizes model complexity [1]. The problem with this approach is that its effectiveness depends greatly on a good choice of the relative weighting of the two score function components, which can be quite difficult to select appropriately.

Another approach to use is that of external validation. This is the approach, we will be using in the simulations that follow. This approach will be used for the selection of the "best" model within each of the various model types in the comparison. The idea here is to split the data into two mutually exclusive sets: the training set and the validation set. The training set is used to construct the model; the validation set is used to test the effectiveness of the model generated from the training set.

Thus, the validation set is used to choose from among candidate models offered by the training set. After building the model on the training dataset, the score function, which often is a measure of the SSE between the predicted and actual response, is reevaluated on the validation set. Thus, the score function is, itself, a random variable where the randomness has two sources. The first is from the training dataset, while the second is from the data being used to validate it.

Ideally, we would like an unbiased estimate of the score function on future data for each model considered. Here, since the two datasets are assumed independent and randomly selected, the validation score for a given model provides an unbiased estimate of the score for that model for an out-of-sample dataset. So the bias from training is absent from the independent validation estimate.

With this approach, we have a data-driven method for handling the problem of overfitting. Now, when comparing the validation scores, we should better be able to distinguish between a high quality and a low quality model, as a model better able to fit the response surface should produce a lower score. This process deters against choosing those models unduly influenced by the noise of the training set.

5.2 Simulation Procedures

To compare the effectiveness of the newly-formulated continuous and discrete projection pursuit learning methods with other high-dimensional prediction methodologies, several sets of simulation examples are presented. As was mentioned previously, the PPLM and DPPLM methods are not restricted to a single choice of bases. Thus, for the simulations that follow, they were allowed their choice of basis functions to be determined from the data: either a global harmonic basis function set or set of a local cubic basis functions. It should be noted that for the case of the global harmonic bases, the Levenberg-Marquardt PPLM algorithm required a good set of starting conditions. Much work has been done in this area, including [48]. For our purposes, a simple 2-degree of coupling GA was run to establish the initial projection direction matrix. With the local cubic bases, this procedure was not necessary. With the DPPLM, this procedure was not necessary for any choice of bases – the method was capable of converging regardless of initial conditions. Experiments were run on simulated datasets with different forms of the response function, varying numbers of observations, different distributions of the data, and varied levels of noise introduced into the dataset. The examples presented below were chosen to give the reader a feel for the varying effectiveness of the distinct modeling methodologies across different types of functions and different data conditions. With each example, a comparison of the normalized mean-squared errors $\left(nMSE = \frac{MSE}{var(Y)}\right)$ of the models will be provided.

For all of the simulations presented here, each modeling methodology utilized the same training sample of data. Two separate samples of equal size were used across each modeling methodology for validation and testing purposes. As each of the models are dependent on specific user-defined parameters, a data-driven approach, with the aid of the cross-validation sample, was used to select these optimal parameters. Model training continued, building up models of greater and greater complexity, until no further improvement was achieved on the validation sample. Thus, each model was brought to the highest level of complexity that produced not only a low training error, but also the lowest validation data error as well. The same training, cross-validation, and holdout (test) samples were used for all models. The prediction results of the discretized projection pursuit learning network were then compared to those of the other models. In each case, the results presented are for the holdout sample without noise, to isolate the model's effectiveness at predicting the intended response. For each simulated response function, both a high noise ($r^2 = 0.90$) and a low noise ($r^2 = 0.99$) case were run. Noise levels have been parametrized by the coefficient of determination, defined as $r^2 = \frac{\sigma_y^2 - \sigma_{resid}^2}{\sigma_y^2}$, where σ_y is the standard deviation of the response and σ_{resid} is the standard deviation of the residual error. Noise has been added to the training and validation response in such a way as to frame $\sigma_{noise} = \sqrt{\frac{\sigma_{resid}^2}{\sigma_y^2 - \sigma_{resid}^2}}$. Thus, the noise is always scaled to the variance of the response such that $\sigma_{noise}^2 = c\sigma_y^2$, where $c = \frac{1}{r^2} - 1$. [83]

For each of the simulations, two different levels of the number of observations (low=1000 observations, high=2,700 observations), along with two distinct distributions for the noise and the input data (normal and uniform) were tested along with the two noise levels described above. In this manner, we can gain a better insight into the predictive abilities of each of the methodologies tested. The different testing regimes along with their designation is now provided. Results will be presented grouped by

low and high observation number groupings. Then, three letter combinations representing the noise level, the input distribution, and then the noise distribution will be provided. Thus, 'LNU' will refer, for instance, to the low noise, normally-distributed inputs, uniformly distributed noise condition, whereas a designation of 'HUN' would refer to the high noise condition with uniform inputs and where the noise is normally-distributed. The specific input distribution used is $N(0,1)$ for the normalized input condition and $U(-4.5, 4.5)$ for the uniform case. The noise levels are determined from the data as described previously.

5.3 Simulation Results

5.3.1 Simulation Set 1

For the first set of simulations, the response function utilized was the following harmonic function:

$$y = f(X_1, ..., X_5) = \frac{1}{5}\sin(\pi X_1)\sin(\pi X_2) + \sin(\pi X_3)\cos(\pi X_4) + noise$$

The results of each of these simulations is provided in table (5) as a set of normalized mean-squared error comparisons.

As can be seen from table (5), both the DPPLM and PPLM approaches provide considerable improvement on the results of the projection pursuit regression (PPR), feedforward artificial neural networks (ANN), and MARS models. The MARS method was allowed up to 5-degrees of coupling and an unlimited number of terms. With all three of the comparison methods (MARS, projection pursuit regression, and the neural network), the method was allowed as many terms and as

Table 5: nMSE Comparison of 5-Dimensional Harmonic Response Simulations

# of Obs	Data Distr	MARS	PPR	ANN	PPLM	DPPLM
$Obs = 1000$	LNN	1.01205	1.03688	0.40579	0.00008	0.00065
	LNU	1.01407	1.05473	0.24158	0.00015	0.00060
	LUN	1.01210	1.08882	0.96785	0.00011	0.00066
	LUU	1.00980	1.03957	0.93544	0.00014	0.00077
	HNN	1.01231	1.06851	0.27598	0.00153	0.00817
	HNU	1.04108	0.54479	0.58958	0.00102	0.00530
	HUN	1.01375	1.02688	0.96635	0.00120	0.00599
	HUU	1.01030	1.01713	0.91333	0.00156	0.00850
$Obs = 2700$	LNN	1.00089	1.01788	0.14809	0.00009	0.00024
	LNU	1.00154	1.01060	0.52295	0.00003	0.00021
	LUN	0.99904	1.01476	1.00282	0.00009	0.00017
	LUU	1.00510	1.02732	1.00667	0.00006	0.00027
	HNN	1.00253	1.02490	0.55273	0.00035	0.00109
	HNU	1.01074	1.02080	0.47061	0.00033	0.00292
	HUN	1.00462	1.01677	0.87353	0.00032	0.00125
	HUU	1.01262	1.02897	1.00866	0.00070	0.00292

many degrees of coupling as it wanted until it could no longer improve upon the performance criterion. The DPPLM approach worked very well on these rather low dimensional harmonic datasets.

To give the reader a sense for the modeling on these projections, figure (30) is included. From the response function, we see can the important directions. Thus, one of these is now plotted in figure (30), with the data projected onto it. The portion of the response function lying along that direction is displayed along with that of the predictions from the DPPLM approach, which fits the response quite nicely along this direction.

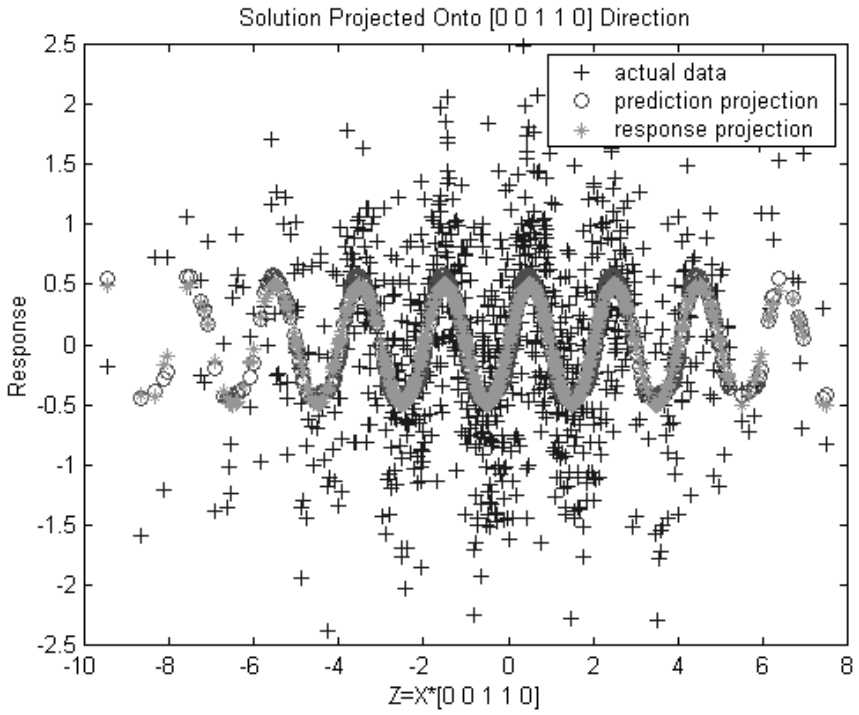

Figure 30: Solution Along a Projection Direction

5.3.2 Simulation Set 2

For the second set of simulations, the harmonic response function was made a bit more complicated, and the dimension of the input space was doubled:

$$\begin{aligned} y = f(X_1, ..., X_{10}) = noise + \frac{4}{5}\sin(\pi X_1)\sin(\pi X_2) \\ + \sin(\pi X_3)\cos(\pi X_4) - \frac{8}{5}\sin(\pi X_5)\cos(2\pi X_6) \\ - \frac{7}{5}\sin(2\pi X_7)\cos(3\pi X_8) + \frac{1}{5}\sin(\pi X_1)\sin(\pi X_8) \end{aligned}$$

Under these conditions, the performance of all of the modeling methodologies suffered. Once again, looking at table (6), the MARS and PPR methods had trouble uncovering any useful information about the response surface. This time, the feedforward neural network was also ineffective at predicting response. The PPLM approach

Table 6: nMSE Comparison of 10-Dimensional Harmonic Response Simulations

# of Obs	Data Distr	MARS	PPR	ANN	PPLM	DPPLM
$Obs = 1000$	LNN	1.02963	1.06282	1.00525	0.92335	0.33605
	LNU	1.01566	1.08564	1.00738	0.87920	0.37324
	LUN	1.01819	1.14552	1.01847	0.81122	0.35239
	LUU	1.02323	1.10336	1.02319	0.80768	0.34996
	HNN	1.03423	1.06936	1.01660	0.71133	0.35320
	HNU	1.03540	1.06042	1.04583	0.85169	0.34261
	HUN	1.02826	1.20014	1.03256	0.90877	0.38122
	HUU	1.01912	1.09495	1.01638	0.86690	0.35624
$Obs = 2700$	LNN	1.01115	1.01612	1.01729	0.78675	0.28828
	LNU	1.01296	1.01304	1.01417	0.87350	0.30172
	LUN	1.00504	1.01708	1.01271	0.80769	0.35072
	LUU	1.00671	1.03233	1.01211	0.89499	0.34385
	HNN	1.00376	1.02066	1.01134	0.78543	0.30855
	HNU	1.00793	1.02914	1.00864	0.77420	0.30291
	HUN	1.00795	1.03343	1.00888	0.82148	0.34802
	HUU	1.00758	1.01373	1.01862	0.84112	0.34719

was still moderately effective at estimating the response function, while the DPPLM did a fine job of identifying this complicated response surface. The primary cause for the success of the DPPLM methodology is quite likely largely due to its ability to employ any basis function. For the past two sets of simulations, the method chose a set of Fourier basis functions for each of the models it constructed.

5.3.3 Simulation Set 3

So, the question arises: how will the PPLM and DPPLM methods perform when restricted to a single choice of bases and on conditions well-suited to the other modeling techniques. For this reason, a two additional sets of simulations were run. For these sets, a polynomial response was chosen, as the other methods are especially good at making predictions on such data. Below, we explore the lower-dimensional case.

$$y = f(X_1, ..., X_5) = -2X_1X_3 - X_2X_3 + \frac{X_1X_4}{2} + noise$$

To constrain the PPLM and DPPLM approaches, the choice of the set of local cubic basis functions was enforced. It should be noted that when employing these

approaches, the researcher could easily choose a polynomial response which would likely generate even better predictions. However, in practice, one does not often know what the functional form of the predictive model should be. Hence, the choice of local cubic bases, which are general all-around good predictors.

Table 7: nMSE(x100) of 5-Dimensional Polynomial Response Simulations

# of Obs	Data Distr	MARS	PPR	ANN	PPLM	DPPLM
$Obs = 1000$	LNN	0.01962	0.16352	0.12224	0.10020	0.05812
	LNU	0.04289	0.32450	0.16333	0.11022	0.06814
	LUN	0.01678	0.08648	0.06713	0.09519	0.02705
	LUU	0.03892	0.11448	0.08617	0.09619	0.05812
	HNN	0.36920	1.14155	1.14028	1.38577	0.53507
	HNU	0.33972	1.75962	0.66132	0.91283	0.44890
	HUN	0.13551	0.81362	1.03206	1.50501	0.42485
	HUU	0.24475	1.48668	0.76152	1.67836	0.64228
$Obs = 2700$	LNN	0.00669	0.11669	0.02409	0.07932	0.01631
	LNU	0.00767	0.10682	0.02372	0.02113	0.01297
	LUN	0.00317	0.03744	0.02780	0.03521	0.01964
	LUU	0.00448	0.04387	0.02261	0.03558	0.01371
	HNN	0.11417	0.50362	0.14789	0.43365	0.15530
	HNU	0.05156	0.78966	0.15048	0.34433	0.14900
	HUN	0.07728	0.51021	0.28725	0.24574	0.18310
	HUU	0.00420	0.47413	0.33840	0.31690	0.19125

With this set of samples, all of the methods performed incredibly well (note that the numbers in table (7) are scaled up by a factor of 100). The continuous PPLM approach provided some slight advantage in performance over PPR, but was generally outpaced by the feedforward neural networks. DPPLM performed quite well versus all of these three, but was unable to match the stellar performance of the MARS models. Still, given the constraints placed on the DPPLM method, it performed admirably. MARS is particularly well suited for approximating relatively simple low-dimensional polynomial response surfaces.

5.3.4 Simulation Set 4

However, this test was for a relatively low-dimensional input space: four important input variables plus one extraneous input. Let us now investigate performance for amore complicated higher-dimensional polynomial response surface.

$$\begin{aligned} y = f(X_1, ..., X_{10}) &= X_1 + X_2 - X_1X_2 - 2X_1X_3 \\ &- X_2X_3 + \frac{X_1X_4}{2} - \frac{X_5X_6}{2} - X_7 \\ &+ \frac{4X_7X_8X_9}{5} + \frac{3X_6X_7X_9}{10} + noise \end{aligned}$$

To constrain the PPLM and DPPLM approaches, the choice of the set of local cubic basis functions was enforced once again.

