NONPARAMETRIC LEARNING FROM EXAMPLES IN VERY HIGH DIMENSIONAL SPACES

By

GREGORY ZLATKO GRUDIĆ

B. A. Sc. (Engineering Physics) University of British Columbia, 1987

M. A. Sc. (Electrical Engineering) University of British Columbia, 1990

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

in

THE FACULTY OF GRADUATE STUDIES

(DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING)

We accept this thesis as conforming

to ~~the~~ required standard

THE UNIVERSITY OF BRITISH COLUMBIA

August 1997

[illegible]

Department of Electrical and Computer Engineering
The University of British Columbia
2075 Wesbrook Place
Vancouver, Canada
V6T 1W5

Date: August 14, 1997

Abstract

Constructing predictive models or mappings from sample data is an important goal in many areas of science. Most of the research in this area has been directed towards relatively low dimensional models; however, many real world problems can be large and very high dimensional. Such problems require general learning methods which are fundamentally different from those used to construct low dimensional models. In this thesis a new nonparametric regression methodology (SPORE) is proposed for very large and very high dimensional learning problems: problems with more than 10,000 learning examples of regression data having 100 or more inputs. The SPORE regression model is constructed incrementally by adding small, low dimensional parametric building blocks one at a time, using the outputs of previously added blocks as inputs to the new ones. This process forms stable regression functions from relatively few training examples, even when there are a large number of input variables. Furthermore, SPORE demands little computational effort to choose between candidate building blocks or inputs, making it computationaly feasible in very high dimensional spaces. SPORE challenges two basic mainstream notions found in contemporary learning algorithms. First, it questions the need to simultaneously fitting large high dimensional structures to model complex high dimensional interactions. Second, it rejects the need for "greedy", computationaly expensive searches used to finding the next "best" building block to add to a regression function. SPORE also allows for the subdivision of the domain of the input space to make incremental construction both computationaly and theoretically feasible. Conditions under which the rate of convergence of the method is independent of the dimension of the data are established. It is also shown that the computational complexity of constructing a SPORE-type regression model is linear with respect to dimension within each domain subdivision. In addition, conditions are given under which no domain subdivision is necessary.

The most basic version of this regression methodology (SPORE-1) is implemented and empirically evaluated on four types of data sets. The SPORE-1 learning algorithm is completely automatic and requires no manual intervention. First, SPORE-1 is applied to 10 regression problems found in the literature and is shown to produce regression functions which are as good or better, with respect to mean squared approximation error, than published results on 9 of these data sets. Second, SPORE-1 is applied to 15 new, synthetic, large, very high dimensional data sets

(40,000 learning examples of 100, 200, 400, 800, and 1600 inputs) and is shown to construct effective regression functions in the presence of both input and output noise. Third, SPORE-1 is used to build mappings from input/output learning examples generated by a human using teleoperation to execute an 'object locate and approach' task sequence. SPORE-1 effectively builds this mapping and directs a robot to autonomously execute the task demonstrated by the human operator, even in a visually cluttered environment with an unpredictable background. Finally, SPORE-1 is successfully applied to the 10-bit parity problem to demonstrate its efficacy on problems which have flat low dimensional projections, thus showing that it is not subject to the same limitations as other algorithms that build regression functions using low dimensional parametric building blocks, added one at a time.

Table of Contents

List of Tables

List of Figures

Acknowledgment

Many thanks to my supervisor Peter Lawrence for his support, patience, and above all, for never showing any doubt. His perspective on my research is invaluable to me and I could not have hoped for a better supervisor.

I am eternally grateful to my wife, Jane Mulligan, for being my best friend and most valued colleague. Much of the work presented here is shaped by her insights, and she has spent many long hours and reading and re-reading various versions of this thesis.

This thesis is dedicated to my mother Julijana and my father Josip.

Chapter 1: Introduction

All scientific predictions are based on models of the phenomena being predicted. This is equally true when one is predicting the path of a projectile as when one is trying to predict tomorrow's weather. All of these models are constructed starting from empirical observations of the phenomena of interest. These empirical observations, in whatever form they occur, are in essence input and output examples of what the model is intended to predict. The resulting predictive models can be of causal relationships between input and output or *visa versa* (such as predicting the path of a projectile given initial velocity, or conversely, predicting the initial velocity given its path), or they can be models reflecting some correlation between variables arbitrarily labelled as inputs and outputs, that have other, as yet unidentified, causal factors. Whatever the nature of the phenomenon in question, the goal is always the same: to convert empirical observations into a model structure which allows accurate prediction of future observations.

Building models from data often has two specific goals: the first is to predict output values given future inputs; the second is to use the resulting model to interpret or analyze the data. Interpretation of models involves analyzing which inputs most affect the output, as well as how these inputs interact with one another. Such analysis can reveal important information about the phenomenon that generated the data. This is especially true when relatively few input variables can effectively predict output values. However, the specific focus of this thesis is model prediction accuracy, and thus the topic of data analysis is largely ignored. One justification for this is that the models we are interested in are very high dimensional, making analysis difficult when many inputs combine in complicated nonlinear ways.

Most major theoretical and practical work on general methods for building models has been done in mathematics, in the form of function approximation, and statistics in the form of regression. More recently, the neural network and machine learning communities have addressed this general modeling problem, mainly using variations of tools which have their origins in mathematics and statistics. Many of these endeavors have met with much success, and many diverse fields, ranging from the physical to social sciences, have benefited from the modeling tools that have emerged.

In studying the literature on general modeling methods, it is apparent that most of the theoretical and practical work in this area has been applied to problems which are relatively small and low dimensional. Relatively little effort has been specifically directed towards general modeling methods where there are many potential inputs, and the model is to be built based on a very large data set.

Numerous examples of large high dimensional regression problems can be found in such diverse fields as meteorology, economics, and robotics. In meteorology, one can imagine predicting tomorrow's weather given temperature, pressure, humidity, and cloud covering readings from hundreds of weather stations surrounding the area of interest. In economics, predicting the major economic indicators based on a detailed time sampling of various economic indices can be a profitable venture. In robotics, building mappings between a robot's sensor inputs and actuator outputs can be considered to be a problem in regression, if, as in human-to-robot skill transfer, one has access to examples of the mapping (see Chapter 4 and [Grudic and Lawrence, 1995] [Grudic and Lawrence, 1996] for details). All of these regression problems have one characteristic in common: a large number of factors influence the output, and therefore, it is not easy to choose a small set of inputs which effectively explain the output.

The goal of this thesis is the theoretical and experimental study of building very high dimensional models from very large data sets. The type of modeling problem addressed typically has at least 100 inputs, and there are 10,000 or more empirical input-to-output examples of the phenomenon of interest. To this end, a methodology, termed SPORE, for building such regression models is presented and theoretically and experimentally analyzed. From a practical standpoint the end result of this thesis is a regression algorithm and structure, called SPORE-1, which can be used to build such large models. A key characteristic of SPORE-1 is that no *a priori* knowledge of the phenomenon of interest is required for model construction: all that the user must provide is a file containing the learning data.

1.1 Constructing Models from Data

The problem being addressed in this thesis is deceptively simple to state. There is a data set containing M input/output examples $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ for $i = 1, \ldots, M$, symbolized by $\{\boldsymbol{x}_i, \boldsymbol{y}_i\}_1^M$. The inputs $\boldsymbol{x} = (x_1, \ldots, x_N)$ are N dimensional and the outputs $\boldsymbol{y} = (y_1, \ldots, y_k)$ are k dimensional. The goal is to find some mapping function, $\hat{\boldsymbol{f}}$, such that:

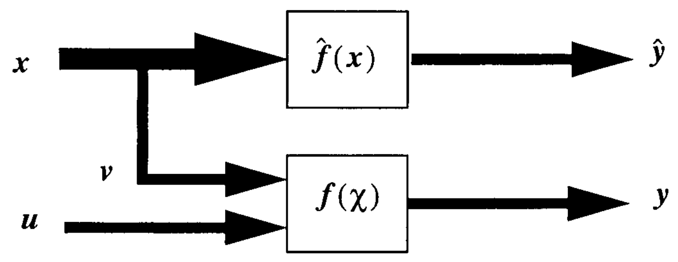


Figure 1.1 The Mapping Problem

$$\hat{y} = \hat{f}(x) \tag{1.1}$$

where $\hat{y}$ is some estimate of y given x. Once constructed, $\hat{f}$ can then be used to predict y given some future input x. In the general case, the data $\{x_i, y_i\}_1^M$ is generated by some unknown phenomenon or function $f(\chi)$, where $\chi = (u, v)$ is a P dimensional input vector. χ can have 2 types of inputs: inputs u which *do not* appear in the data set $\{x_i, y_i\}_1^M$, and inputs v which *do* appear in the data $\{x_i, y_i\}_1^M$. Therefore, one is often required to construct $\hat{f}$ using an incomplete set of inputs (i.e. we may not have access to the inputs u). A further difficulty arises when the accessible inputs x include some unknown set of inputs which do not affect the output vector y (i.e. not all inputs found in x will also be found in v). This is diagrammatically represented in Figure 1.1. Not knowing which inputs actually have some definable relationship to the output is just one of the potential difficulties encountered in the general model construction problem.