Table 8: nMSE Comparison of 10-Dimensional Polynomial Response Simulations

# of Obs	Data Distr	MARS	PPR	ANN	PPLM	DPPLM
$Obs = 1000$	LNN	0.1518	0.1468	0.0110	0.0046	0.0024
	LNU	0.1254	0.0880	0.0123	0.0043	0.0209
	LUN	0.2935	0.1652	0.0105	0.0018	0.0014
	LUU	0.2914	0.2693	0.0138	0.0050	0.0012
	HNN	0.0873	0.1155	0.0460	0.0760	0.0218
	HNU	0.1514	0.1116	0.0213	0.0671	0.0300
	HUN	0.2579	0.1673	0.0424	0.0308	0.0166
	HUU	0.2626	0.1043	0.0265	0.0494	0.0310
$Obs = 2700$	LNN	0.1396	0.0746	0.0048	0.0013	0.0007
	LNU	0.1113	0.0744	0.0029	0.0007	0.0006
	LUN	0.2902	0.0557	0.0124	0.0006	0.0003
	LUU	0.2459	0.0987	0.0024	0.0020	0.0004
	HNN	0.1233	0.0729	0.0151	0.0172	0.0257
	HNU	0.0979	0.0761	0.0101	0.0074	0.0043
	HUN	0.2567	0.1399	0.0094	0.0077	0.0057
	HUU	0.2893	0.0706	0.0086	0.0060	0.0034

With this set of samples, both the MARS and the projection pursuit regression methods performed admirably, but their respective performances appear to have

suffered severely with the increase in dimensionality and the corresponding expanding complexity of the response surface. They were both outpaced by the feedforward neural network, PPLM, and DPPLM approaches as is evidenced from table (8). While the discretized projection pursuit learning model and the continuous PPLM managed to generally achieve the best fit of the simulated response surfaces, they also took the most computation time. Thus, if online CPU time is a factor, then either the projection pursuit regression or neural network approaches might prove more suitable for the researcher.

5.3.5 Simulation Set 5

Yet, thus far, we have looked only at response surfaces consisting purely of lower dimensional superpositions of functions of the same form: harmonic in the first two sets and polynomial in the next two sets of simulations. The question of performance on other types of response surfaces still remains. Thus, the last two sets of simulations were run with unusual response surfaces consisting of mixtures of different types of functions. For the first of these sets, let us investigate the low-dimensional case.

$$\begin{aligned} y &= f(X_1, ..., X_5) = -\ln\left(\frac{|X_1 X_2|}{2} + \frac{3}{2}\right) + \left|X_3 X_4 + \frac{2}{5}\right| \\ &\quad -X_4^2 \cdot sign(X_4) - \frac{1}{5}X_5^2 + noise \end{aligned}$$

All of the methods performed well (note that the numbers in table (9) are scaled by a factor of 100). The ANN, PPLM, and DPPLM approaches produce the best results overall, with DPPLM consistently achieving the best performance.

Table 9: nMSE(x100) of 5-Dimensional Nonlinear Response Simulations

# of Obs	Data Distr	MARS	PPR	ANN	PPLM	DPPLM
$Obs = 1000$	LNN	3.4939	2.8827	1.0050	0.6493	0.3076
	LNU	4.1143	2.4604	0.5601	0.5581	0.2906
	LUN	1.4347	1.7756	0.4900	0.6343	0.2826
	LUU	1.8082	3.9834	0.5451	0.5511	0.2585
	HNN	3.5209	7.8786	2.0731	2.4379	1.1894
	HNU	2.5183	6.6348	2.0822	2.3828	1.3918
	HUN	1.6731	4.3674	3.2285	2.7014	1.3076
	HUU	1.6046	3.3031	2.5411	2.2545	1.6994
$Obs = 2700$	LNN	4.2739	1.9636	0.6201	0.4522	0.2413
	LNU	2.8221	2.1333	0.6234	0.2557	0.2561
	LUN	1.5489	1.4339	0.4059	0.3354	0.1271
	LUU	1.6030	1.3755	0.3328	0.3714	0.1423
	HNN	2.9740	1.9548	1.7250	1.6723	1.0174
	HNU	3.2455	2.2868	0.9559	0.8784	0.4689
	HUN	1.7386	3.8024	0.9970	1.3903	0.4859
	HUU	1.9665	2.4337	1.0467	1.3736	0.5619

5.3.6 Simulation Set 6

For the final set of simulations, we expand the dimensionality and complexity of the response surface.

$$\begin{aligned} y &= f(X_1, ..., X_{10}) = noise - \ln\left(\frac{|X_1 X_2|}{2} + \frac{3}{2}\right) \\ &\quad + \left|X_3 X_4 + \frac{2}{5}\right| - sign(X_3) - \frac{1}{5}X_5^2 - \left|X_6 X_7 - \frac{1}{5}\right| \\ &\quad + \ln\left(\frac{9}{20} \cdot |X_8 X_9| + \frac{2}{5}\right) + \frac{2}{5} sign(X_7) \end{aligned}$$

Relative to the prior set of simulations, table (10) shows that the prediction performance of each of the methods deteriorates. The multivariate adaptive regression splines, projection pursuit regression, artificial neural network, and the PPLM approaches all seem to achieve similar performance with a small number of observations.

Table 10: nMSE Comparison of 10-Dimensional Nonlinear Response Simulations

# of Obs	Data Distr	MARS	PPR	ANN	PPLM	DPPLM
$Obs = 1000$	LNN	0.22659	0.32261	0.14285	0.14197	0.04839
	LNU	0.23510	0.29065	0.10220	0.10769	0.04633
	LUN	0.10914	0.21986	0.17577	0.05966	0.02028
	LUU	0.12980	0.16989	0.20783	0.08871	0.01961
	HNN	0.23797	0.33263	0.22984	0.21703	0.08676
	HNU	0.23360	0.28718	0.25080	0.28681	0.08884
	HUN	0.10641	0.26644	0.21387	0.17400	0.04427
	HUU	0.11321	0.22751	0.17431	0.08848	0.03839
$Obs = 2700$	LNN	0.20486	0.23808	0.09210	0.06131	0.05543
	LNU	0.19906	0.24333	0.08534	0.06645	0.05956
	LUN	0.10120	0.19676	0.13347	0.05655	0.02145
	LUU	0.10029	0.19024	0.07432	0.05814	0.02226
	HNN	0.19299	0.23727	0.10056	0.08930	0.06275
	HNU	0.19451	0.24274	0.10772	0.08430	0.06883
	HUN	0.12149	0.22036	0.17326	0.08376	0.03176
	HUU	0.11153	0.19191	0.10980	0.08671	0.03745

However, PPLM and ANN outpace the other two with more observations. Across the board, though, the DPPLM clearly achieves the best results.

But, this type of analysis bring ups an interesting question: with all of this data, can any clear trends be gleaned? Is one type of model well-suited for a certain type of problem, while another is the best choice for a different type of problem? To answer these questions, we now peruse the data more carefully.

5.4 Comprehensive Comparison

For a more comprehensive investigation, we now turn to the composite results of all of the simulations. In figure (31), we can see the average performance of the various methods.

Overall, the DPPLM and PPLM approaches outperformed. However, a closer investigation may shed some light on how differences in the response surface being modeled or in the input conditions might change the performance amongst each of

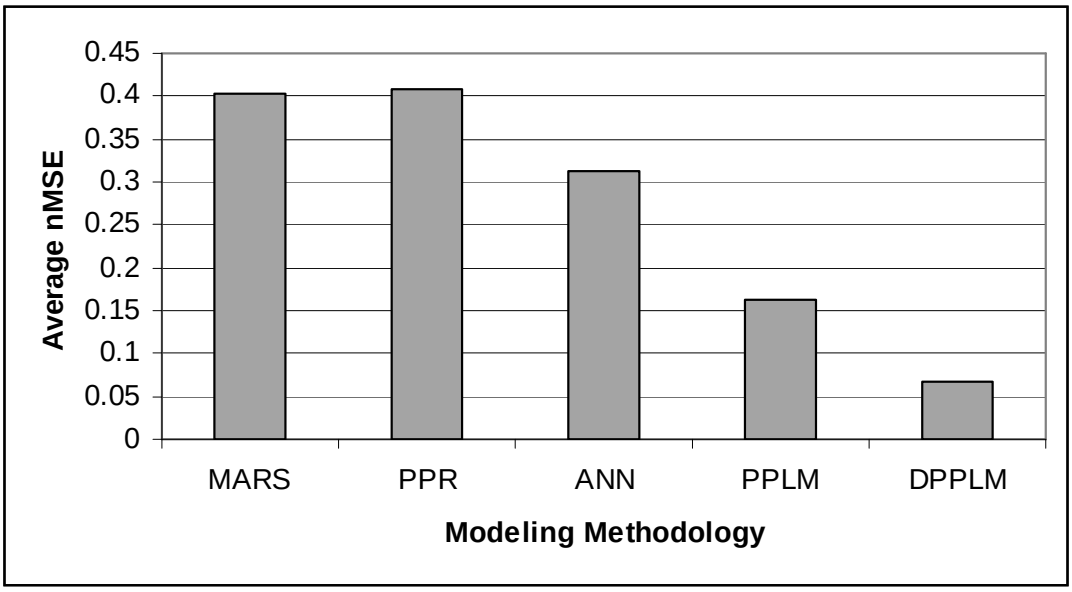

Figure 31: Overall Average Performance By Modeling Methodology

these methodologies. We begin by looking at the noise condition in figures (32) and (33).

On the whole, adjusting the noise amplitude has little effect on performance results. As shown in Figure (33), the average nMSE of the predictions are very similar within each specific model type.

An investigation of the distribution of the noise, figures (34) and (35), shows similar results. All of the modeling methods tested seem to be quite insensitive to noise.

Next, a look at the distribution of the inputs, figures (36) and (37), shows that each of the models are insensitive to changes in the condition as well. The sole exception is the feedforward neural network, which had superior performance when presented with input data that were normally distributed. Still, the relative rank ordering of performance of the models remained the same across input distribution type, with the neural nets outpacing both the MARS and PPR models, and the