The process of constructing models from data has been studied in a variety of fields, and, not surprisingly has been given different names in each field. In applied mathematics this process is termed *function approximation*, and one speaks of approximating the dependent variable y as some function of the independent variable x. In statistics, the process of model building is called *regression* and the terminology used is as follows: estimate the responses y given the explanatory or predictor variables x. More recently, the machine learning and neural network communities have termed this type of model building *supervised learning* and typically refer to learning the mapping between input variables x and output variables y.

In this thesis, the process of building models from data is studied from the point of view of applied mathematics (function approximation) and statistics (regression). Referring to Figure 1.1, function approximation typically deals with the case of $u = \varnothing$ (i.e. the empty set) and $v \equiv x$. Thus, function approximation assumes that the independent variables x exactly define the dependent variables y, and the goal is to determine how well one can model f given some defined structure for the approximating function $\hat{f}$. Regression on the other hand, deals with the case when $u \neq \varnothing$ and/or $v \neq x$. Typically, this is represented in the following form:

$$\hat{y} = \hat{f}(x) + \varepsilon \tag{1.2}$$

where, ε is a noise term or, more formally, a random variable obeying some probability distribution. In most instances, ε is assumed to be normally distributed with mean zero and fixed standard deviation. Hence, from the perspective of regression, not knowing whether the given inputs completely define the output is treated as estimation noise.

There are two basic approaches to constructing the function $\hat{f}$: parametric and nonparametric. In the parametric approach, knowledge about the nature of the problem is used to define the structure and size of the function $\hat{f}$. Typically $\hat{f}$ will contain a set of parameters that need to be assigned values using some numerical algorithm such as gradient descent, least squares, or simulated annealing (to mention just a few). For example, if $\hat{f}$ is a polynomial with a finite number of terms, then the parameters which are "learned" using sample data are the polynomial coefficients. A key characteristic of the parametric method is that because both structure and size are fixed, the space or class of functions representable by any single model $\hat{f}$ is inherently constrained. It is because of this characteristic that parametric models are often fast to compute and extremely effective when they are representative of the phenomenon being studied. Parametric models are in fact the best form of model to use when the nature of the problem is well understood.

In contrast, nonparametric models are most useful when the nature of the problem is poorly understood. In nonparametric modeling, the *basic* structure of the model is usually defined, but not the number of parameters in the final constructed $\hat{f}$ function. One example of a nonparametric model is a polynomial which does not have a fixed number of coefficients. A nonparametric model grows in size in order to effectively model the desired data. Thus, in the case of a polynomial, new terms are added until a sufficient model is constructed. In theory, a nonparametric model is able to represent a greater space of functions than a parametric model. However, because nonparametric regression does not have a fixed parameter space, the task of constructing models becomes considerably more difficult than with parametric models. As a result, nonparametric models are typically more time consuming to construct.

The specific problem being addressed in this thesis is model construction when little or no information about the phenomenon being studied is available. Thus, the main focus of this thesis is nonparametric model construction, or nonparametric regression. As indicated above, there are difficulties associated with nonparametric regression. One of the most pervasive of these difficulties is the *curse of dimensionality* [Bellman, 1961], which is briefly addressed below.

1.2 The Curse of Dimensionality

From a practical standpoint, the curse of dimensionality has two major implications for high dimensional nonparametric regression. The first results from the rate of convergence of nonparametric regression, and the second from the computational complexity of nonparametric regression. If we are to build regression algorithms which work well in high dimensional spaces, we must find ways of addressing both of these issues.

1.2.1 Rate of Convergence of Nonparametric Regression

There are a number of theoretical results associated with the rate of convergence of nonparametric regression (see Section 2.3 of Chapter 2 for details). In this thesis, we are mainly concerned with rate of convergence results which measure the decrease in approximation error as the number of sample points used to construct the regression function increases. For example, Stone [Stone, 1982] has shown that in general, and under appropriate regularity conditions, the number of sample points required to approximate a bounded continuous N dimensional function which is at least once differentiable and defined over a closed domain, grows exponentially with dimension N. This means that it is not feasible to densely sample high dimensional space (e.g. 10 or more input variables), and therefore, in to order approximate high dimensional functions, one must impose some type of smoothness constraint on the approximation [Friedman, 1994b]. In fact, all learning methods impose such a constraint. Nonparametric methods differ from parametric methods in that they impose fewer such constraints, and are thus able to form more flexible approximations. Imposing a smoothness constraint on approximation means that fewer sample points are required to determine the parameters of the approximation.

However, as is argued throughout this thesis and elsewhere [Friedman, 1994b], the problem associated with approximating high dimensional functions is not strictly due to the number of input variables, but rather due to the underlying complexity of the function being approximated. We measure complexity as the number of samples necessary to build an effective nonparametric approximation (see Section 4.3 of Chapter 4 for an example). Clearly it is possible to have a very complicated 1 dimensional function, or a very simple high dimensional function. Thus, one can argue that a regression problem is difficult because it requires large numbers of learning data, which is not by necessity related to the dimension of the problem. Our goal then, as model builders, should be to find regression algorithms which perform well in high dimensional spaces, when only relatively few learning examples (i.e. tens of thousands) are sufficient to build adequate models.

The goal then, when formulating structures for very high dimensional nonparametric regression is two-fold. First, we must ensure that the rate of convergence of the regression function is not overly dependent on the dimension of the underlying function which generated the learning data. In other words, the rate at which the nonparametric regression function converges to some stable function should not depend adversely on the dimension of the problem. Second, one must have an understanding of what class of functions are representable by a given nonparametric regression technique. If the data being used to construct the regression function is being generated by a function which belongs to this class, then the regression function will converge to it. In this case the approximation error converges to zero. However, if the function generating the sample points does not belong to this class of functions, then the approximation error will converge to some finite non-zero value. From a practical standpoint, if this approximation error is greater than the maximum error that is appropriate for the regression problem at hand, then the regression function being used is inadequate for the task.

Recently, much theoretical work has been done to define classes of functions which have corresponding regression functions with rate of convergence independent of dimension (see Section 2.3 of Chapter 2 for details). One of the main theoretical contributions of this thesis is that we establish a rate of convergence result which is independent of dimension on function spaces which have not been previously studied in this way.

1.2.2 Computational Complexity of Nonparametric Regression

The second aspect of the curse of dimensionality is the computational complexity of constructing high dimensional regression functions. Generally speaking, there are two commonly used methods for constructing nonparametric regression functions. The first builds regression functions by including all input variables simultaneously, thus building one or more large component functions which may be combined to form the final regression function. The second uses some form of variable selection to build up the regression function using smaller functional subunits. Both of these approaches can be effective for relatively low dimensional problems. However, consider the problem of approximating a function of 1000 variables. If one uses the first approach and attempts to fit all the variables simultaneously, then even if the regression function is a simple second order polynomial (i.e. one which has relatively few nonlinear terms) of 1000 variables, the regression algorithm needs to solve simultaneously for over 500,000 model parameters (calculated using the identity $m = (s+v)!/(s! \cdot v!)$, where m is the number of polynomial terms and hence the number of model parameters, s is the order of the polynomial and v is the number of input variables). Whether solved via gradient descent or least squares, the computational cost of simultaneously fitting 500,000 parameters is not insignificant [Press et al., 1988]. Thus even a simple simultaneous fit of a second order polynomial of 1000 variables is computationaly significant, and adding more nonlinear terms makes the problem correspondingly more expensive (for example a third order polynomial of 1000 variables has more than 1.6×10^8 model parameters). The computational cost becomes even more significant when the appropriate model structure is unknown (thus making the problem nonparametric), and many different types of large nonlinear models need to be constructed before an adequate one is found.

An additional difficulty associated with attempting to simultaneously fit many variables is due to the large number of training examples needed to form a stable fit. This results because a simultaneous fit of a large number of model inputs implies that many model parameters need to be fit at the same time (as discussed in the previous paragraph), which in turn requires many training samples for the model to converge to a stable solution [Friedman, 1994b] [Breiman, 1996b]. In terms of computational complexity, large numbers of learning data result in an increase in the computational complexity of the learning algorithm.

Next consider the 1000 variable regression problem using some method of variable selection. In the literature, variable selection is typically achieved by choosing the best q of possible N inputs. Thus, in order to determine the optimal set of q input variables, the number of model constructions and evaluations required is $\binom{N}{q} = \frac{N!}{q!(N-q)!}$. As q and N become large, this search for an optimal solution quickly becomes impractical. For example, choosing the best 500 out of a possible 1000 variables requires the evaluation of $\frac{1000!}{(500!)(500!)}$ model constructions (usually nonlinear models are needed and hence each individual construction can itself be computationaly expensive); such an exhaustive search is not currently feasible, and is unlikely to be in the near future. Even if one chooses the suboptimal solution of removing one variable at a time from the model, one would require $\frac{1}{2}(N(N+1)-(q-1)(q))$ model constructions and evaluations. Hence, using this method to find the "best" 500 out of 1000 inputs requires 375,750 model constructions and evaluations. Even this suboptimal search strategy is computationaly difficult given the computational complexity of constructing and evaluating nonlinear models of more than 500 variables. Similar computational complexity arguments can be made for any method which attempts to choose the "best" (by whatever measure one uses) low dimensional projections of high dimensional data (see Section 2.1.1 of Chapter 2).

Any practical nonparametric algorithm required to work on high dimensional regression functions must deal with the above complexity issues. In the next section, the SPORE approximation is introduced as a methodology designed to address both the computational complexity and rate of convergence problems associated with high dimensional nonparametric regression.

1.3 The SPORE Approximation

When we speak of the difficulties inherent in high dimensional nonparametric regression (i.e. the curse on dimensionality), we are referring to the difficulties of applying existing approximation techniques to high dimensional problems. Any approximation problem is potentially difficult. In fact, one dimensional problems are potentially just as difficult as one thousand dimensional problems, and the intrinsic difficulty of any approximation problem is directly related to the number of learning examples required to build a sufficient approximation. Thus, the curse of dimensionality is only a curse because most algorithms have been designed to work well in low dimensional spaces, and there is no theoretical reason why a one dimensional function requiring 50,000 training examples to build a sufficient model, should be any more easier to approximate than a one thousand dimensional function requiring the same number of training samples.

We propose the following methodology to address high dimensional nonparametric regression problems. First, in order to ensure that the number of learning samples required to build a sufficient model depends on the complexity of the regression problem and *not* on how many input variables there are, we build approximations by projecting the high dimensional learning data onto low dimensional parametric building blocks. Thus, the number of learning samples required to form stable approximations depends on the number of parameters in our parametric building blocks, and not on the dimension of the learning data. The result is that the rate of convergence of the algorithm to some fixed error, is in fact independent of dimension.

This use of low dimensional parametric projections has been extensively used in the past (see Chapter 2, Section 2.1.1.1 for details), however, it has invariably involved a systematic search for building blocks and associated inputs for the *next best* (assuming some measure of best) building block to add. As we have argued in Section 1.2.2, these types of search procedures are not feasible when we have very high dimensional input spaces. Therefore, such search strategies are, of necessity, not used in the algorithms proposed in this thesis.

The second key aspect of our methodology is that the algorithmic procedure we use to determine which building block to add next, or what its inputs will be, is not dependent on the number of input variables. This may appear to be an impossible or impractical algorithmic characteristic, however, from a theoretical standpoint, there is no reason why computationaly expensive variable or building block selection is necessary. This is illustrated in the Appendices of this thesis (see Section 5.2.1 of Chapter 5) where we present an approximation algorithm which is theoretically guaranteed to construct uniformly converging approximations to any bounded continuous function of arbitrary dimension, while always using the same building blocks and a random ordering of input variables. The arguments in these Appendices demonstrate that computationaly expensive searches for building blocks and their inputs is not a theoretical necessity. A practical example of this aspect of our methodology is the SPORE-1 algorithm (see Section 1.3.3), where the same parametric building block is used throughout the algorithm, and the order in which input variables are added to the building blocks is random.

The third and final key component of our methodology is that the input space of the regression function is sub-divided in order to avoid flat or zero projections from high dimensional learning data onto our low dimensional parametric building blocks. This is a necessary algorithmic component because it gives convergence to zero error when other algorithms which use low dimensional projections do not (see Section 4.5 of Chapter 4). In fact, one of the main criticisms of algorithms which use lower dimensional building blocks is that they do not work well on high dimensional data. In the appendices of this thesis we demonstrate a systematic method of space subdivision which gives theoretically proven convergence (see Section 5.2.1 of Chapter 5). In addition, we show that the benefits of space subdivision can be obtained with minimal computational effort and without actually subdividing the input space (see Section 3.3.4 of Chapter 3). We refer to this as Virtual Space Partitioning, and show that it is a direct consequence of using bootstrap sampling techniques [Efron, 1983] to randomly subdivide the input space when adding parametric building blocks to the regression function.

Science often works by pushing a *thesis* to an extreme point of view in order to establish conditions under which this view is valid, as well as conditions under which it breaks down. This document is an example of this form of scientific endeavor. The *thesis* being put forward, and upon which the SPORE approximation is based, is the following:

> *Very high dimensional nonparametric regression models can be effectively constructed by using a sufficient number of small, low dimensional parametric building blocks which are added to the model one at a time, using the outputs of previously added blocks as inputs to the new ones. Furthermore, one can use a random selection process to choose candidate building blocks (picked from some small finite set) and their inputs, thus little or no computational effort is required to determine what to fit next.*

Our objective is to both experimentally and theoretically establish under which conditions this *thesis* is valid, and to determine how restrictive these conditions are. It is argued throughout this document that the principal implication of the above *thesis* is that very high dimensional nonparametric regression is feasible, both in terms of rate of convergence and computational complexity. The surprising conclusion is that it is both feasible and effective on a large class of functions.

The **S**pace **P**artitioning, self-**O**rganizing, and dimensionality **RE**ducing, or **SPORE**, algorithmic philosophy defines our basic methodology for nonparametric regression. As the name suggests, the input *space* of the regression data is *partitioned* or divided into regions where it is possible to *reduce the dimension* of the problem by constructing the approximation only a few variables at a time. SPORE approximations are *self-organizing*, or equivalently nonparametric, in that the size of the resulting regression function is defined by the learning or regression data, and is not defined *a priori*. In Chapter 5, the general SPORE methodology is defined and theoretically analyzed. In the following, a brief description of the basic concepts behind SPORE are presented.

1.3.1 A Conceptual Summary of SPORE

We view SPORE as a methodology for designing learning algorithms which construct SPORE-type regression functions or approximations. The SPORE methodology has five basic characteristics which can be summarized as follows:

C1. The SPORE approximation consists of a finite number of functional units. These functional units are either structurally all identical (for example 2 dimensional, third order polynomials as in Chapter 4) or of different types (for example some may belong to a class of polynomials of a given range of dimension and order, while others may belong to one or more classes of radial basis functions of some range of dimension and number of basis units). From a practical perspective, these functional units should have the following property: the process of individually fitting these functional units to input/output data should be relatively easy when compared to the complete regression problem.

C2. The functional units are added to the approximation one unit at a time, and are not modified after they are added. Input variables, as well as the outputs of previously added functional units, can serve as inputs to new units. The addition of functional units stops when a desired residual error is reached.