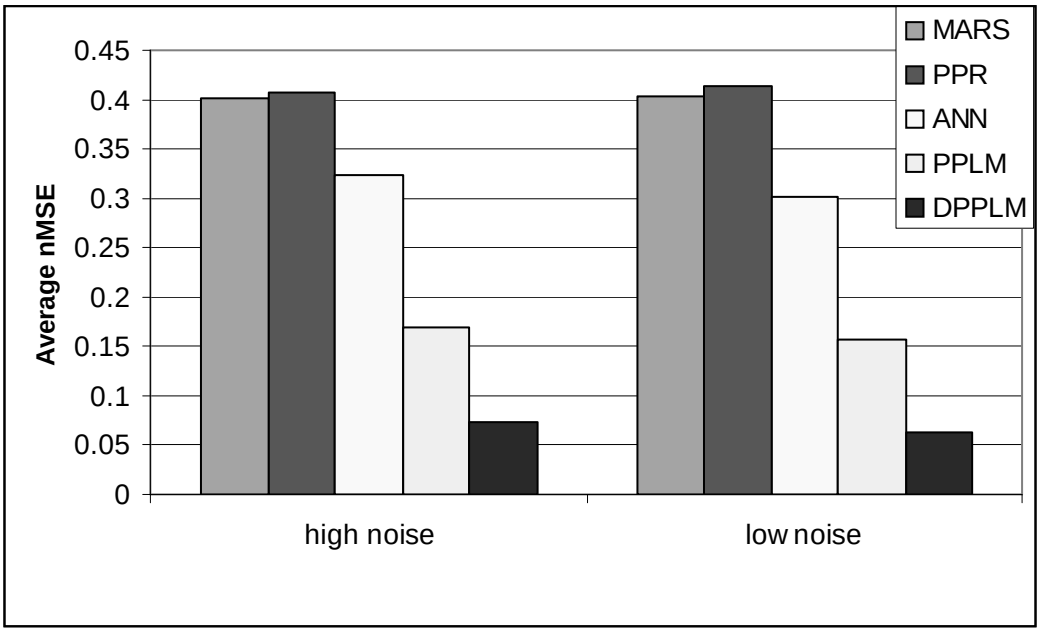

Figure 32: Performance Comparison Within Noise Amplitude Groups

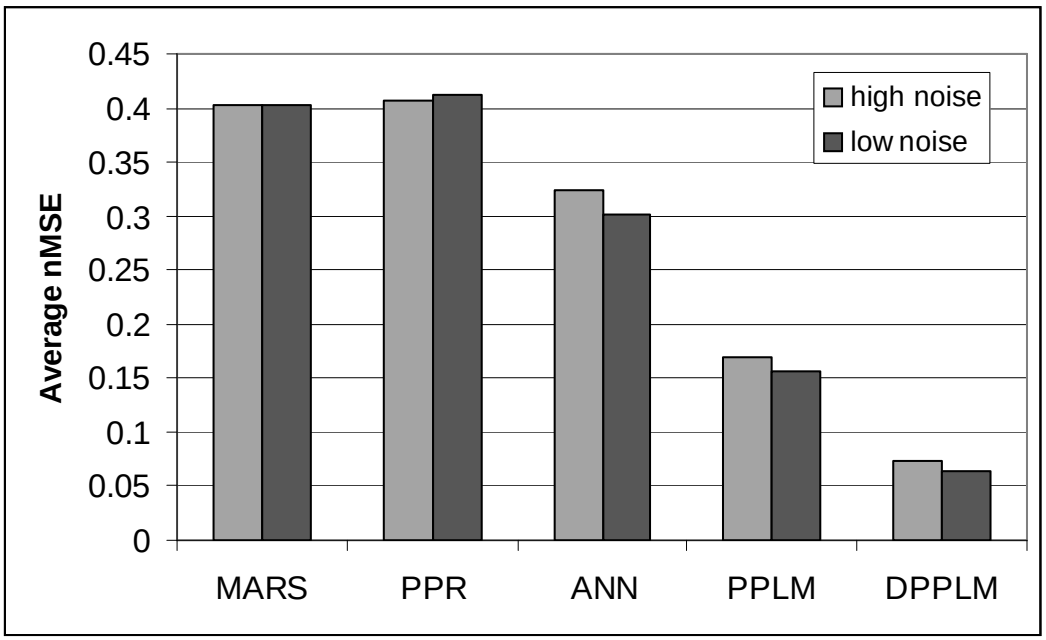

Figure 33: Impact of Noise Amplitude on Model Performance

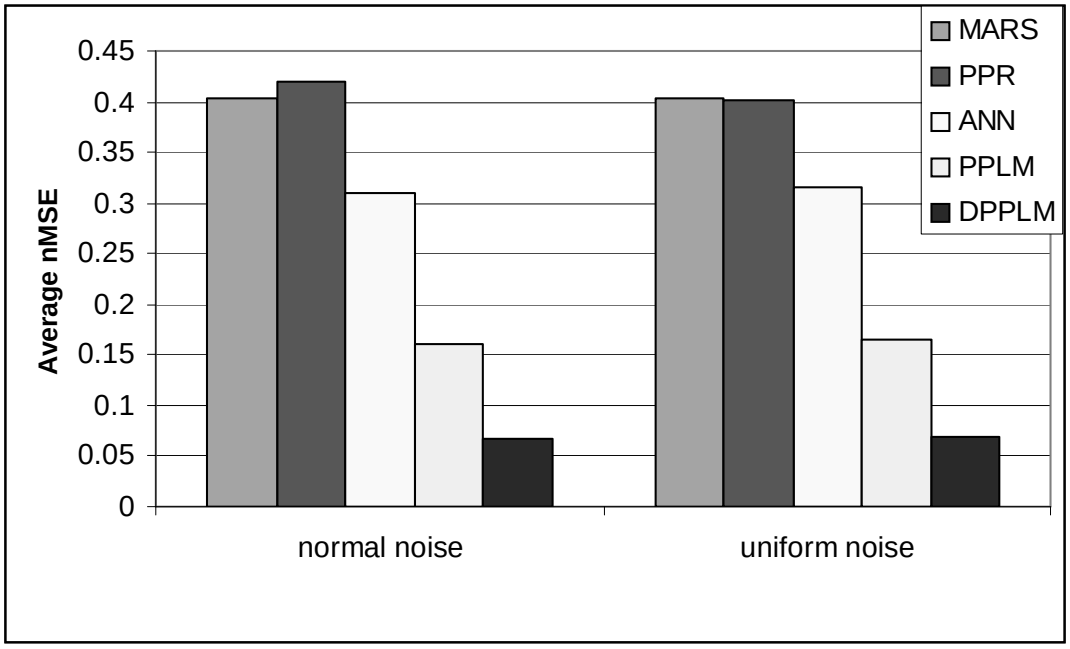

Figure 34: Performance Comparison Within Noise Distribution Groups

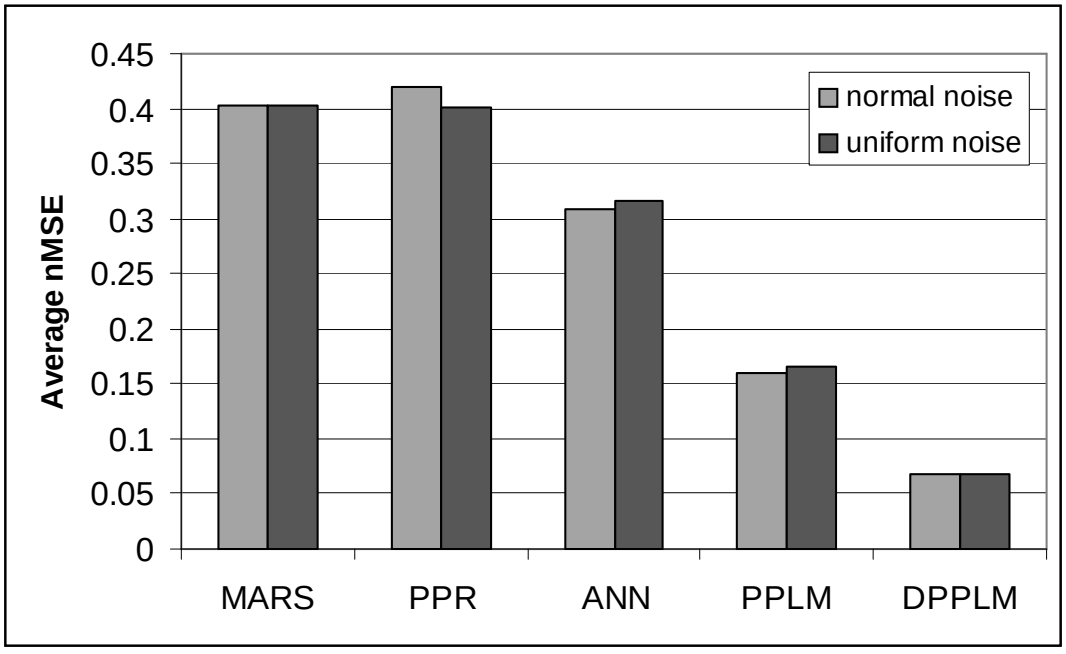

Figure 35: Impact of Noise Distribution on Model Performance

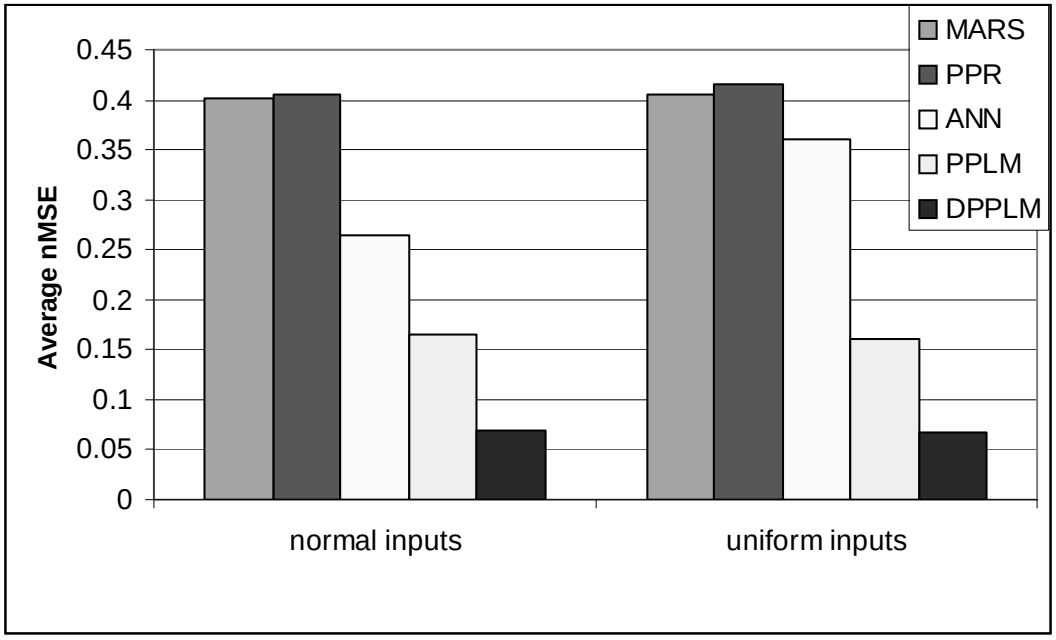

Figure 36: Performance Comparison Within Input Distribution Groups

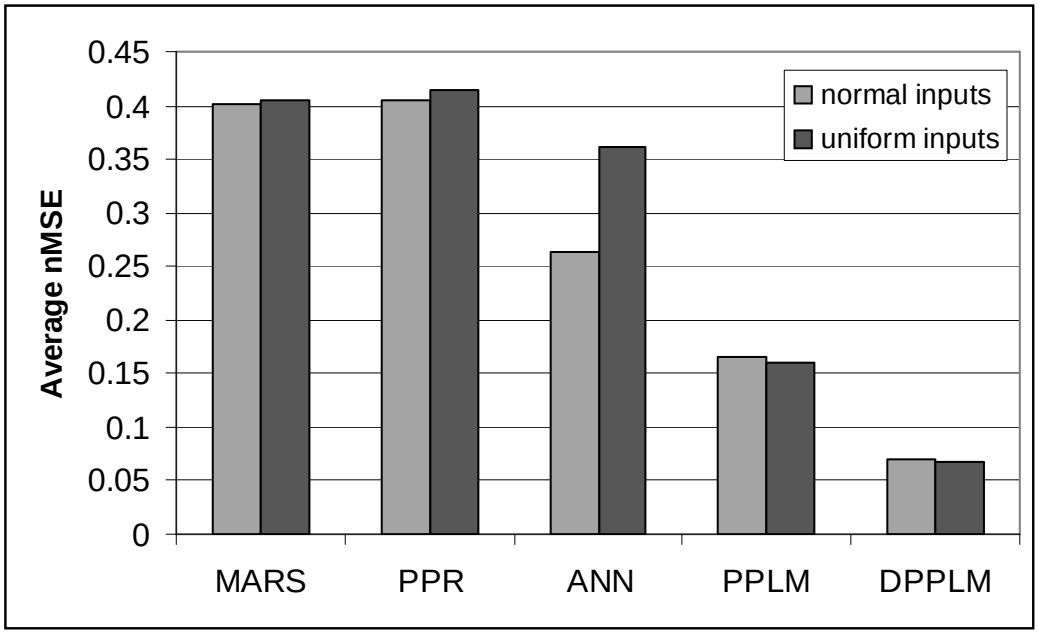

Figure 37: Impact of Input Distribution on Model Performance

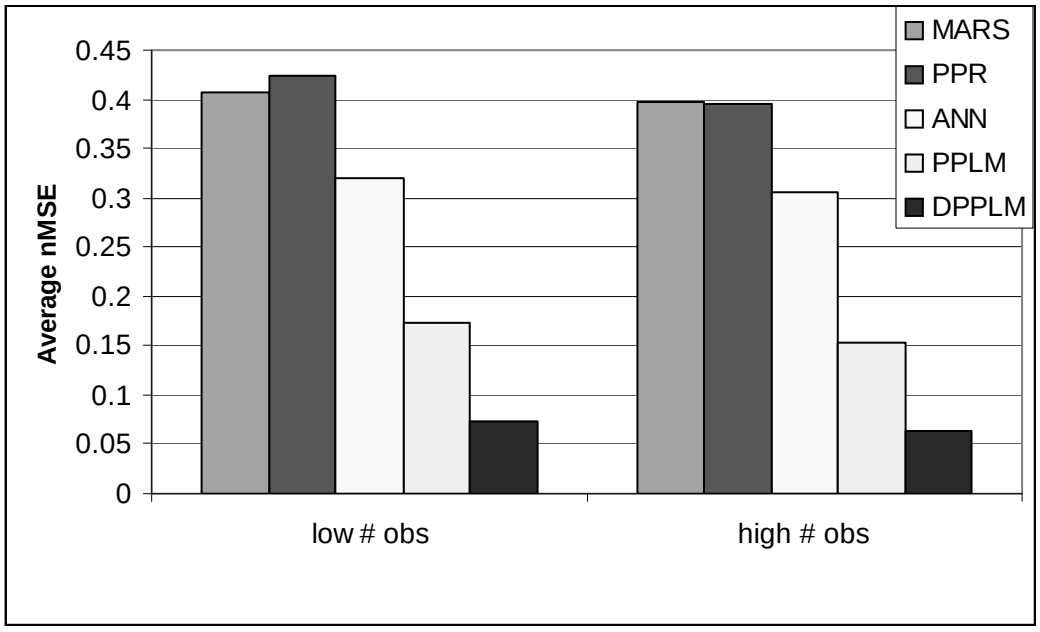

Figure 38: Performance Comparison Within Sample Size Groups

projection pursuit learning methods improving upon these performances.

An investigation of the numbers of observation in figures (38) and (39) reveals a similar trend across all model types: an increase in performance with the higher observation condition ($n = 2,700$). Intuitively, this makes sense – the models performed better when presented with more data.

Turning to the effects of the response surface, figures (40) and (41), the same graph is presented twice. Because of the vast differences in magnitude of performance across response surface type, the graph grouping by model type is left out while the performance chart that groups by response is presented a second time. This is done to re-scale the chart such that we might better be able to discern average performance on the polynomial response functions.

From these performance graphs, the DPPLM performs best across all response function types tested. However, the feedforward neural network modeling approach

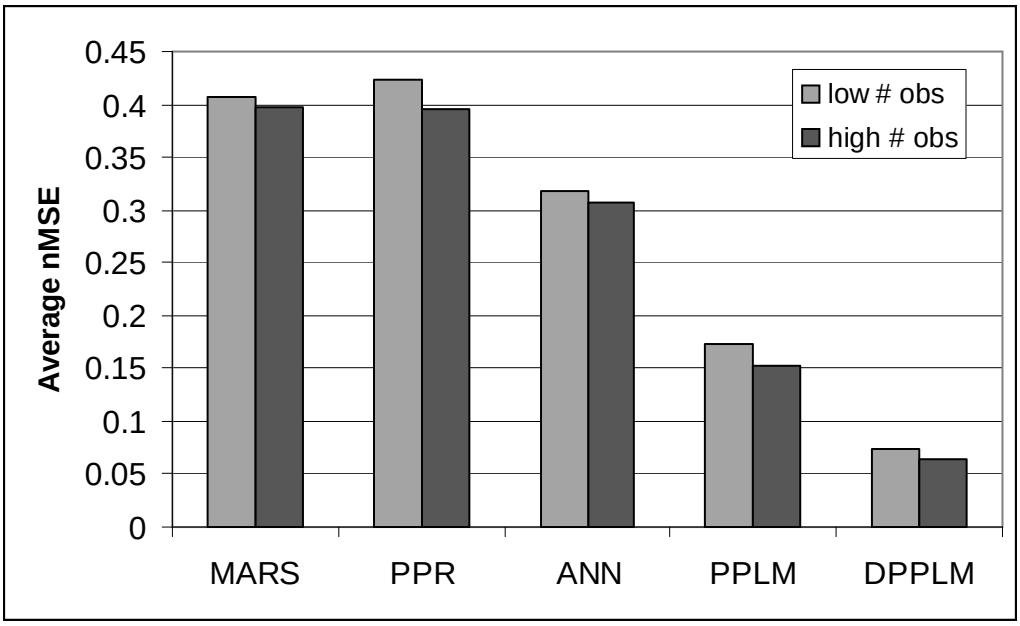

Figure 39: Impact of Sample Size on Model Performance

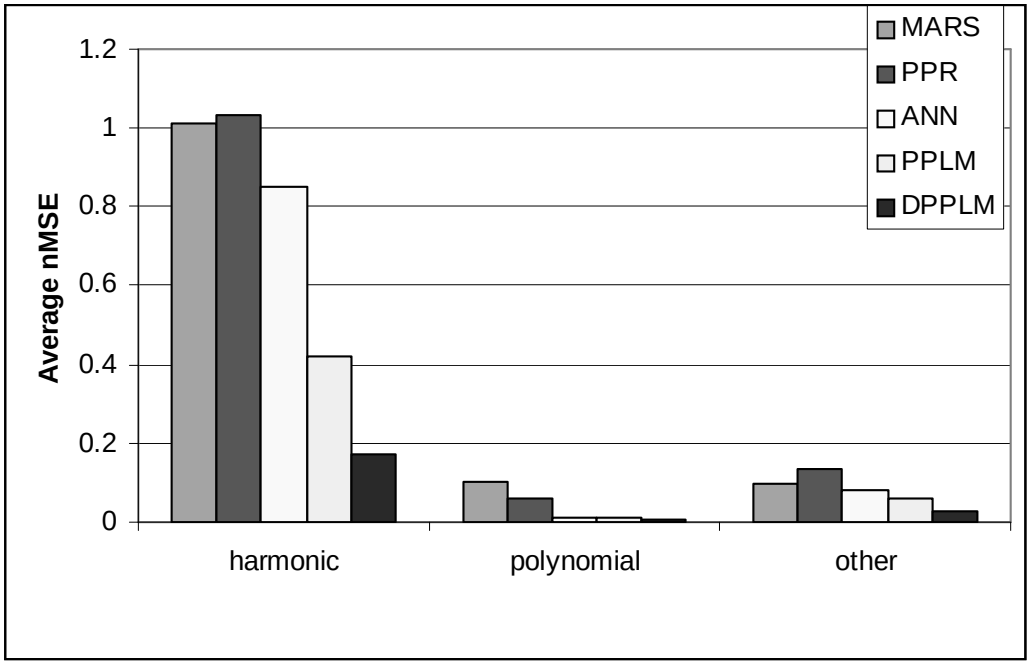

Figure 40: Performance Comparison Within Response Function Groups

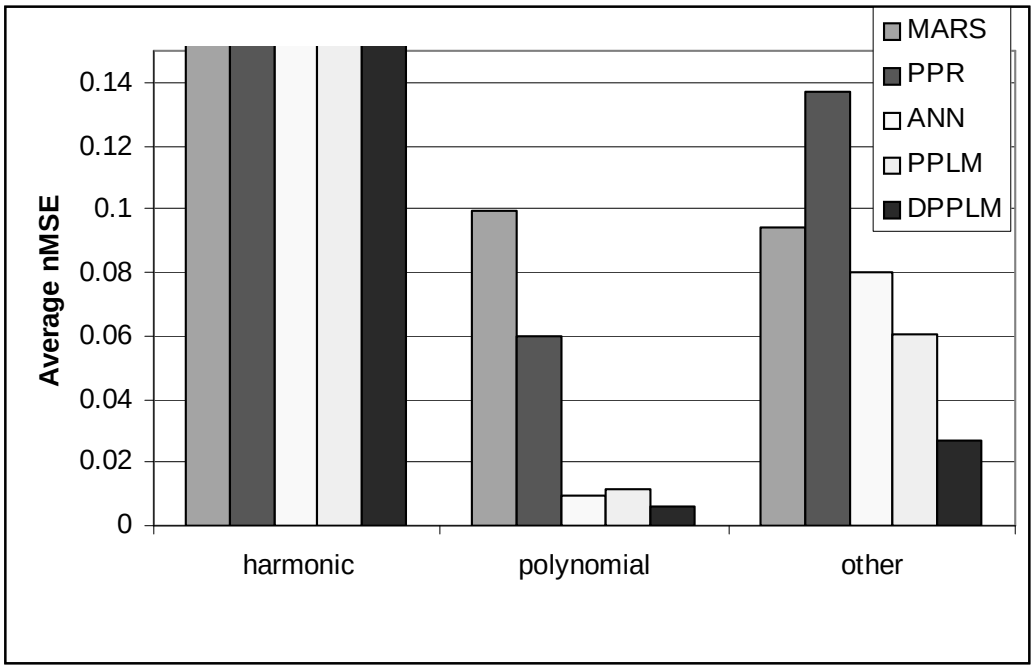

Figure 41: Performance Comparison Within Response Function Groups (Rescaled)

does appear to achieve a slight improvement in results relative to the Levenberg-Marquardt PPLM model. Once again, all three of these approaches produce better predictions on average than PPR or MARS.