C3. Little or no computational effort is spent on selecting inputs to structural units.

C4. Little or no computational effort is spent on selecting which type of structural unit is added next.

C5. The input space is partitioned in order to reduce approximation error.

Thus, dimensionality reduction is implied by characteristics **C1** and **C2**, self-organization is implied by **C2**, and space partitioning is implied by **C5** (thus the name SPORE as defined above). One should note that characteristics **C1** and **C2** are common to a number of other nonparametric regression architectures. In Chapter 2, these architectures are discussed in detail. Two notable examples are the Group Method of Data Handling (GMDH) [Ivankhnenko, 1971] (as well as various other related algorithms), and Cascade Correlation [Fahlman and Lebiere, 1990]. All of these algorithms build successively on structures, using the outputs of previously constructed structural units as inputs to new units. However, it is the further incorporation of characteristics **C3** and **C4** which make SPORE different from all other approaches to nonparametric regression. As described in Section 2.4 of Chapter 2, these two characteristics lead to learning algorithms which are significantly different from those studied previously. They are also a key to formulating a practical algorithm for very high dimensional regression. As argued in Section 1.2.2, the computational effort required for variable selection makes it difficult, if not impossible, to do in very high dimensional spaces. The same argument applies to structural unit selection in high dimensional spaces.

The purpose of the space partitioning characteristic, **C5**, is to ensure that the SPORE regression function is a universal approximator, or in other words is (theoretically) able to approximate any bounded continuous function defined over a finite domain in N dimensional space (see Chapter 5). As is briefly outlined in the next section, it is also used to avoid problems with rate of convergence in very high dimensional regression.

1.3.2 Theoretical Results

As stated previously, Stone [Stone, 1982] has shown that, in general and under appropriate regularity conditions, the number of sample points required to approximate a bounded continuous N dimensional function which is at least once differentiable and defined over a closed domain, grows exponentially with dimension N. The main theoretical result of this thesis is the following: We prove that such functions have a finite number of sub-domains where there exists at least one type of regression function (e.g. one of the SPORE structures define in Chapter 5) which has a rate of convergence independent of dimension. The SPORE approximation is in fact built by partitioning the input space into subdomains where the rate of convergence to zero approximation error is independent of dimension. This result is significant from both a theoretical and practical point of view. Theoretically it demonstrates that much broader function classes can have rates of convergence independent of dimension (see Section 2.3 of Chapter 2 for details). In practice it points the way to practical nonparametric regression algorithms which can work in very high dimensional spaces.

A second theoretical result given in this thesis concerns the computational complexity required to build a SPORE structure. As stated above, SPORE subdivides the input space of the function being approximated into a finite number of subdomains where the rate of convergence is independent of dimension. In Chapter 5, it is further shown that the computational complexity of constructing SPORE in each of these subdomains is on the order of $O(N \times C)$, where N is the dimension of the data and C is the average complexity of adding a functional unit (C is indepen-

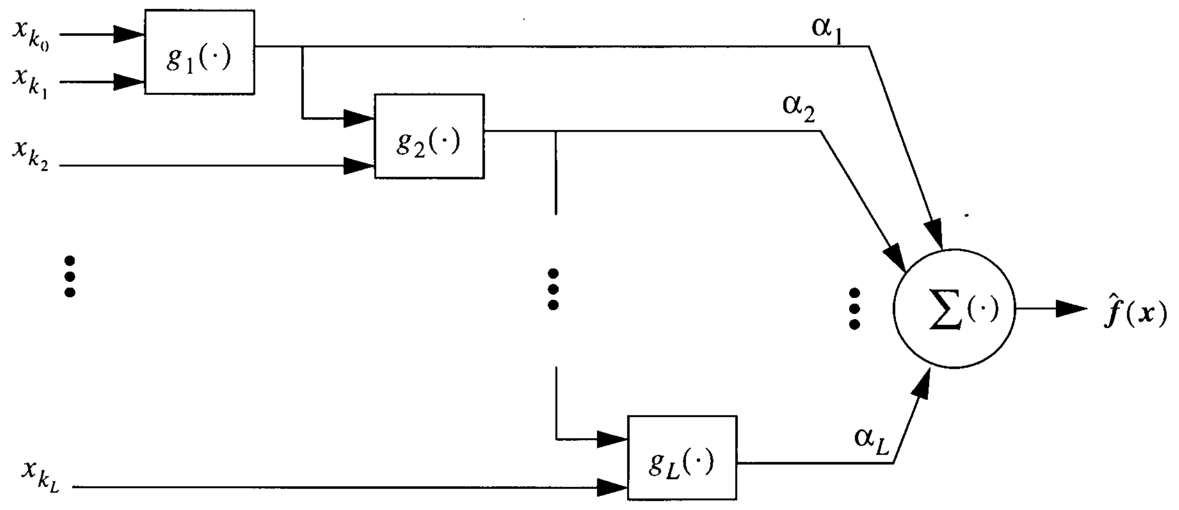


Figure 1.2 The SPORE-1 Structure

dent of N). This indicates that, at least within each subdomain, the computational complexity of constructing an approximation is at most linear with dimension. This is a promising result from the point of view of practical nonparametric regression.

One theoretical question which this thesis does not address is how many such subdomains are necessary. Having desirable theoretical properties in each subdomain may not mean much if there are exponentially many such domains in practice. Although this remains an open theoretical question, in Chapter 5 some theoretical justification is given which suggests that functions requiring many subdomains are unlikely. Furthermore, in Chapter 3 and Chapter 5 we establish conditions under which subdivision is not necessary. It is interesting to observe that none of the empirical evaluations of the SPORE approximation required domain subdivision (see Chapter 4 for details). Thus, for at least the regression problems studied in this thesis, one domain is sufficient (although virtual space subdivision was necessary to obtain these results: see Section 3.3.4 of Chapter 3).

1.3.3 The SPORE-1 Regression Function

In Chapter 3, the most basic form of SPORE, termed SPORE-1, is defined and a construction or learning algorithm is given. All of the experimental evaluations presented in this thesis (see Chapter 4) are done using SPORE-1. A brief description of SPORE-1 is give next.

As shown in Figure 1.2, SPORE-1 consists of a cascade of 2 dimensional functions $g_l(\cdot)$, which are scaled (α_l) and summed to produce the regression function $\hat{f}(\boldsymbol{x})$: the subscripts $k_0, \ldots, k_L \in \{1, \ldots, N\}$ serve to identify the input variables $(x_1, \ldots, x_N)$. The exact algorithm used to construct the SPORE-1 approximation is fully defined in Chapter 5. The points of note in the learning algorithm are as follows:

1. the functions (building blocks) $g_l(\cdot)$ are 2-dimensional polynomials, all having the same number of coefficients;
2. the functions $g_l(\cdot)$ are added one at a time in order $g_1(\cdot)$, $g_2(\cdot)$,..., $g_L(\cdot)$ (the learning data determines how many levels are constructed);
3. the order of inputs $(x_1, \ldots, x_N)$ is random, and any single input may enter the cascade at many levels; and,
4. the input space is randomly subdivided (using bootstrap) whenever a new $g_l(\cdot)$ is added.

Hence, the SPORE-1 learning algorithm conforms to the general SPORE methodology given in Section 1.3.1.

Consider the simple nature of SPORE-1. The algorithm does not use a complicated variable selection technique (it is in fact random variable selection), and simple parametric functions (all identical) are incrementally added to the regression function. In fact, SPORE-1 is an experiment in pushing the lower bounds in algorithmic and structural simplicity, the goal being to test the efficacy of the general SPORE methodology at this most basic level. However, as summarized in Section 1.3.5, this surprisingly simple SPORE-1 structure can indeed form effective nonparametric regression functions.

1.3.4 Why Does SPORE-1 Work and When Does It Work Best?

Given the simple structure of SPORE-1, it is not unreasonable to ask why SPORE-1 is able to form effective regression functions. Roughly speaking there are three basic reasons for this (see Chapter 3 for details). These are briefly addressed below.

First, that the addition of each new building block to the structure defined in Figure 1.2 can have, on average, only one of 2 consequences: 1) it can reduce the model error because of the addition of the new input variable (chosen randomly) to the regression function; or 2) it can leave the model error unchanged, because if the addition of the new input variable does not contribute to a better model, then the new building block $g_l(\cdot)$ simply passes the output of the previous level

$g_{l-1}(\cdot)$ to the input of the next level $g_{l+1}(\cdot)$ (i.e. $g_l(\cdot) = g_{l-1}(\cdot)$). The result is that the order in which input variables are added to the regression functions does not significantly affect model error, as long as the relevant variables are eventually incorporated.

The second reason why SPORE-1 is effective is because it forms stable high dimension regression functions using relatively few learning examples. This is because 1) high dimensional data is projected onto low dimensional parametric building blocks $g_l(\cdot)$ which implies that relatively few training examples are required to form stable model parameters [Friedman, 1994b], and 2) the outputs from these building blocks are combined using a weighted average (in the form of the scaling factors α_l) which tends to further stabilize the regression function [Breiman, 1996b] (see Section 3.3.3 of Chapter 3).

The third reason why SPORE-1 is effective is because the low dimensional projections (parametric building blocks) $g_l(\cdot)$ tend to have un-correlated residual errors and are unlikely to be flat (i.e. $g_l(\cdot) = g_{l-1}(\cdot)$) if the input x_{k_l} to $g_l(\cdot)$ has an effect on the output. The reason for this is due to the random subdivision of the input space resulting from the use of bootstrap samples of the training data whenever a new level $g_l(\cdot)$ is added (see Section 3.3.3 and Section 3.3.4 of Chapter 3). The result of this is that adding new levels (building blocks) to the regression function tends to improve the approximation, as long as the input variables continue to effect the output.

The next question one might ask is the following: what types of problems is SPORE-1 best suited for? Given that the SPORE-1 structure consists of a long cascade of parametric building blocks (see Figure 1.2), with each level in the cascade incorporating additional input variables into the regression function, it is reasonable to assume that SPORE-1 is best suited to regression problems where many input variables are required to produce an effective model, and the relevant input variables have approximately equally significant effects on the output. In addition, if all of the parametric building blocks ($g_l(\cdot)$) are continuous (which they are in the current implementation; see Chapter 3 for details), in the ideal case, the target function would also be continuous. Although this is a compelling argument for defining an ideal class of target functions, there is significant experimental evidence that, in practice, SPORE-1 is not limited to only these types of regression problems. This is briefly discussed in the next section.

1.3.5 Experimental Evaluation of the SPORE-1 Approximation

The SPORE-1 approximation is evaluated experimentally in Chapter 4. First, SPORE-1 is compared with other algorithms by applying it to 10 regression problems found in the literature [Grudic and Lawrence, 1997]. This initial evaluation shows that SPORE-1, on average, does at least as well or better (with respect to mean squared approximation error) than published results on 9 of these data sets. All of these published regression problems are relatively small and low dimensional, and thus not the type of problems SPORE-1 was designed for. Hence it is interesting to observe SPORE-1's effectiveness on such problems.

Next, SPORE is evaluated on 15 synthetic, large, high dimensional data sets (40,000 learning examples of data having 100, 200, 400, 800 and 1,600 inputs) [Grudic and Lawrence, 1997]. This synthetic data is generated specifically to evaluate SPORE-1 on the type of data sets it was designed for. Some of these data sets have output noise (i.e. the ε in equation (1.2)), while others include input variables which do not affect the output (i.e. $\boldsymbol{v} \neq \boldsymbol{x}$ in Section 1.1). Synthetic data allows us to test theoretical limits with full knowledge of what the best attainable approximation errors are. The results from these experiments indicate that SPORE-1 closely approaches these best possible approximation errors. As no data sets of sufficient size and dimensionality were found in the literature, this is the first use of this data for evaluation of high dimensional regression.

Next, SPORE-1 is applied to 3 high dimensional (1024 pixel inputs) human-to-robot skill transfer problems. These experiments involve having a human operator demonstrate a mobile robot "locate object and approach object" motion sequence using video input. During this demonstration, the human uses only the robot's sensors as inputs, and controls the robot's actuator as output. The demonstration generates a learning data set via recording the image pixels seen by the human, as well as the corresponding output commands, during each time step of the demonstration. This data set is used to construct a mapping between sensor inputs and actuator outputs, which when constructed, is used to autonomously control the robot. Experimental results indicate that the robot is able to autonomously accomplish the desired task. Although, as indicated in Chapter 4, the task being transferred from human to robot is relatively simple, these experiments demonstrate the ability of SPORE-1 to form very high dimensional regression functions, based on real world learning data.

Finally, SPORE-1 is applied to the 10-bit parity problem [Grudic and Lawrence, 1997]. We chose this problem to demonstrated that SPORE-1 does not have the same limitations as other systems which build regression functions using small dimensional units which are added one at a time.

1.4 Plan of Presentation

In Chapter 2 a review of nonparametric regression is presented. In Chapter 3 the simplest version of SPORE (SPORE-1) is described, and theoretical results are given. In Chapter 4 an empirical evaluation of SPORE-1 is presented. In Chapter 5 the general SPORE methodology is defined and the basic theory behind it is described. Chapter 6 contains a summary of the conclusions and major technical contributions of this thesis, and includes a proposal for future theoretical and empirical evaluation of SPORE.

Chapter 2: Review of Nonparametric Regression

This Chapter gives a brief review of nonparametric regression. In Section 2.1, a discussion of various nonparametric regression algorithms is given. In Section 2.2, the problem of over-fitting in nonparametric regression is discussed and various published solutions are mentioned. In Section 2.3, convergence results of various regression functions are discussed. Finally, in Section 2.4, a comparison is made between published regression algorithms and the regression methodology proposed in this thesis (SPORE).

2.1 Regression Algorithms: Addressing the Curse of Dimensionality

As discussed in Chapter 1, the curse of dimensionality has two serious detrimental consequences for practical high dimensional nonparametric regression. The first is the potential necessity for a large learning data set which is impractical, and the second is the computational complexity associated with a large number of input variables. Many algorithms have been presented in the literature which are designed to deal with these two difficulties. Good reviews and descriptions of many types of regression and approximation algorithms can be found in the literature [Kwok and Yeung, 1997] [Haykin, 1994] [Friedman, 1994b] [Hastie and Tibshirani, 1994] [Alfeld, 1989]. In the following, we discuss some representative examples of these algorithms. All of the algorithms discussed below have been demonstrated to be useful on a variety of different types of regression problems. Each has its own particular strengths and weaknesses, however, none of these algorithms have been demonstrated to be effective on the large, very high dimension nonparametric regression problems studied in this thesis.

For the purposes of this thesis, we view nonparametric learning algorithms as broadly grouped into two major categories. The first is termed *dimensionality reducing algorithms,* and the second is called *space partitioning algorithms.*

2.1.1 Dimensionality Reducing Algorithms

The goal in dimensionality reduction is to break a large regression problem into a number of smaller regression problems. The end result of this is a reduction in the number of parameters which are being fitted to the learning data. A reduced parameter set implies that algorithms which fit learning data to these parameters can potentially be more efficient. In addition, fewer parameters generally require fewer learning samples in order to achieve stable parameter values. In the following, we will briefly discuss some of the better known examples of dimensionality reducing algorithms. For organizational purposes, we divide these algorithms into two types.

2.1.1.1 Sum of Approximated Lower Dimensional Projections

Let $f(\mathbf{x})$, where $\mathbf{x} = (x_1, \ldots, x_N) \in D \subseteq \Re^N$, be an N dimensional function, and let $\hat{f}(\mathbf{x})$ be some approximation of $f(\mathbf{x})$. Then in one form of dimensionality reduction, $\hat{f}(\mathbf{x})$ has the following structure:

$$\begin{aligned}\hat{f}(\mathbf{x}) = & \sum_{i_1 \in \{1, \ldots, N\}} f_{i_1}(x_{i_1}) + \sum_{i_1, i_2 \in \{1, \ldots, N\}} f_{i_1 i_2}(x_{i_1}, x_{i_2}) + \ldots + \\ & \sum_{i_1, \ldots, i_m \in \{1, \ldots, N\}} f_{i_1 \ldots i_m}(x_{i_1}, \ldots, x_{i_m})\end{aligned} \tag{2.1}$$

where, for all $m \in \{1, \ldots, N\}$, $f_{i_1 \ldots i_m}$ are m dimensional functions defined over the domain D of $f(\mathbf{x})$. Generally speaking, the goal is to have m much smaller than N, and to find some finite set of functions $f_{i_1 \ldots i_m}$ which sum to effectively approximate the target function. The hope is that effective approximations are possible using functions which are either lower dimensional or have some fixed finite parametric structure. Examples of this type of dimensionality reduction include Generalized Additive Models [Hastie and Tibshirani, 1986], MARS [Friedman, 1991], GMDH [Ivankhnenko, 1971], ASPN [Elder and Brown, 1995], and SONN [Tenorio and Lee, 1990]. These are briefly discussed below.