Investigating the effects of dimensionality and function complexity on performance reveal dramatic results. Figures (42) and (43) illustrate these results. Figure (43), especially, shows the expected effect of dimensionality: performance suffers with increasing dimensionality. This is what would be anticipated from our earlier investigation into the phenomenon of data sparsity. Perhaps, more surprising is the dramatic improvement in modeling capability of the continuous and discrete PPLM models when dimensions are reduced.

A closer look reveals more insight into this phenomenon. As was done earlier, the same figure was replicated and re-scaledto magnify the effects.

From figure (44) and (45), we see what's going on. While all of the models

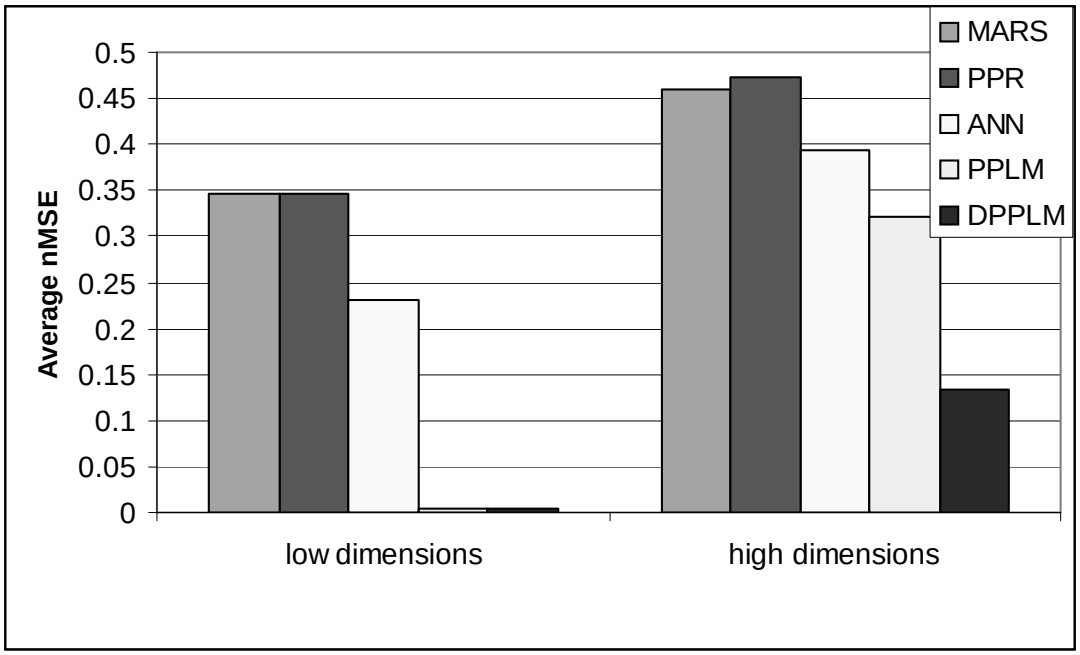

Figure 42: Performance Comparison Within Input Dimensionality Groups

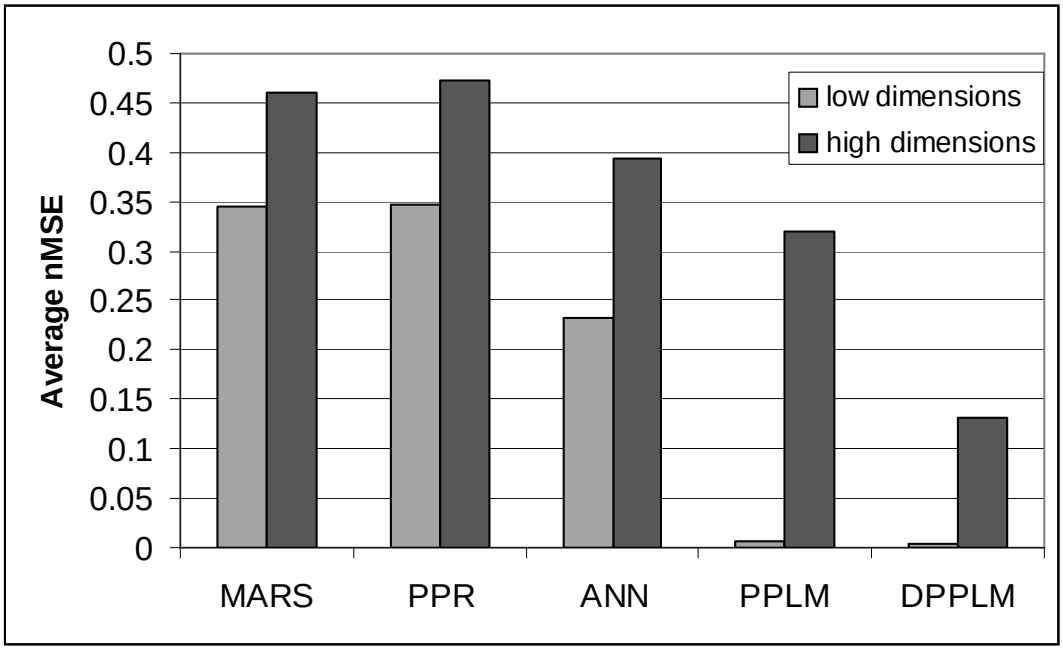

Figure 43: Impact of Dimensionality on Model Performance

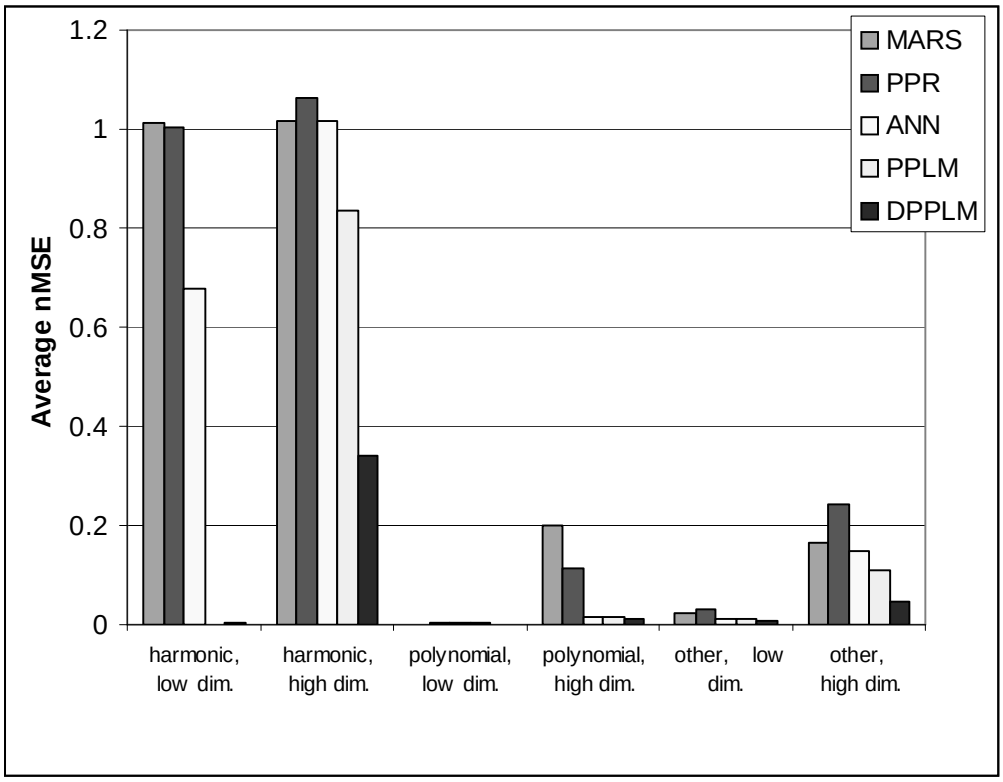

Figure 44: Dual-Level Performance Comparison Within Response Function and Dimensionality Groups

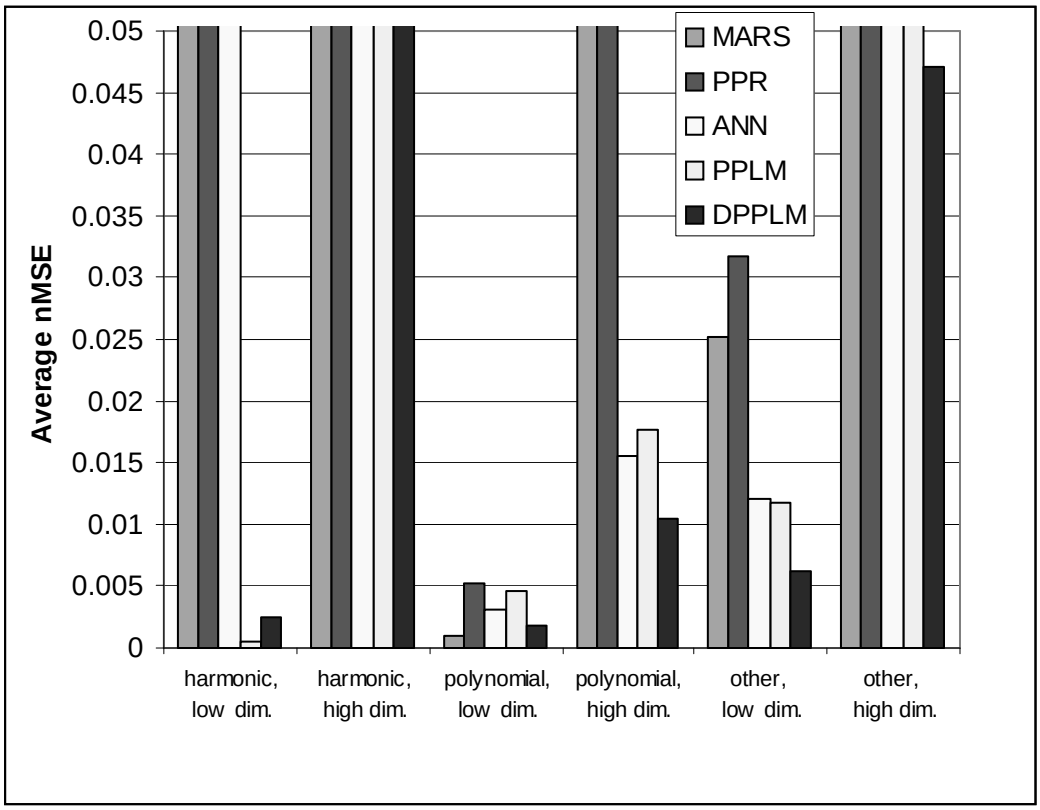

Figure 45: Dual-Level Performance Comparison Within Response Function and Dimensionality Groups (Rescaled)

perform better with reduced dimensionality and reduced response surface complexity, what is clear is that the PPLM and DPPLM models worked exceedingly well in the case of a low-dimensional harmonic response function. Discounting for this, the other modeling approaches actually tended to experience a greater deterioration in performance with increased dimensionality and subsequent model complexity. While this is not evident for the harmonic response function simulations, perhaps this is due to the ceiling effect of the nMSE. Essentially, an nMSE of approximately 1.0 is what would be obtained by predicting the response with only its true mean value. Thus, each of these modeling methodologies, each fully capable of at least predicting the mean value of the response, will tend to generate nMSE values of approximately 1.0 when faced with a response surface they cannot approximate. Thus, the error is capped to an extent. So, we cannot gauge the true deterioration of the MARS and PPR models on predictions of the harmonic response simulation functions when faced with increasing dimensionality because their respective prediction errors were already maxed out in the lower dimensional cases. Likewise, the feedforward neural network's performance degradation cannot be quantified either in the case of harmonic response given that its error was capped in the higher dimensional simulations. Thus, we can only base our conclusions about this dimensionality performance deterioration on the cases of nonharmonic response surfaces. For these cases, it is clear that both the PPLM and DPPLM approaches seem, to some extent, to have mitigated the effects of the curse of dimensionality.