In Generalized Additive Models [Hastie and Tibshirani, 1986] [Hastie and Tibshirani, 1990] regression functions are constructed using a sum of smooth 1 dimensional functions $\sum f_{i_1}(x_{i_1})$. The 1 dimensional functions f_{i_1} are nonparametrically constructed, one at a time, using a scatterplot smoother. One advantage of this technique is that the construction of the functions f_{i_1} is completely automated, requiring no interventions in the learning process. In addition, the resulting regression model can often be effectively interpreted, allowing a statistician to analyze significant factors involved in model generation. One limitation of this method is that higher dimensional interactions are difficult to model, however, in many lower dimensional regression problems this is not a significant concern.

In Multidimensional Adaptive Regression Splines (MARS) [Friedman, 1991], regression functions are constructed using a sum of products of spline functions. These functions are built in a forward stepwise hierarchical manner which is based on recursive partitioning (see Section 2.1.2). MARS is able to automatically account for interactions between variables, in a controlled manner, allowing for effective model interpretation. A limitation of MARS is that its computational complexity increases rapidly with dimension, making large regression problems (i.e. thousands of learning samples) of more than about 20 input variables, impractical when high dimensional interactions occur. However, in many applications, low dimensional interactions are sufficient to form effective approximations.

Polynomials have been widely used to build regression models. Polynomial models have the following form (note that these types of multidimensional polynomials are often termed multinomials):

$$\hat{f}(\mathbf{x}) = \sum_{\substack{i_1, \ldots, i_N \in \{0, \ldots, M\}}}^{i_1 + \ldots + i_N \le M} a_{i_1 \ldots i_N} \cdot x_1^{i_1} \ldots x_N^{i_N} \tag{2.2}$$

where M is the order of the polynomial. These models are built by fitting the parameters $a_{i_1 \ldots i_N}$ to the learning data. One difficulty with this general polynomial formulation is that the number of parameters $a_{i_1 \ldots i_N}$ increases rapidly with polynomial order and dimension, thus making it difficult to determine which subset of parameters are significant. Many iterative schemes have been developed which attempt to chose the most relevant parameters, and construct the approximation a few terms at a time. One such algorithm, originated by Ivakhnenko [Ivankhnenko, 1971], is the Group Method of Data Handling (GMDH). The original algorithm builds the global approximating polynomial using layers of 2 dimensional quadratic polynomials. The first layer is constructed using enough quadratic polynomials to exhaustively account for all possible input combinations. The outputs of this first layer are then used as inputs to the next layer, where they are once more exhaustively combined two inputs at a time. This process of adding layers continues until further additions to the polynomial do not reduce the approximation error. There have been many enhancements to this original algorithm since it was introduced by Ivakhnenko [Farlow, 1984]. Some specific examples include the Algorithm for Synthesis of Polynomial Networks (ASPN) [Elder and Brown, 1995], which extends the paradigm to include 3 dimensional polynomials, and the Self Organizing Neural Network (SONN) [Tenorio and Lee, 1990], which allows for basis functions other than polynomials and uses a Simulated Annealing technique to search for an optimal structure. Other variations of this include the method by Sanger [Sanger, 1991] where subnetworks of polynomials (or other basis functions) are grown below the previously added terms which exhibit maximum variance in approximation error. All of these algorithms have demonstrated both good generalization and interpretation properties, on many types of regression problems. One common characteristic of these methods is that they all use some form of search in variable space (and often basis function space as well) to define the structure of the resulting polynomial. Ultimately, due to the computational complexity of such searches (see Section 1.2.2 of Chapter 1), these algorithms are not practical for the very high dimensional regression problems studied in this thesis.

2.1.1.2 Summed High Dimensional Basis Functions

A second general type of dimensionality reducing algorithm constructs regression functions that have the following structure:

$$\hat{f}(\mathbf{x}) = \sum_{q=0}^{M} g_q(h_q(\mathbf{x})) \tag{2.3}$$

where g_q are 1 dimensional functions, the basis functions h_q are functions of a known type chosen *a priori*, and the number of terms M is chosen large enough to give effective approximations. Examples of these regression functions include Projection Pursuit [Friedman and Stuetzle, 1981], Radial Basis Functions [Hardy, 1971], Hinging Hyperplanes [Breiman, 1993], Wavelets [Sjoberg et al., 1995] [Juditsky et al., 1995] [Zhang, 1997], Cascade Correlation [Fahlman and Lebiere, 1990], and other adaptive Neural Network type algorithms [Reed, 1993] [Igelnik and Pao, 1995] [Yao and Lui, 1997]. These are briefly discussed below.

In Projection Pursuit [Friedman and Stuetzle, 1981], the functions h_q are linear, having the form $h_q(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i \cdot x_i$, where the parameters α_i are chosen to maximize error reduction, given that the functions g_q are restricted to be smooth and nonparametric. This minimization of error process can be accomplished using Newton-Raphson [Press et al., 1988] on an appropriate cost function. The residual errors are updated after the construction of each h_q and g_q function, and the construction of new functions stops when the approximation error is sufficiently small. Projection Pursuit is often used as an analysis tool because determining which projection axis (defined by function h_q) gives the most error reduction, can shed light on which independent variables are important and how they interact. A limitation of this method is that all approximations are made along N dimensional linear projection axes (defined by functions h_q), making the construction of these axes computationally difficult for very high dimensional regression problems ($N > 100$).

Cascade-Correlation [Fahlman and Lebiere, 1990] is an example of a nonparametric Artificial Neural Network regression function. Neural networks are generally implemented using sigmoidal basis functions (artificial neuron) of the form $s(z) = 1/(1 + e^{-z})$, where $z = \alpha_0 + \sum_{i=1}^{M} \alpha_i \cdot u_i$, with u_i being the inputs to the basis function and the parameters α_i being adjusted to fit the training data. Neural networks often consist of layers of these sigmoidal functions, with each layer containing a potentially large number of *neurons* or sigmoidal units, and the output of lower layers serving as inputs to higher layers (each unit in the first layer having as its inputs all of the predictor variables).

The Cascade-Correlation algorithm is designed to automatically determine the network structure based on the training data. Cascade-Correlation adds hidden units (i.e. units not directly connected to the output) to the network one at a time, with each hidden unit having an input connection from all the previously added hidden units, as well as all of the predictor variables. Thus the number of inputs into hidden units grows linearly with the number of levels added to the network, with the first unit having N inputs (one from each of the predictor variables). In addition, each time a hidden unit is added, many potential candidate units are trained using the Widrow-Hoff delta rule [Widrow et al., 1976] starting from randomly chosen weights α_i. Only the one which best correlates with the current residual error is added as the next hidden unit.

The efficacy of Cascade-Correlation on high dimensional data has been widely demonstrated in the literature [Michie et al., 1994]. However, because each hidden unit accepts inputs from all predictor variables, as well as the outputs of all existing hidden units, the method exhibits high computational complexity and potential numerical instability (due to a large number of parameters), when the regression problem is very high dimensional. As a result, Cascade-

Correlation is unsuitable when large numbers of hidden units are necessary for effective approximation, and thus is not ideally suited for the very high dimensional regression problems studied in this thesis.