A comparison of computation time for the artificial neural networks, PPLM, and DPPLM models is provided in figure (46) and figure (47). The PPR and MARS models were left off of the comparison because they were run using the R software package. These runs were very quick, however such comparison in CPU time across different platforms would be spurious, at best. Thus, the other three models were

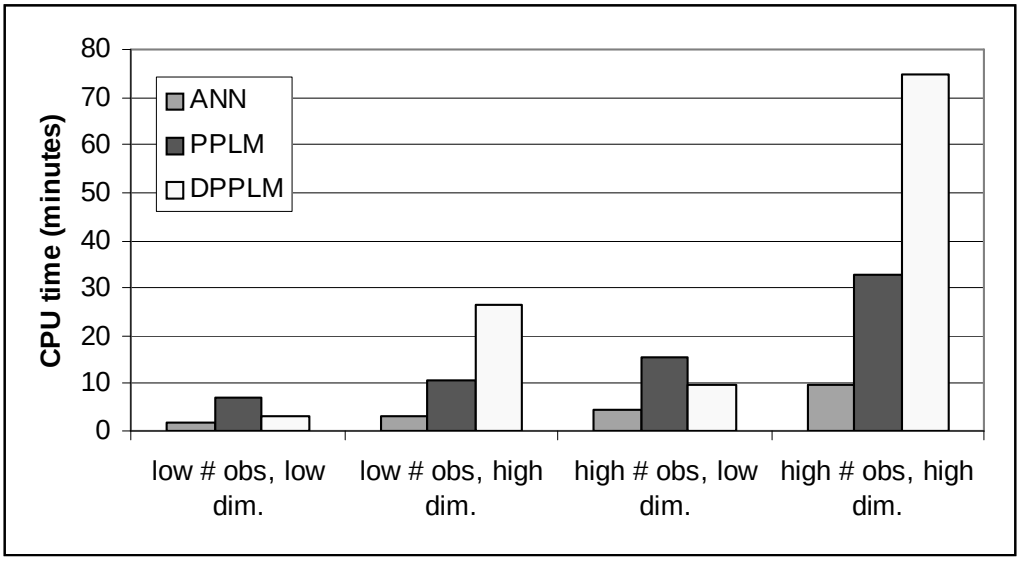

Figure 46: Comparison of Computational Time By Modeling Methodology

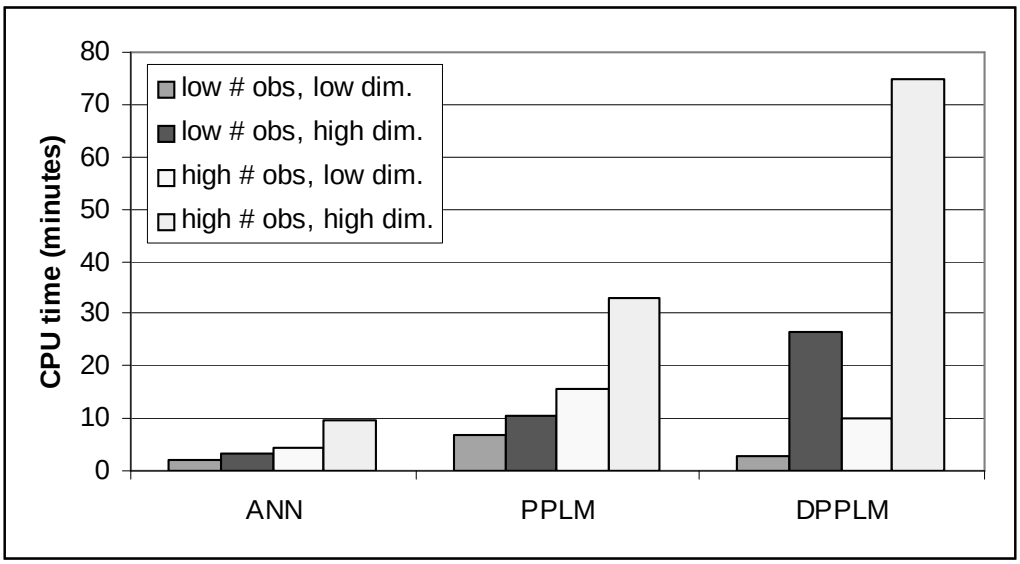

Figure 47: Impact of Sample Size and Dimensionality on CPU Time

compared for computation time, as all three were coded and implemented in Matlab. This comparison shows that the feedforward neural network is computational quicker to train and score than either of the two approaches developed for this dissertation. Part of this is due to inefficiencies in coding. A more computationally efficient algorithm could be coded for the PPLM and DPPLM approaches, however this was not the goal of this research. Yet, a further perusal of the charts show something intriguing. In particular, the DPPLM shows a dramatic increase in processing time with an increase in dimensionality. The culprit here lies with the approach to choosing the initial population of potential discrete projection directions. For the simulations, a low-dimensional exhaustive search of the direction search space was conducted. This was done for greater reproducibility of results and because the dimensionality was low enough to do so without great computational inconveniences. In an even higher-dimensional space, for instance, the direction search routine could easily be switched to a computationally more efficient search method such as a genetic algorithm. This change would bring the computation time in line with the other methods.

As an illustration of this, two of the sets of simulations (the harmonic response simulations) were rerun with a GA search algorithm in place of the exhaustive search method that had been employed for the DPPLM approach. Implementing this genetic algorithm version of DPPLM brings a marked increase in computational efficiency over the original version. Computational times of the GA approach are cut in half (to an average of only 45% of those for the exhaustive search DPPLM) on low dimensional search spaces and are shrunk over 12-fold to just 7.7% of the average time for DPPLM over the 10-dimensional input space problems. In fact, the improvement is so substantial that the GA implementation clocks in at even faster times than the neural nets, as shown in the table that follows:

But an improvement in computation speed is meaningless without the ability to

Table 11: Average CPU Times for Harmonic Response Simulations

Model	**ANN**	**DPPLM(GA)**
low # obs, low dim.	1.5	1.0
low # obs, high dim.	3.5	1.0
high # obs, low dim.	3.9	2.8
high # obs, high dim	10.5	2.8

still achieve good performance. Thus, table (12) is provided in an attempt to help quantify the performance degradation cost of the speedier implementation. Figure (48) also beautifully illustrate these results.

Table 12: Average nMSE on Harmonic Response Simulations

Model	MARS	PPR	ANN	PPLM	DPPLM	DPPLM(GA)
low obs & dim	1.01568	0.98466	0.66199	0.00073	0.00383	0.01112
low obs & dim	1.02546	1.10278	1.02071	0.84502	0.35561	0.59587
high obs & dim	1.00463	1.02025	0.69826	0.00025	0.00113	0.00113
high obs & dim	1.00788	1.02194	1.01297	0.82315	0.32390	0.44004

As can be seen from chart (48) and table (12), performance is comparable to the exhaustive search DPPLM method. Thus, if computational speed is an issue, one could still implement a version of the DPPLM method, such as one with a genetic algorithm directional search scheme, that would provide improvement over a traditional feedforward network.

5.5 Simulation Conclusions

Overall, both the PPLM and DPPLM methods worked quite well when presented with a multitude of varied simulation conditions. From the individual simulation run tables provided earlier in this chapter, it would appear that the discretized parametric projection pursuit method presented in Chapter 4 generally outperformed

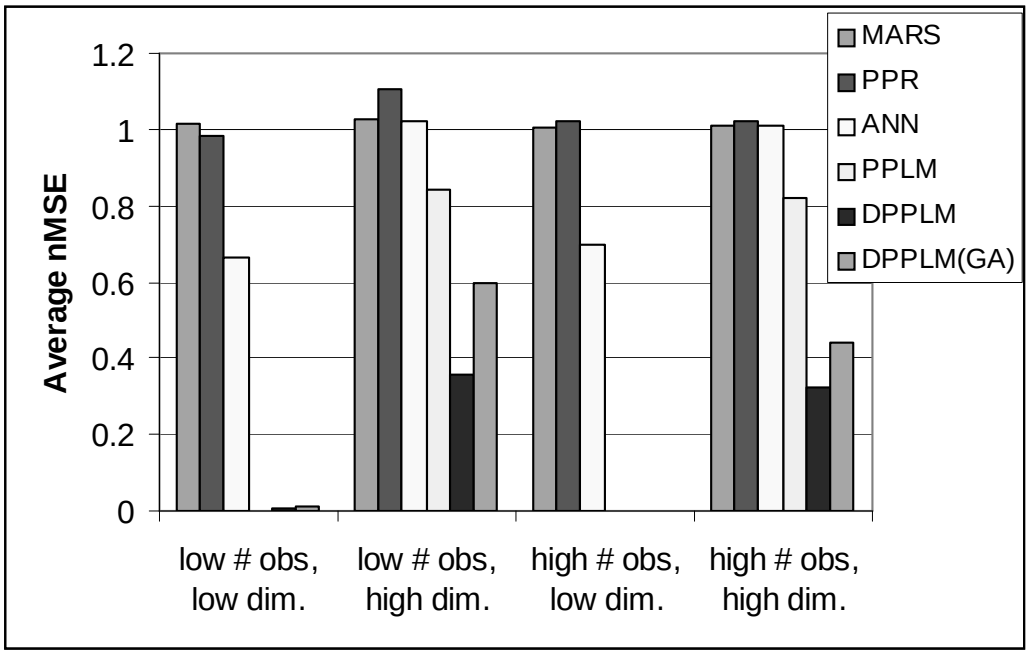

Figure 48: Harmonic Response Performance Comparison of Genetic Algorithm DPPLM with Other Modeling Types

each of the other methods Only in the cases of the low-dimensional harmonic response simulations and the low-dimensional polynomial response was the DPPLM method outpaced by another modeling methodology. In the first of these cases, the iterative continuous PPLM approach described in Chapter 3 slightly bettered the DPPLM in terms of average performance. In the second case, the MARS model performed surprisingly well, achieving a slight error improvement over DPPLM for low-dimensional, polynomial functions. But, overall, both PPLM and DPPLM were quite robust, and suffered only a disadvantage in computation time when compared against the commonly-used high-dimensional modeling methods tested in these simulations. However, certain modifications to the optimization scheme could be implemented to alleviate the lengthy computation time of the DPPLM methodology without suffering greatly in terms of overall performance.

CHAPTER VI

CASE STUDY: STOCK MARKET MODELING

6.1 Background

6.1.1 Overview

Financial markets provide a rich source of data. And indeed, an analysis of such data can lead to some very interesting and useful findings. With this case study, we tap into this abundant source of data. Our approach will be to investigate the efficient market hypothesis (EMH) from the context of our newly-formulated DPPLM approach. According to EMH, stock prices should not be predictable as information is freely and widely disseminated. Thus, new information should be incorporated into current stock prices making the direction of future prices unpredictable. To test this hypothesis, we will build a predictive model to anticipate intraday price changes. The data used in this endeavor is raw 5-minute price and volume data for one security.

6.1.2 Pockets of Predictability

While the efficient market hypothesis in economics might tend to suggest that the market is not predictable, some researchers feel that there are certain regions within an otherwise complicated phenomenon such as the stock market that can be predicted accurately [50]. In other words, the distribution of unpredictability is not uniform throughout systems. Most of the time, most of a complex system may not be forecastable, but some small part of it may be for short times.

David Berreby, writing in the March 1993 issue of Discover magazine [8], puts the search for pockets of predictability in terms of a lovely metaphor: "Looking at

market chaos is like looking at a raging white-water river filled with wildly tossing waves and unpredictably swirling eddies. But suddenly, in one part of the river, you spot a familiar swirl of current, and for the next five or ten seconds you know the direction the water will move in that section of the river." So, while we may not be able to predict where the water will go a half-mile downstream, for a short period of time on a small section of the river, we may be able to forecast such movement

6.2 Experiment Preparation

6.2.1 Data Acquisition

To acquire the data, a PERL script was written that seeks out and downloads the raw variables from the internet. However, during the process of contacting trading firms to work with in partnership, I managed to secure via Joel Lander another method for acquiring the data. A zipped file of archived data consisting of the dates mentioned was emailed to me for use with the analysis. This data was procured by Dr. Lander from an intraday data vendor: Tick Data, a division of Nexa Technologies, Inc.

6.2.2 Potential Modeling Pitfalls:

Before getting into the model-building process, there is a potential problem to address which could lead us to poor results: the issue of omitted variables. When building financial models, variable omission is to be expected. It would be virtually impossible to include all potential variables. Thus, extreme precaution must be taken in the formulation of our model sample and the construction of our explanatory variables to limit the number of omitted variables. To combat this, we shall limit the effects of economic conditions and underlying company fundamentals on our response variable by focusing on a very short outcome period. A 4-hour outcome period has been chosen for this analysis. Furthermore, our data will be comprised of a single security – the semiconductor holders exchange traded fund (SMH). The SMH is an exchange

traded fund (ETF) designed to track the price movements of 20 semiconductor stocks. So, it is very similar to a stock index, but it is also tradeable. With this focus on a security that averages the price performance of many stocks within a single sector, we are able to alleviate many of the omitted variables concerned addressed earlier.

6.3 Modeling Procedures

For the intraday stock market modeling experiments, The input variables consist of the raw price and volume information with some simple transformations being applied to these data. Unfortunately, the specific variables used cannot be disclosed as they remain the proprietary possession of the investment firms which have so graciously lent their assistance and expertise for this academic investigation. For all of the experiments run on this data, the data consisted of 5-minutes intraday data collected on one security (SMH) over a $2\frac{1}{2}$ year period (6/3/2002-1/31/2005). From this data, the training, validation, and test samples were constructed as follows:

training sample = (6/3/2002-10/17/2003)

validation sample = (10/20/2003-5/24/2004)

testing sample = (5/25/2004-1/31/2005)

The response variable is:

Y = The % change in the stock price over the next 4 trading hours

For this case study, the feedforward artificial neural networks is compared with the DPPLM approach (this time, also allowing for polynomial bases as the choice of basis functions). As each of the models have a number of varying parameters to be set, the validation sample is again used as a data-driven selection method for choosing the appropriate parameters. Each time the experiment was run, the training and cross-validation samples are held constant for all of the methods tested in order to give a fair comparison of the results. Once again, the best neural network and the DPPLM with parameters optimized by this validation sample were thus chosen and

then applied to the holdout (test) sample. This process should provide an accurate reflection of the prediction capabilities of the compared modeling methodologies.

6.4 Results

A summary of the results on the test sample is provided in table (13).

Table 13: nMSE on Test Sample

ANN	DPPLM
0.9926	0.9791

From these results, it is obvious that neither model can predict this sample of stock market data with impeccable accuracy. But, this is to be expected. The stock market is notorious for being highly unpredictable. However, looking at this data in another way, as $1-nMSE$, we can gauge how much of the variability in the response is effectively forecasted by each method. From this, we see that the feedforward neural nets can only predict a fraction of a percent of this variability, whereas DPPLM is capable of forecasting over 2% of the variability. Still, this nMSE metric that we have used so effectively as a performance comparison for past simulations may not be the best comparison tool for this sample given its very limited degree of predictability.

But the question still remains: obviously, predicting on this sample is very difficult, but how effective are these results? Is this apparently slight improvement in predictive power really that significant? To answer this question, we have devised an interesting method to test the effectiveness of the predictions for this data.

6.5 Trading Strategy - a model implementation

In practice, the implementation of such a model would not be confined to just looking at nMSE results. A trading strategy built around the model predictions would be constructed in order to implement the results on the actual stock data.

Thus, we have set up a simple trading strategy as follows:

Trade Initiations:

If $score < -0.1$,	then short sell
elseif $score > 0.1$,	then buy
else	do not initiate trade

Exit Criteria:

1.) Exit initiated trades 4 hours after time of initiation, unless there is a new initiation triggered.

2.) On New Initiation trigger

a.) If new initiation triggered in opposite direction, then exit trade and follow new signal.

b.) If new initiation triggered in same direction, then maintain position but use this new pseudo-initiation for initiation time.

The reasoning behind the trading strategy is as follows. Due to commissions and slippage (caused by bid-ask spread), a score cut-off is used for initiations. This enables the investor to only put on trades that would overcome the costs of trading. The exit time of 4 hours corresponds to the outcome variable used by the model. The modifications to the exit strategy are implemented because the strategy is trading at full-exposure on a first initiation.

6.6 Strategy Simulations

Using the trading strategy outlined in the prior section, both the results of the neural network and those of the DPPLM were then simulated on the test sample. An illustration of these results is provided in figures (49) and (50) for the ANN and DPPLM approaches, respectively.

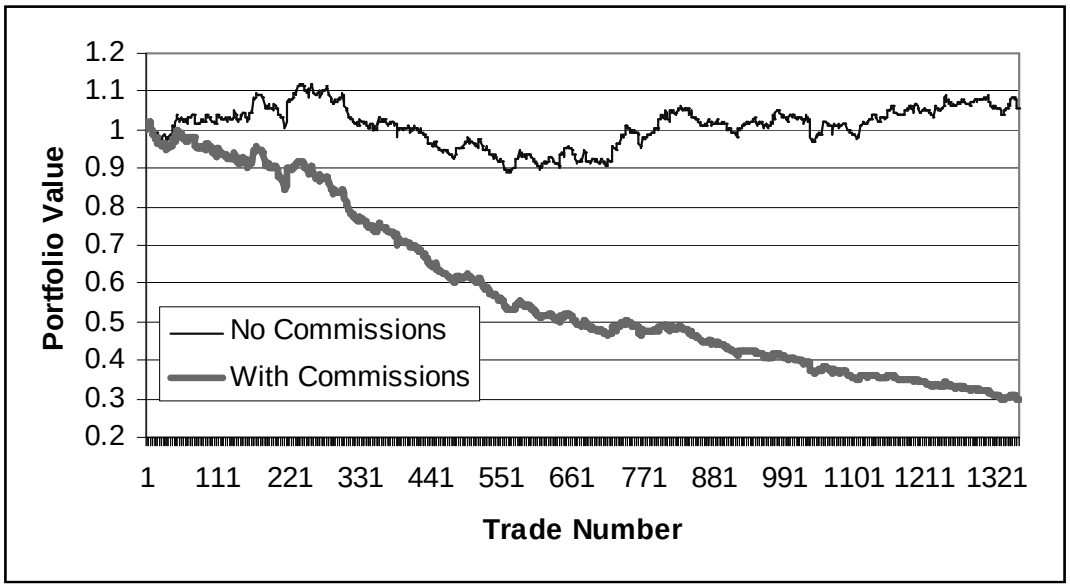

Figure 49: Simulated returns using trading strategy based on ANN model predictions (05/25/04-01/31/05)

There are a couple of items to note with these graphs. The first is the x-axis. Results are plotted by each initiated trade. The time period of the test sample is identical for both of the graphs. So, it stands out that the ANN is initiating quite a few more trades than is the DPPLM. For the DPPLM approach, the average trade duration is 7 hours, or just over 1 trading day. But, the trade duration can vary quite a bit – the longest over this test period is about 5.5 days, while the shortest is 15 minutes. With the ANN model, the average trade duration is only 50 minutes, with some trades lasting as long as a couple of days with other lasting only 5 minutes. However, the same score cutoff of ± 0.1 was used as the criterion for initiating trades.

Another point to note is that each figure consists of the graph of trading strategy with and without estimated commissions. For this graph, commissions and slippage have been estimated to have a combined effect of 1.5 cents per share for each trade. Note that in practice, this is a highly achievable and quite realistic estimate of commissions rates given a reasonable amount of trading capital is dedicated to the strategy.

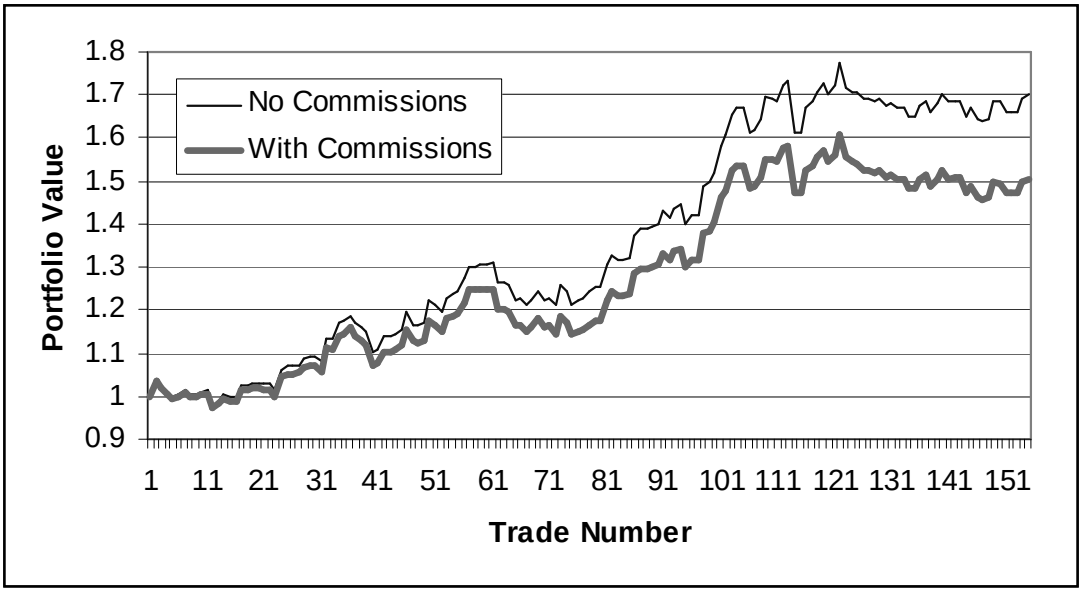

Figure 50: Simulated returns using trading strategy based on DPPLM predictions (05/25/04-01/31/05)

Without commissions, we see from the graphs, that both models can be implemented profitability. However, when taking commissions into account, only the DPPLM is profitable. In fact, commissions take a much larger toll on the ANN approach since there are nearly ten times the number of trades being initiated with that approach over the same simulation time period. While these results seem quite encouraging as a measure of the of the DPPLM, still, a more extensive investigation is in order.

6.7 Comprehensive Statistical Investigation

For this investigation, we will perform a number of statistical tests in an effort to gauge how effective the DPPLM approach was at making predictions on this data. To do this, we will first pose a series of question, each in turn, to be answered by a specific statistical test designed to address that question.

6.7.1 Statistical Test #1

Our first question is: "Are the DDPLM trading results just the result of lucky guesses?"

To address this question, a bootstrap of 10,000 replications (w/o replacement) was run given our distribution of long/short/neutral signals over the specified period.

H_0 : The returns achieved by our trading strategy are the result of random guesses

H_1 : The trading strategy returns are statistically higher than the mean returns given the distribution of signal directions

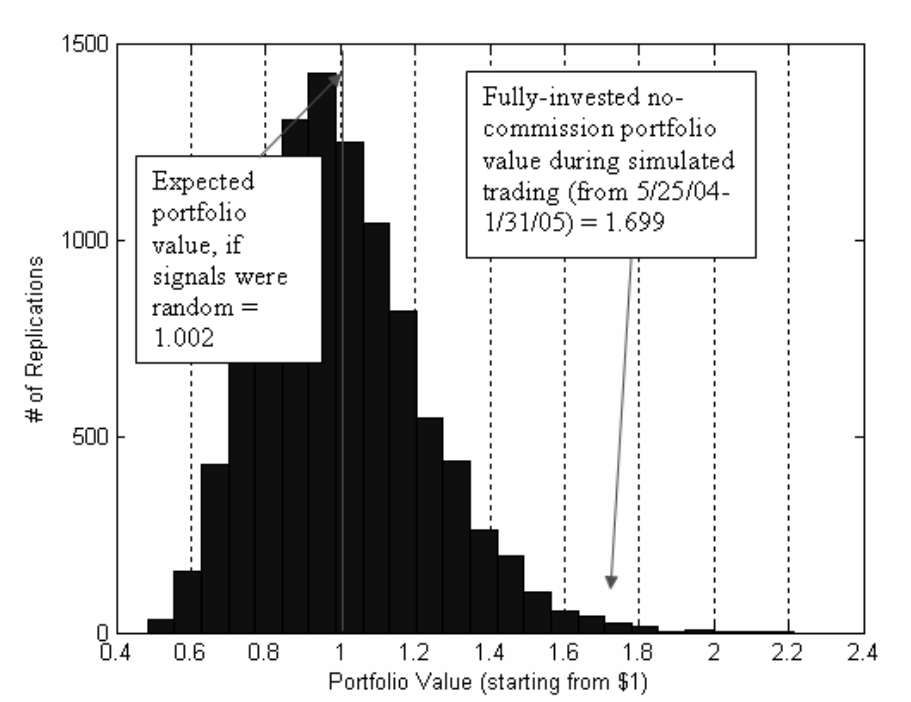

Figure 51: Bootstrap of DPPLM trading results on test sample

Employing the bootstrap test yielded a mean return of 0.21% over the period. we see from figure (51) that the returns achieved over this period are statistically significantly higher than the mean returns at the 5% significance level. In fact, only 57 of the 10,000 replications would have yielded a return equal to or higher than those

achieved by the trading strategy over the specified period. Thus, we reject the null hypothesis.

6.7.2 Statistical Test #2

Is this trading strategy capable of picking the direction of the market more accurately than a random guess?

Over the simulated period, the trading strategy correctly identifies the direction of the intraday move in the SMH with the majority of its signals.

Table 14: Classification of accuracy of DPPLM trading strategy signal directions

Pred. Direction	**# of Trades**	**Avg. Return**
Correct	84	1.52%
Incorrect	48	-1.50%
Neutral	1	(N/A)

The Upper-Tailed Sign Test, which is a nonparametric statistical test, was used to determine if the increased frequency of positive return days is statistically significant.

$$H_0 : P(correct\_signal\_direction) \leq P(incorrect\_signal\_direction)$$
$$H_1 : P(correct\_signal\_direction) > P(incorrect\_signal\_direction)$$

The significance level used for this test was $\alpha = 0.01$. So,

$$t = \frac{1}{2}\left(132 - 2.3263\sqrt{132}\right) = 52.64.$$

Since, $n - t = 79.36$, and $T = 84$, where T is the number of positive signals, then we reject the null hypothesis at a 1% significance level (99% confidence level) as $T > n - t$. Therefore, the Sign Test suggests that the trading strategy is capable of correctly predicting the direction of the SMH more frequently than not.

6.7.3 Statistical Test #3

Is the magnitude of the trading strategy score correlated with actual SMH returns?

First, we state our null and alternate hypotheses in an attempt to address this question.

H_0 : SMH signal returns are independent of trading strategy model scores

H_1 : SMH signal returns are correlated with trading strategy model scores (i.e., high predicted scores tend to lead to higher price moves)

Now, to answer this question, we can employ the Spearman's Rho test. This is a nonparametric test, selected specifically so as not to depend on an implied distribution.

Spearman's rho, is computed by:

$$\rho = \frac{\sum_{i=1}^{n} R(X_i) R(Y_i) - n\left(\frac{n+1}{2}\right)^2}{\sqrt{\sum_{i=1}^{n} R(X_i)^2 - n\left(\frac{n+1}{2}\right)^2}\sqrt{\sum_{i=1}^{n} R(Y_i)^2 - n\left(\frac{n+1}{2}\right)^2}} = 0.1320.$$

The overall significance then is written as:

$$p - value = P\left(Z \geq \rho\sqrt{n-1}\right) = P\left(Z \geq 0.1320\sqrt{132-1}\right) << 0.01$$

Thus, we easily reject the null hypothesis at the 1% significance level.

6.8 Summary

Even without running any of these statistical tests, we can intuitively see the significance of the DPPLM model results with a graph of model predictions versus actual returns, as is provided in figure (52):

Thus, this case study gives us further affirmation of the effectiveness of the DP-PLM technique to address a wide array of modeling problems.

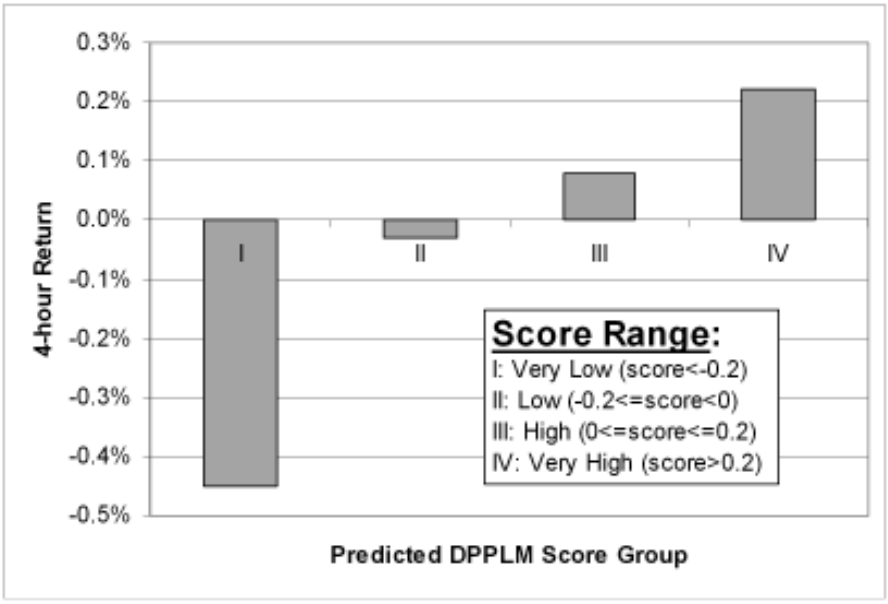

Figure 52: DPPLM Predictions vs. Actuals Quartile Plot

CHAPTER VII

CONCLUSIONS

This thesis presented two new learning algorithms, PPLM and DPPLM: both nonlinear function approximation models that are particularly well-suited for high- dimensional nonlinear datasets. Theorems were presented that established the methods' mathematical foundations, and proofs were detailed to provide insight into the both of the methods' approximation capabilities. The essence of these novel approaches is to approximate functions with the superposition of a series of piecewise one-dimensional models that are fit to specific projection directions. The key to their effectiveness lies in their ability to find efficient projections for reducing the dimensionality of the input space to best fit the underlying response surface. Moreover, these methods are capable of effectively selecting appropriate projections from the input data in the presence of relatively high levels of noise. For illustration purposes, this paper demonstrated how PPLM leads to excellent function approximation results on two simulated datasets, each exhibiting very different characteristics.

This was accomplished by formulating algorithms that rigorously adhere to the theoretical conditions of approximating the solution space, taking full advantage of the principles of optimization for maximizing the efficiency of the algorithms, deriving the theory of function construction from a series of low-dimensional projections, developing a new universal approximation theorem for each of the methods to prove their convergence, and constructing such algorithms capable of hedging against the curse of dimensionality.

7.1 Contributions

The contributions of this work in the realm of nonlinear modeling and function approximation can be outlined as follows:

1. In Chapter 2, the "curse of dimensionality" was rigorously examined to derive some theoretical results showing the properties of data distributions in high-dimensional space.

2. In Chapter 3, the projection pursuit learning model (PPLM) was fully developed from theoretical underpinning to the construction of the algorithm

   a.) The theory was derived to show that a function $f \in \mathcal{F}_n$ can be decomposed into an *infinite* number of single variable, mutually orthogonal functions. This forms the basis for the development of a class of projection pursuit learning models capable of approximating high-dimensional functions by the construction of lower-dimensional projections.

   b.) As an extension of this, a theorem for the universal approximation capabilities of this approach to function approximation is rigorously derived. This theorem provides an enhancement over existing universal approximation theorems in the field today as it extends the approximation abilities to a wider class of functions.

   c.) Based on the prior theory, a theorem for optimizing the selection of optimal projections is developed. This is central to the algorithms in this thesis.

   d.) An algorithm is constructed that employs all of the foundational theory of the chapter to effectively model high-dimensional response surfaces

3. Given that many real-world response surface may be approximated by functions of low-degrees of coupling, Chapter 4 presented the discretized projection

pursuit learning model (DPPLM). The theory behind this method and the subsequent implementation of the algorithm are provided. Also, given the unique properties of this discrete projection search space, a more efficient method of optimization is described.

a.) The theory was derived to show a continuous function $f \in \mathcal{L}_2(\mathcal{D})$ can be decomposed into an *infinite* number of single variable, mutually orthogonal functions.

b.) A universal approximation theorem is derived to prove the approximation capabilities of this approach to function approximation.

c.) A theorem for optimizing the selection of optimal projections is developed that has its basis in the prior theory.

d.) The culmination of the foundational theory of the chapter is an algorithm that effectively incorporates this theory into a method of function approximation capable of mitigating the effects of data sparsity.

e.) Another possible construction of the algorithm based on a GA projection direction search routine is suggested that could achieve similar approximation results with substantial computational savings.