Other nonparametric artificial neural network algorithms work by either starting with a large network and pruning until an effective approximation has been achieved (*pruning* algorithms [Reed, 1993]), or by starting with a small network and adding interconnections and basis functions (artificial neurons) until no further error reduction is possible (*constructive* algorithms [Kwok and Yeung, 1997]). Both constructive and pruning type neural network algorithm have the same goal: to find a neural network structure which is large enough to effectively model the learning data, while small enough to ensure that over-fitting does not occur (see Section 2.2 for a discussion of variance reduction techniques and the problem of over-fitting). Other learning algorithms use genetic programming techniques [Yao and Lui, 1997] and stochastic basis function generation techniques [Igelnik and Pao, 1995] to achieve the same goal. These algorithms have been shown to perform well when relatively small numbers of artificial neurons effectively model the learning data. However, when large numbers of artificial neurons are necessary, the learning process can become unstable (small changes in learning data leading to large changes in model structure [Breiman, 1996b]) because of the large number of parameters in the neural network. This makes these types of neural network learning algorithms unsuitable for the large high dimensional problems studied in this thesis.

In Radial Basis Function methods [Jackson, 1988] [Hardy, 1971] [Poggio and Girosi, 1989] [Broomhead and Lowe, 1988] [Casdagli, 1989] [Renals and Rohwer, 1989] [Poggio and Girosi, 1990] [Edelman and Poggio, 1990] [Mel and Kock, 1990] [Saha and Keeler, 1990] [Girosi, 1994], the functions h_q have the form $h_q(\mathbf{x}) = \phi(\|\mathbf{x} - \mathbf{x}_q\|)$, where ϕ is a function which varies according to some distance metric from a center point $\mathbf{x}_q$. The functions g_q take the form $g_q(\mathbf{x}) = \alpha_q \cdot h_q(\mathbf{x})$ where the parameters α_i are defined by the training data. Radial Basis Functions become nonparametric when the number and type of basis functions h_q, and the location of the centers $\mathbf{x}_q$ is determined by the learning data. A limitation of Radial Basis Functions is that they all use some measure of distance between points in space, and in high dimensional space, almost every learning sample point is closer to the sample boundary than to another learning point [Friedman, 1994b]. This makes radial basis functions not ideally suited for very high dimensional nonparametric regression.

Hinging Hyperplanes [Breiman, 1993] represent regression functions as a linear superposition of basis functions which consist of 2 hinging hyperplanes. These hyperplanes are hinged in the sense that two planes in N dimensional space must always intersect at a line (or hinge) in N dimensional space. The learning algorithm works by finding the hinge between 2 hyperplanes which best fit the current residual error, thus producing a basis function. The next basis function is then produced in a similar manner using the new residual errors resulting because of the addition of the previous basis function. If N is small, the hyperplanes are chosen to be N dimensional, otherwise the best $M < N$ dimensions are chosen for the hyperplanes. As is argued in Chapter 1, such searches make very high dimensional regression computationally difficult. Thus Hinging Hyperplanes are not an ideal candidate for the type of regression problems studied in this thesis. However, the computational efficiency of Hinging Hyperplanes on certain high dimensional problems (see [Breiman, 1993] for examples), make it an important potential candidate for many regression problems.

Wavelets [Sjoberg et al., 1995] [Juditsky et al., 1995] have also been effectively used in nonparametric regression. Wavelets have the attractive property that they are spatially adaptive, and thus they are effective at approximating functions which are locally spiky (i.e. are smooth throughout most of the input space, but have certain regions where they are rapidly changing). Wavelet shrinking algorithms have demonstrated good performance on relatively low dimensional problems (3 inputs or less) [Sjoberg et al., 1995]. However, wavelet algorithms which demonstrate good performance on very high dimensional problems have not yet been proposed in the literature.

2.1.2 Space Partitioning Algorithms

In space partitioning algorithms the domain D of the function being approximated is divided into a finite number of disjoint subdomains D_i such that $\bigcup D_i = D$. Then, in each subdomain D_i, a regression function is constructed such that:

$$\forall \mathbf{x} \in D_i, \quad y = \hat{f}_i(\mathbf{x}) \tag{2.4}$$

Often, the functions $\hat{f}_i(\mathbf{x})$ are simply constant over the domain D_i. There are two main types of space partitioning algorithms: recursive partitioning or decision tree algorithms, and nearest neighbor algorithms.

Examples of recursive partitioning algorithms or Decision Trees [Michie et al., 1994], include Classification and Regression Trees (CART) [Breiman et al., 1984], and C4.5 [Quinlan, 1993]. Recursive partitioning works as follows: starting with the entire domain D, subdivide it into two daughter subregions such that a parametric regression function (usually chosen to be a single constant value) fitted in each region, reduces the overall approximation error. This process of binary subdivision continues with each daughter region (creating new daughter subregions) until some predefined cost trade-off between approximation error and number of subregions is met. Recursive partitioning algorithms have proven to be very useful, especially for classification problems. In addition, decision trees can be easily interpreted, allowing analysis of how various predictor variables interact with one another. However, because functions are approximated by many discontinuous subregions, recursive partitioning methods tend not to do well on regression problems which are continuous [Friedman, 1991]. This is especially true when there are many predictor variables.

Another notable example of a tree-structured space partitioning architecture is the Hierarchical Mixtures of Experts (HMM) model which uses the Expectation-Maximization (EM) learning algorithm [Jordan and Jacobs, 1994]. This is a parametric approach that uses a fixed number of experts (or parametric building blocks) which are hierarchically combined using gating networks to form the final regression function. In general, each "expert" is trained to work in a particular region of the input space, and the gating networks combine these regions to form a global approximation. This HMM model has been successfully applied to many learning problems, however, because it uses a parametric learning algorithm, it is not directly applicable to the nonparametric regression problems studied in this thesis.

In the simplest implementation, K-Nearest Neighbor algorithms work as follows: given a point $\mathbf{x}$, find the K points in the learning set which are nearest to this point, according to some distance metric. Then the regression estimate at $\mathbf{x}$ becomes the average (perhaps a weighted average) response value of these K-Nearest Neighbors [Michie et al., 1994]. A criticism of Nearest Neighbor algorithms has been that they often exhibit poor generalization [Lowe, 1995]. This problem is mainly due to a poor choice of distance metrics (or kernels). Recently, algorithmic improvements have been suggested which allow the learning data to define the most appropriate distance metric [Lowe, 1995] [Friedman, 1994a] [Hastie and Tibshirani, 1996], the result being much better generalization, especially for classification problems. However, as with regression trees, because functions are approximated by many discontinuous subregions, Nearest Neighbor algorithms have yet to demonstrate good performance on regression problems which are very high dimensional (greater than 100 inputs) and continuous.

2.2 Regression Model Variance Reduction Techniques

For any given set of learning data, there are infinitely many regression models which fit this data exactly. An example of this is given in Figure 2.1, where the 5 data points are fitted exactly with 3 different model types: the piece-wise *linear fit* model, the *smooth fit* model (smoothness here refers to how rapidly the model varies within its domain rather than how many times it is differentiable) and the *high variance fit* model. These 3 models all describe the learning data exactly, however, they interpolate between the data points in very different ways. Most likely, at most one of these models most appropriately describes the target function that generated the data. In fact, only a small subset of all models which fit any given training data accurately, also effectively describe the target function that actually generated the data. Thus, one of the more difficult problems in nonparametric regression is choosing the most appropriate regression model. The problem becomes even more difficult when the training data is noisy, in which case, exactly modeling the learning data can lead to greater approximation errors. This makes the regression model search space even larger because, in addition to models which exactly fit the training data, the search space is extended to include those models which closely fit the training data. As an example of this, the *smooth approximation* model in Figure 2.1 would be just one model which can be used to model the *noisy* sample points. Any number of small variations to this smooth approximation of the sample points could equally well fit the data points.