4. Drudging through an extensive series of simulations in Chapter 5 and the case study of Chapter 6, it is shown that the algorithms presented in the thesis are quite capable of approximating a wide array of response surfaces under various sample conditions.

7.2 Major Creative Contributions

The major creative contributions are as delineated below.

1. A theorem for the universal approximation capabilities of the continuous PPLM

approach to function approximation is rigorously derived. This theorem provides an enhancement over existing universal approximation theorems in the field today as it extends the approximation abilities to a wider class of functions. Specifically, the class of functions for which the universal approximation theorem holds has been expanded to include unbounded, $\mathcal{L}_p$ space. In the literature, universal approximaton theorems, [38], [39], [54], are proven for functions restricted to the class of bounded, $\mathcal{L}_p$ functions or are even more restrictive, such as is the case of [16], which is confined to $\mathcal{L}_2$ space.

2. The structure of the optimization procedure is an expansion upon the current literature for both of the learning models presented. Specifically, neural network algorithms focus on solving for both the magnitude and direction of their intended projections with an iterative procedure. In the approaches described in this thesis, the 1-dimensional basis functions are derived in closed-form. Thus, the optimization can focus its reseources on only searching for optimal projection directions.

3. The DPPLM approach offers another considerable creative contribution to optimization that is not found elsewhere in the literature. Because of the constraints on what the algorithm is trying to optimize (only the directions, and not also the magnitude) and because of the discretization of these projections, the set of possible projection directions is finite and countable. Therefore, a new set of optimization routines can be used to search this space of potential projections. In this thesis, an exhaustive search and a genetic algorithm approach are employed as possible optimization procedures. While genetic algorithms have previously been used to optimize specific network topology parameters, such as the network size, nowhere in the literature have we found a genetic algorithm used for optimizing projection directions. In fact, as previously constructed,

this problem would be intractable to solve with a GA given the infinite possibilities of projections. Thus, constructing the problem in such a way that a random search technique can be used to search for projection directions is a significant contribution, particularly considering that the problem of finding the optimal projection directions is key in projection pursuit function approximation.

4. Another significant contribution of this thesis is the comprehensive comparison study of these new methods with the major high-dimensional function approximation methodologies in use today. While a considerable number of other resources [15], [20], [21], [22], [64], [84], [92], [93], [95] have previously presented comparison studies of various different methodologies, we have found no study as comprehensive in its investigation of the various attributes of the feature space.

7.3 Impact of Work

1. The resulting algorithms constructed in this work have wide-ranging applications, each as a methods for improving prediction capabilities for problems in high-dimensional space (problems with dozens of inputs). These methods can easily be applied in such diverse fields as machine vision, speech recognition, motor control, machine learning, financial prediction, economic forecasting, and a range of other engineering and data mining applications.

2. The framing of the optimization problem into a discrete problem of finding optimal projections from a choice of finitely many possibilities is a substantial contribution that has a rather profound impact on the high-dimensional function approximation problem. Because of this new framework, a whole new class of optimization procedures is now available for use in optimizing projection directions. The problem of finding the optimal projection directions has been

stated to be the most difficult problem in function approximation for projection pursuit methods [32], [83], and [84]. Thus, it is possible that this new framework will spark a new area of research focused on finding the best discrete projection direction optimization techniques.

7.4 Recommendations for Future Work

This research suggest a number of exciting future directions in both the theoretical and implementation domains:

1. The methods could be extended into the realm of classification problems.

2. The convergence rates of the PPLM and DPPLM methods could be derived to prove their computational advantages over other nonlinear high-dimensional modeling techniques.

3. From a theoretical standpoint, it would appear that the continuous PPLM methodology should be able to achieve better performance than that of DPPLM. In practice, the opposite was shown to be the case. Thus if an improved method of identifying the optimal directions could be found, than this method might prove most effective for a wide array of high-dimensional prediction problems.

4. Algorithmic refinements to the Matlab code (or implementation in a lower-level computing language) could improve the computational efficiency of both the PPLM and DPPLM models.

5. The parameters and structure of the genetic algorithm approach to DPPLM projection selection could be subjected to a comprehensive investigation. Such an undertaking would likely yield a highly computationally efficient model that retains the approximation capabilities of the original DPPLM presented and would enable the application of the method to be extended considerably, up to even higher dimensional datasets than is presently possible with this method.

APPENDIX A

ADDITIONAL INFORMATION

A.1 Appendix: Matrix Calculus

The first set of notations we introduce are the stacking operation and Kronecker product used extensively in matrix calculus [11], [56].

Given a matrix $A \in \Re^{m \times n}$, its stack, denoted by A^s, is the column-wise vectorization of A:

$$A = \begin{bmatrix} A_{11} & \cdots & A_{1n} \\ \vdots & \ddots & \vdots \\ A_{n1} & \cdots & A_{nn} \end{bmatrix}, \text{ Then } A^s = \begin{bmatrix} A_{11} \\ \vdots \\ A_{1n} \\ \vdots \\ A_{n1} \\ \vdots \\ A_{nn} \end{bmatrix}$$

The Kronecker product of two matrices $A \in \Re^{m \times n}$ and $B = \Re^{p \times q}$ is

$$C = A \otimes B = \begin{bmatrix} A_{11}B & \cdots & A_{1n}B \\ \vdots & \ddots & \vdots \\ A_{n1}B & \cdots & A_{nn}B \end{bmatrix}$$

A useful identity [56] used later in our matrix calculus is

$$(ABC)^s = \left(C^T \otimes A\right) B^s \tag{28}$$

Next we define the Jacobian of vector and matrix valued functions. For a smooth vector–valued function $f : \Re^n \rightarrow \Re^m$, Df is the $m \times n$ Jacobian matrix of $f(x)$:

$$Df = \begin{bmatrix} \frac{\partial f}{\partial x_1} & \cdots & \frac{\partial f}{\partial x_n} \end{bmatrix}$$

The differential of $f(x)$ corresponding to $\Delta x \in \Re^n$ is defined to be

$$\Delta f(x, \Delta x) = f(x + \Delta x) - f(x)$$

The Jacobian of a matrix valued function, $A(x) \in \Re^{m \times n}$is defined to be the Jacobian of its vectorization: $DA(x) := DA^s(x)$.

The following lemma extends the differentiation product rule to matrix multiplication:

Lemma 25 *Given smooth matrix valued functions $A(x) \in \Re^{m \times n}$ and $B(x) \in \Re^{n \times p}$ we have*

$$D(AB) = \left(B^T \otimes I\right) DA + (I \otimes A) DB$$

Proof. Let $F = AB$ and let x be perturbed by Δx. Then,

$$F + \Delta F = (A + \Delta A)(B + \Delta B) = AB + A\Delta B + \Delta AB + \Delta A \Delta B.$$

Thus,

$$(A + \Delta A)(B + \Delta B) - AB = A\Delta B + \Delta AB + O\left(\left\|\Delta x^2\right\|\right)$$

So, $\Delta F = A\Delta BI + I\Delta AB + O\left(\|\Delta x^2\|\right)$. Vectorizing both sides using identity (28), we have

$$\Delta F^s = (I \otimes A)\Delta B^s + \left(B^T \otimes I\right)\Delta A^s + O\left(\left\|\Delta x^2\right\|\right)^s$$

Taking the limit of both sides as $\Delta x \to 0$ yields

$$DF = (I \otimes A) DB + \left(B^T \otimes I\right) DA$$

■

A.2 Bearing Experiment Data

Width (um)	Height (um)	Speed (RPM)	Load (PSI)	Replication #
0	0	800	200	1
0	0	800	200	2
0	0	800	400	1
0	0	800	400	2
0	0	800	600	1
0	0	800	600	2
0	0	1200	200	1
0	0	1200	200	2
0	0	1200	400	1
0	0	1200	400	2
0	0	1200	600	1
0	0	1200	600	2
0	0	1600	200	1
0	0	1600	200	2
0	0	1600	400	1
0	0	1600	400	2
0	0	1600	600	1
0	0	1600	600	2
35.33	2.46	800	200	1
35.33	2.46	800	200	2
35.33	2.46	800	400	1
35.33	2.46	800	400	2
35.33	2.46	800	600	1
35.33	2.46	800	600	2
35.33	2.46	1200	200	1
35.33	2.46	1200	200	2
35.33	2.46	1200	400	1
35.33	2.46	1200	400	2
35.33	2.46	1200	600	1
35.33	2.46	1200	600	2
35.33	2.46	1600	200	1
35.33	2.46	1600	200	2
35.33	2.46	1600	400	1
35.33	2.46	1600	400	2
35.33	2.46	1600	600	1
35.33	2.46	1600	600	2
37.67	10.56	800	200	1
37.67	10.56	800	200	2
37.67	10.56	800	400	1
37.67	10.56	800	400	2
37.67	10.56	800	600	1
37.67	10.56	800	600	2
37.67	10.56	1200	200	1
37.67	10.56	1200	200	2
37.67	10.56	1200	400	1
37.67	10.56	1200	400	2
37.67	10.56	1200	600	1
37.67	10.56	1200	600	2

Width (um)	Height (um)	Speed (RPM)	Load (PSI)	Replication #
37.67	10.56	1600	200	1
37.67	10.56	1600	200	2
37.67	10.56	1600	400	1
37.67	10.56	1600	400	2
37.67	10.56	1600	600	1
37.67	10.56	1600	600	2
48.33	2.38	800	200	1
48.33	2.38	800	200	2
48.33	2.38	800	400	1
48.33	2.38	800	400	2
48.33	2.38	800	600	1
48.33	2.38	800	600	2
48.33	2.38	1200	200	1
48.33	2.38	1200	200	2
48.33	2.38	1200	400	1
48.33	2.38	1200	400	2
48.33	2.38	1200	600	1
48.33	2.38	1200	600	2
48.33	2.38	1600	200	1
48.33	2.38	1600	200	2
48.33	2.38	1600	400	1
48.33	2.38	1600	400	2
48.33	2.38	1600	600	1
48.33	2.38	1600	600	2
49.33	4.88	800	200	1
49.33	4.88	800	200	2
49.33	4.88	800	400	1
49.33	4.88	800	400	2
49.33	4.88	800	600	1
49.33	4.88	800	600	2
49.33	4.88	1200	200	1
49.33	4.88	1200	200	2
49.33	4.88	1200	400	1
49.33	4.88	1200	400	2
49.33	4.88	1200	600	1
49.33	4.88	1200	600	2
49.33	4.88	1600	200	1
49.33	4.88	1600	200	2
49.33	4.88	1600	400	1
49.33	4.88	1600	400	2
49.33	4.88	1600	600	1
49.33	4.88	1600	600	2
61	5.8	800	200	1
61	5.8	800	200	2
61	5.8	800	400	1
61	5.8	800	400	2
61	5.8	800	600	1
61	5.8	800	600	2

Width (um)	Height (um)	Speed (RPM)	Load (PSI)	Replication #
61	5.8	1200	200	1
61	5.8	1200	200	2
61	5.8	1200	400	1
61	5.8	1200	400	2
61	5.8	1200	600	1
61	5.8	1200	600	2
61	5.8	1600	200	1
61	5.8	1600	200	2
61	5.8	1600	400	1
61	5.8	1600	600	1
61	5.8	1600	600	2
64	11	800	200	1
64	11	800	200	2
64	11	800	400	1
64	11	800	400	2
64	11	800	600	1
64	11	800	600	2
64	11	1200	200	1
64	11	1200	200	2
64	11	1200	400	1
64	11	1200	400	2
64	11	1200	600	1
64	11	1200	600	2
64	11	1600	200	1
64	11	1600	200	2
64	11	1600	400	1
64	11	1600	400	2
64	11	1600	600	1
64	11	1600	600	2
131.33	1.4	800	200	1
131.33	1.4	800	200	2
131.33	1.4	800	400	1
131.33	1.4	800	400	2
131.33	1.4	800	600	1
131.33	1.4	800	600	2
131.33	1.4	1200	200	1
131.33	1.4	1200	200	2
131.33	1.4	1200	400	1
131.33	1.4	1200	400	2
131.33	1.4	1200	600	1
131.33	1.4	1200	600	2
131.33	1.4	1600	200	1
131.33	1.4	1600	200	2
131.33	1.4	1600	400	1
131.33	1.4	1600	400	2
131.33	1.4	1600	600	1
131.33	1.4	1600	600	2

A.3 Bearing Defect Experiment Inputs

Table 15: Full List of Inputs Used In Bearing Defect Experiment

Speed	yRMS_bandpass	zKurtosis_bandpass
Load	yVpeak_bandpass	zKurtosis_over
Replication	yRMS_over	zCrest_bandpass
xRMS_bandpass	yKurtosis_bandpass	zCrest_over
xVpeak_bandpass	yKurtosis_over	ae_RMS
xRMS_over	yCrest_bandpass	ae_Vpeak_over
xKurtosis_bandpass	yCrest_over	ae_Vpeak_bandpass
xKurtosis_over	zRMS_bandpass	ae_Kurtosis
xCrest_bandpass	zVpeak_bandpass	ae_Crest
xCrest_over	zRMS_over	

REFERENCES

[1] Aerts, M., Claeskens, G., and Hart, J., "Testing lack of fit in multiple regression," *Journal of the American Statistical Association*, vol. 94, pp. 869–879, 1999.

[2] Atkeson, C., Moore, A., and Schaal, S., "Locally weighted learning," *Artificial Intelligence Review*, vol. 11, pp. 11–73, 1997.

[3] Barron, A., "Complexity regularization with application to artificial neural networks," in *Nonparametric Functional Estimation and Related Topics* (Roussas, G., ed.), (Dordrecht, The Netherlands), pp. 561–576, Kluwer Academic Publishers, 1991.

[4] Barron, A. and Barron, R., "Statistical learning networks: A unifying view," in *Computing Science and Statistics: Proceedings 20th Symposium Interface* (Wegman, E., ed.), vol. 2, (Washington, D.C.), pp. 192–203, 1988. American Statistical Association.

[5] Barth, E., *Approximating Infinite Horizon Discrete-time Optimal Control Using CMAC Networks.* Georgia Institute of Technology: Ph. D. Dissertation, 2000.

[6] Barth, E. J. and Sadegh, N., "The limited coupling approximation with application to cmac networks," *International Journal of Smart Engineering System Design*, vol. 4, no. 3, pp. 195–204, 2002.

[7] Bellman, R., *Adaptive Control Processes: A Guided Tour.* Princeton, NJ: Princeton University Press, 1961.

[8] Berreby, D., "Chaos hits wall street," *Discover*, vol. 14, Mar 1993.

[9] Breiman, L., "Bagging predictors," *Machine Learning*, vol. 24, pp. 123–140, 1996.

[10] Breiman, L., "Arcing classifiers," *The Annals of Statistics*, vol. 26, no. 3, pp. 801–849, 1998.

[11] Brogan, W. L., *Modern Control Theory*. Englewood Cliffs, NJ: Prentice Hall, 3rd ed., 1991.

[12] Carreira-Perpinan, M., "A review of dimension reduction techniques," in *Technical Report CS-96-09*, (University of Sheffield), pp. 1–69, 1997.

[13] Chen, H., "Estimation of a projection-pursuit type regression model," *The Annals of Statistics*, vol. 17, no. 2, pp. 453–555, 1989.

[14] Cleveland, W. and Devlin, S., "Locally weighted regression: An approach to regression analysis by local fitting," *Journal of the American Statistical Association*, vol. 83, pp. 596–610, 1988.

[15] Cohen, S. and Intrator, N., "A hybrid projection based and radial basis function architecture," in *Proc. Int. Workshop on Multiple Classifier Systems (LNCS1857)* (Kittler, J. and Roli, F., eds.), (Sardingia), pp. 147–156, Springer, Jun 2000.

[16] Conan-Guez, B. and Rossi, F., "Multi-layer perceptrons for functional data analysis: a projection based approach," in *Neural Networks* (Dorronsoro, J., ed.), (Madrid, Spain), pp. 667–672, Springer, Aug 2002.

[17] Cover, T. and Thomas, J., *Elements of Information Theory*. New York, NY: John Wiley and Sons, Inc., 1991.

[18] Cowden, D., "Weights for fitting polynomial secular trends," in *Technical Paper 4*, (University of North Carolina), 1962.

[19] Davis, P. and Rabinowitz, P., *Methods of Numerical Integration*. New York: Academic press, 1975.

[20] De Veaux, R., Psichogios, D., and Ungar, L., "A comparison of two nonparametric estimation schemes: Mars and neural networks," *Computers in Chemical Engineering*, vol. 17, no. 8, pp. 819–837, 1993.

[21] De Veaux, R., Psichogios, D., and Ungar, L., "A guided tour of modern regression methods," in *Proceedings of the Section on Statistics in the Physical and Engineering Sciences*, American Statistical Assocation, 1995.

[22] De Veaux, R. and Ungar, L., "Multicollinearity: A tale of two non-parametric regressions," *Selecting Models from Data: AI and Statistics IV*, pp. 293–302, 1994.

[23] DeForest, E., "On some methods of interpolation applicable to the graduation of irregular series, such as tables of mortality," in *Annual Report of the Board of Regents*, (Smithsonian Institiute), 1873.

[24] Diaconis, P. and Freedman, D., "Asymptotics of graphical projection pursuit," *Annals of Statistics*, vol. 12, pp. 793–815, 1984.

[25] Donohoe, D. and Johnstone, I., "Projection-based approximation and a duality with kernel methods," *The Annals of Statistics*, vol. 17, no. 1, pp. 58–106, 1989.

[26] DSouza, A., Vijayakumar, S., and Schaal, S., "Are internal models of the entire body learnable?," *Society for Neuroscience Abstracts*, vol. 27, no. 406.2, 2001.

[27] FAN, J. and GIJBELS, I., *Local Polynomial Modelling and its Applications*. London: Chapman and Hall, 1996.

[28] FOLEY, D. C. and SADEGH, N., "Modelling of nonlinear systems from input-output data for state space realization," in *Proceeding of the 40th IEEE Conference on Decision and Control*, (Orlando, FL), pp. 2980–2985, Dec 2001.

[29] FRANK, I. and FRIEDMAN, J., "A statistical view of some chemometric regression tools," *Technometrics*, vol. 35, no. 2, pp. 109–135, 1993.

[30] FREUND, J. E., *Mathematical Statistics*. Englewood Cliffs, NJ: Prentice Hall, 2nd ed., 1971.

[31] FRIEDMAN, J., "Multivariate adaptive regression splines," *Annals of Statistics*, vol. 19, no. 1, pp. 1–141, 1991.

[32] FRIEDMAN, J., GROSSE, E., and STUETZLE, W., "Multidimensional additive spline approximation," *SIAM Journal on Scientific and Statistical Computing*, vol. 4, pp. 291–301, 1983.

[33] FRIEDMAN, J. and STUETZLE, W., "Projection pursuit regression," *Journal of the American Statistical Association*, vol. 76, pp. 817–823, 1981.

[34] GOLDBERG, D., *Genetic Algorithms in Search, Optimization, and Machine Learning*. AddisonŋWesley, 1989.

[35] GUGLIELMO, K. and SADEGH., N., "Theory and implementation of a robot repetitive controller with cartesian trajectory description," *ASME Journal of Dynamic Systems, Measurement and Control*, vol. 118, pp. 15–21, March 1996.

[36] HALL, P., "On projection pursuit regression," *The Annals of Statistics*, vol. 17, no. 2, pp. 573–588, 1989.

[37] HASTIE, T., TIBSHIRANI, R., and FRIEDMAN, J., *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. New York: Springer-Verlag, 2001.

[38] HORNIK, K., "Approximation capabilities of multilayer feedforward networks," *Neural Networks*, vol. 4, pp. 251–257, 1991.

[39] HORNIK, K., "Some new results on neural network approximation," *Neural Networks*, vol. 6, pp. 1069–1072, 1993.

[40] HUBER, P., "Projection pursuit," *The Annals of Statistics*, vol. 13, no. 2, pp. 435–475, 1985.

[41] HWANG, J., LAY, S., MAECHLER, M., MARTIN, R., and SCHIMERT, J., "Regression modeling in backpropagation and projection pursuit learning," *IEEE Transactions on Neural Networks*, vol. 5, no. 3, pp. 342–353, 1994.

[42] HWANG, J., YOU, S., LAY, S., and JOU, I., "The cascade correlation learning: a projection pursuit learning perspective," *IEEE Transactions on Neural Networks*, vol. 7, no. 2, pp. 278–289, 1996.

[43] INTRATOR, N., "Robust prediction in many parameter models: Specific control of variance and bias," *Statistics and Neural Networks: Advances at the Interface*, pp. 97–218, 2000.

[44] INTRATOR, O. and INTRATOR, N., "Interpreting neural-network results: a simulation study," *Computational Statistics and Data Analysis*, vol. 37, no. 3, pp. 373–393, 2001.

[45] JIMENEZ, L. and LANDGREBE, D., *High Dimensional Feature Reduction via Projection Pursuit*. West Lafayette, Indiana: Purdue University, 1995.

[46] JONES, L. K., "A simple lemma on greedy aproximation in hilbert space and convergence rates for projection pursuit regression and neural network training," *Annals of Statistics*, vol. 20, no. 1, pp. 608–613, 1992.

[47] JONES, L., "On a conjecture of huber concerning the convergence of projection pursuit regression," *The Annals of Statistics*, vol. 15, pp. 880–882, 1987.

[48] JONES, L., "Good weights and hyperbolic kernels for neural networks, projection pursuit, and pattern classification: Fourier strategies for extracting information from high dimensional data," *IEEE Transactions on Information Theory*, vol. 40, no. 2, pp. 439–454, 1994.

[49] KATKOVNIK, V., *Nonparametric Identification and Smoothing of Data: Local Approximation Method*. Moscow: Nouka, 1985.

[50] KELLY, K., *Out of Control: the rise of neo-biological civilization*. Perseus Books, 1994.

[51] KENDALL, M., *A Course in the Geometry of n-dimensions*. New York: Hafner Publishing, Co., 1961.

[52] KENNEDY, L. and BATRU, M., "Application of projection pursuit learning to boundary detection and deblurring in images," *Pattern Recognition*, vol. 33, no. 12, pp. 2019–2031, 2000.

[53] KENNY, P. and DURBIN, J., "Local trend estimation and seasonal adjustment of economic and social time series (with discussion)," *Journal of the Royal Statistical Society, Series A*, vol. 45, pp. 1–41, 1982.

[54] KWOK, T. and YEUNG, D., "Use of bias term in projection pursuit learning improves approximation and convergence properties," *IEEE Transactions on Neural Networks*, vol. 7, no. 5, pp. 1168–1183, 1996.

[55] LANCASTER, P. and SALKAUSKAS, K., *Curve and Surface Fitting: An Introduction*. London: Academic Press, 1986.

[56] LEWIS, F. L. and SYRMOS, V. L., *Optimal Control*. New York, NY: John Wiley and Sons, Inc., 2nd ed., 1995.

[57] LOADER, C., "Bandwidth selection: classical or plug-in?," *The Annals of Statistics*, vol. 27, 1999.

[58] MAECHLER, M., MARTIN, R. D., SCHIMERT, J., CSOPPENSZKY, M., and HWANG, J. N., "Projection pursuit learning networks for regression," in *Proceedings of the 2nd International IEEE Conference on Tools for Artificial Intelligence*, (Los Alamitos, CA), pp. 350–358, 1990.

[59] MARSDEN, J., *Elementary Classical Analysis*. San Francisco, CA: W.H. Freeman and Company, 1974.

[60] MASSYM, W., "Principal component regression in exploratory statistical research," *Journal of American Statistical Association*, vol. 60, pp. 234–246, 1965.

[61] MAYER, T., "Economics as a hard science: Realistic goal or wishful thinking," *Economic Inquiry*, p. 175, April 1980.

[62] MCLAIN, D., "Drawing contours from arbitrary data," *Computer Journal*, vol. 17, pp. 318–324, 1974.

[63] MICHALEWICZ, Z., *Genetic Algorithms + Data Structures = Evolution Programs*. New York, NY: Springerŋ Verlag, 1994.

[64] MOISEN, G. and FRESCINO, T., "Comparing five modelling techniques for predicting forest characteristics," *Ecological Modelling*, vol. 157, pp. 209–226, 2002.

[65] Olshausen, B. and Field, D., "Emergence of simple cell receptive field properties by learning a sparse code for natural images," *Nature*, vol. 381, pp. 607–609, 1996.

[66] Raviv, Y. and Intrator, N., "Bootstrapping with noise: An effective regularization technique," *Connection Science, Special issue on Combining Estimators*, vol. 8, pp. 356–372, 1996.

[67] Roweis, S. and Saul, L., "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, pp. 2323–2326, 2000.

[68] Rudin, W., *Functional Analysis.* New York, NY: McGraw-Hill, 2nd ed., 1991.

[69] Sadegh, N., "A multilayer nodal link perceptron network with least squares training algorithm," *International Journal of Control*, vol. 70, pp. 385–404, Mar 1998.

[70] Sadegh, P. and Ojelund, H., "Hierarchical local regression," in *Informatics and Mathematical Modeling Technical Report*, (Technical University of Denmark), pp. 1–25, Nov 2000.

[71] Savitsky, A. and Golay, M., "Smoothing and differentiation of data by simplified least squares procedures," *Analytical Chemistry*, vol. 36, pp. 1627–1639, 1964.

[72] Schiaparelli, G., "Sul modo di ricavare la vera ezpressione delle leggi delta natura dalle curve empiricae (translated=on the way to gain an understanding of how natural laws effect change from empirical curves)," *Effermeridi Astronomiche di Milano per l'Arno*, vol. 857, pp. 3–56, 1866.

[73] Scott, D., *Multivariate Density Estimation.* New York, NY: John Wiley and Sons, Inc., 1992.

[74] Shishkin, J., Young, A., and Musgrave, J., "The x-11 variant of the census method ii seasonal adjustment program," in *Technical Paper 15*, (Bureau of the Census, U.S. Department of Commerce), 1967.

[75] Sobolev, S., *Cubature Formulas and Modern Analysis: An Introduction*. Philadelphia, PA: Gordon and Breach Science Publishers, 1992.

[76] Stone, C., "Consistent nonparametric regression," *The Annals of Statistics*, vol. 25, pp. 1371–1470, 1977.

[77] Tagliaferri, R., Longo, G., Milano, L., Acernese, F., Barone, F., Ciaramella, A., De Rosa, R., Donalek, C., Eleuteri, A., Raiconi, G., Sessa, S., Staiano, A., and Volpicelli, A., "Neural networks in astronomy," *Neural Networks*, vol. 16, no. 3-4, pp. 297–319, 2003.

[78] Teeuwsen, S., Erlich, I., and El-Sharkawi, M., "Small-signal stability assessment based on advanced neural network methods," in *IEEE PES General Meeting*, (Toronto, Canada), 2003.

[79] Teh, Y. and Roweis, S., "Automatic alignment of local representations," *Advances in Neural Information Processing Systems*, vol. 15, pp. 841–848, 2003.

[80] Tenenbaum, J., de Silva, V., and Langford, J., "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, pp. 2319–2323, 2000.

[81] Vidakovic, B., *Statistical Modeling by Wavelets*. New York, NY: John Wiley and Sons, Inc., 1999.

[82] Vijayakumar, S. and Schaal, S., "Local adaptive subspace regression," *Neural Processing Letters*, vol. 7, no. 3, pp. 139–149, 1998.

[83] VIJAYAKUMAR, S. and SCHAAL, S., "Fast and efficient incremental learning for high-dimensional movement systems," in *Proc. International Conference on Robotics and Automation (ICRA2000)*, vol. 2, (San Francisco, California), pp. 1894–1899, 2000.

[84] VIJAYAKUMAR, S. and SCHAAL, S., "Lwpr: An o(n) algorithm for incremental real time learning in high dimensional space," in *Proceedings of Seventeenth International Conference on Machine Learning (ICML2000)*, (Stanford, California), pp. 1079–1086, 2000.

[85] VLASSIS, N., MOTOMURA, Y., and KROSE, B., "Supervised dimension reduction of intrinsically low-dimensional data," *Neural Computation*, vol. 14, pp. 191–215, Jan 2002.

[86] WAHBA, G., *Spline Models for Observational Data*. Philadelphia: SIAM, 1990.

[87] WAND, M. and JONES, M., *Kernel Smoothing*. London: Chapman and Hall, 1995.

[88] WANG, Z., ISAKSON, T., and KOWALSKI, B., "New approach for distance measurement in locally weighted regression," *Analytical Chemistry*, vol. 66, pp. 249–260, 1994.

[89] WOLD, H., "Soft modeling by latent variables: the noniterative partial least squares approach," in *Perspectives In Probability and Statistics*, (London), pp. 520–540, 1975.

[90] WOLPERT, D., "Stacked generalization," *Neural Networks*, vol. 5, pp. 241–259, 1992.

[91] WU, L. and TUMA, N., "Local hazard models," *Sociological Methodology*, vol. 20, pp. 141–180, 1990.

[92] Xu, Z.-B., Qiao, H., Peng, J., and Zhang, B., "A comparative study of two modeling approaches in neural networks," *Neural Networks*, vol. 17, no. 1, pp. 73–85, 2004.

[93] Yi, L. and Zhang, H., "Component selection and smoothing in smoothing spline analysis of variance models," in *Technical Report No. 1072r*, (University of Wisconsin), pp. 1–29, Jan 2003.

[94] Zhang, J. and Liang, S., "Sensor integrated on-line prognostics of bearing incipient defect based on artificial neural network," *57th Meeting of the Society for Machinery Failure Prevention Technology (MFPT 57)*, pp. 43–52, April 2003.

[95] Zhao, Y. and Atkeson, C., "Implementing projection pursuit learning," *IEEE Transactions on Neural Networks*, vol. 7, no. 2, pp. 362–373, 1996.

VITA

Michael D. Swinson was born on January 3, 1974 and grew up in New Castle, Delaware. He received his B.S. degree in Economics and in Mechanical Engineering in 1996 from Duke University, an M.S. degree in 2000 in Mechanical Engineering from Georgia Tech, and an M.S. in Statistics from Georgia Tech in 2002. He has worked in numerous positions ranging from a medicinal chemistry researcher with ICI Pharmaceuticals to Vice President of credit card forecasting with Bank of America to business analyst with Capital One to energy engineer with DuPont and much else in between. It is anticipated that he will receive his Ph.D. in December 2005.