A model with high variance is one which is not stable with respect to which set of training samples are used to construct it. The more variance a model has, the more sensitive it is to small changes in learning data (this notion of model variance is related to model stability as define in Breiman, 1996b). If a model has high variance then two different sets of learning data randomly generated by the same target function, can produce two very different regression models. Often, as with the high variance fit model in Figure 2.1, both models exactly fit their respective learning data, however neither interpolates effectively between data points (this is often referred to as model *over-fitting*). Thus, an important goal in regression is to reduce model variance. For most regression problems, failing other evidence the model with least variance is often the one which

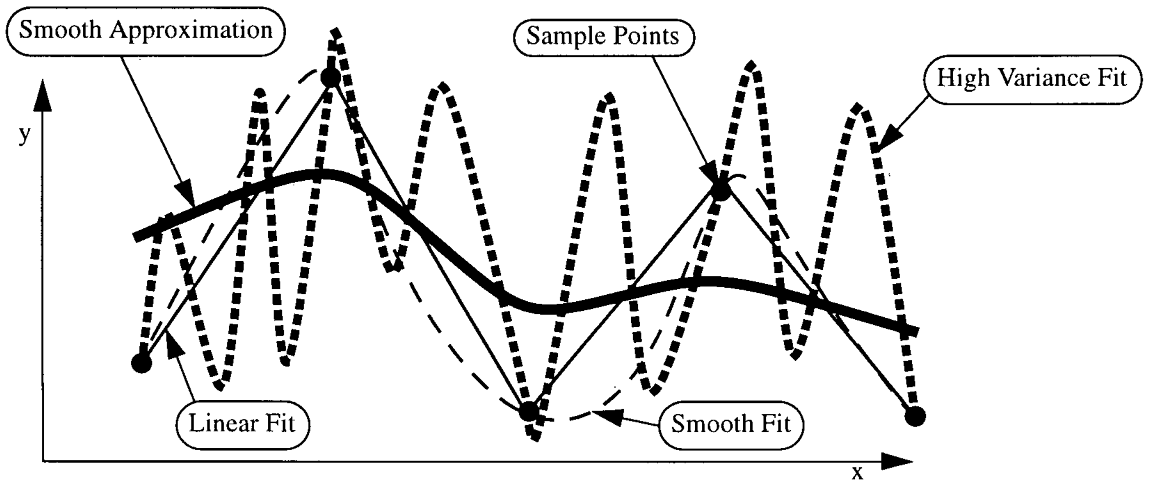


Figure 2.1 Model Variance

most smoothly interpolates through data points (for example the smooth fit model in Figure 2.1).

The variance reduction problem is related to the bias-variance trade-off [Friedman, 1994b], which divides the approximation error into a bias term and a variance term. The bias term defines how closely the regression model fits the learning data, and the variance term defines how much variance there is in the regression model. For example, the three models in Figure 2.1 that fit the data exactly, all have a zero bias error, but have varying degrees of variance. Conversely, the smooth approximation model in Figure 2.1 has a nonzero bias error because it doesn't pass exactly through sample points, however, if the data is assumed to be noisy, its variance error may be the smallest of all three models. Thus, the smooth approximation model may be the most appropriate model when data is noisy.

A variety of techniques have been developed for nonparametric model variance reduction. In this thesis we group these methods into two categories: *penalty based variance reduction* and *averaging based variance reduction*. These are briefly discussed below.

Methods which use penalties for variance reduction work by giving a specific cost to an increase in model complexity. The basic reason for this is that an increase in model complexity (i.e. an increase in the number of model parameters) often means an increase in model variance. Thus, the trade-off between the error reduction obtained by the addition of new model terms, and the increase in complexity due to the addition of these new terms, must be analyzed to determine if this increase in model complexity is justified. Examples of these types of penalty based variance reduction techniques include the *generalized cross validation* criteria used in MARS

[Friedman, 1991], and the *minimum description length* criteria used in SONN [Tenorio and Lee, 1990]. Generalized cross validation works by multiplying the average squared residual error of the training data by a penalty factor which increases as terms are added to the MARS model. Similarly, minimum description length attempts to find the smallest model which effectively fits the training data. Other examples of penalty based variance reduction techniques can be found in [Friedman, 1994b]. One key advantage of variance reduction techniques which use penalties, is that a single regression model is constructed, thus allowing for potentially useful model interpretation. However, one of the key problems associated with penalty based variance reduction techniques is that they are inherently unstable. Breiman [Breiman, 1996b] has shown that this form of model selection is unstable because small changes in learning data can lead to large changes in model structure, making it difficult to choose the most appropriate model.

In contrast to penalty based variance reduction techniques, averaging methods work by forming ensembles of many different regression functions, and modeling a final predictor value as a weighted average of all the regression functions. One of the first noted examples of this is Stacked Generalization [Wolpert, 1992] where potentially nonlinear combinations of regression functions, each trained on some subset of the total data, are used to form the overall regression function. This idea is simplified by Breiman in Stacked Regression [Breiman, 1996c], where the stacked regression functions are simply weighted by non-negative real values. Theoretical justification for such weighted averaging is given by Perrone [Perrone, 1993], where derivations are given for the optimal weighted average of a fixed number of regression functions. In addition, Breiman has shown that averaging can effectively stabilize predictors which are inherently unstable [Breiman, 1996b].

One example of an averaging variance reduction technique is Breiman's bootstrap aggregation or Bagging [Breiman, 1996a]. Bagging uses the average value of many regression functions, each trained using a bootstrap sample of the learning data, to represent the final regression function. The key idea here being that the bootstrap samples tend to create regression functions which have uncorrelated error residuals, thus averaging their values tends to cancel out approximation errors, giving potentially better regression functions when compared with a single regression. Bagging has proven to be an effective variance reduction technique, and variations of it have been successfully applied to many types of regression problems [Parmanto et al., 1996]. However, one significant drawback of all averaging methods is that they do not easily allow for interpretation of the resulting regression model.

2.3 Convergence Results in Nonparametric Regression

In this section, we consider three main types of known convergence results for nonparametric regression. First we briefly discuss some known results in representability, or what function class a given regression function is theoretically able to converge to. Second, we look at two types of rate of convergence results. The first measures how, given an infinite number of learning sample points, the approximation error decreases with the size of the regression model. The second measures how the approximation error decreases as the number of sample points of the target function increases. Examples of these types of convergence results are given below.

Given a particular regression function, one of the first theoretical questions which is asked is *what class of functions can it represent*? Indeed, the most common question posed is whether the regression model is a universal approximator; i.e. whether it is able to form arbitrarily good approximations of bounded continuous functions defined over a finite domain. Examples of universal approximators include polynomials [Lorentz, 1986], radial basis functions [Girosi, 1994], artificial neural networks [Cybenko, 1989] [Hornik et al., 1989], space partitioning techniques [Breiman et al., 1984], projection pursuit [Huber, 1985], and wavelets [Sjoberg et al., 1995] [Juditsky et al., 1995]. These representation results are important because they establish what the respective regression functions are capable of representing. However, representation results in themselves do not establish the rate at which these regression functions converge to the target function.

The first type of rate of convergence result we consider is one which measures the decrease in approximation error as the number of parameters in the regression function increases. From a practical standpoint, this rate of convergence measure tells how the size of the regression model depends on the dimension of the target function. In particular, from the standpoint of high dimensional regression, we can ask the following question: "under what conditions is the rate of convergence of the regression model independent of dimension?" A general answer to this question is the following: by limiting the class of target functions which one is attempting to approximate, many regression functions have a rate of convergence which is independent of dimension [Girosi, 1994]. This includes radial basis functions [Girosi, 1994], artificial neural networks [Barron, 1993], hinging hyperplanes [Breiman, 1993], and projection pursuit

[Jones, 1992]. All of these theoretical results demonstrate rates of convergence independent of dimension are based on a lemma which can be jointly attributed to Muarey [Barron, 1993], Jones [Jones, 1992] and Barron [Barron, 1993] [Girosi, 1994]. In all these results, the class of target functions is limited to those functions for which the magnitude of the first moment of the Fourier transform is finite and independent of dimension. In addition, all of these results are based on *greedy* algorithms which search an infinite function space for the globally optimal regression function. Because of this large search problem, it is not clear how computationally practical these convergence results are [Barron, 1993] [Breiman, 1993] [Juditsky et al., 1995].

The second rate of convergence result we discuss here is how approximation error decreases with the number of sample points of the target function. Stone [Stone, 1982] showed that, in the general case of a target function which is at least once differentiable, the optimal rate of convergence as a function of the number of sample points, does depend on dimension. In fact, this dependence is significant enough that general very high dimensional nonparametric regression requires an impractical number of sample points (i.e. the curse of dimensionality). Once more, by putting more limits on the target function, it is possible to improve on these rates. There are examples in the literature which demonstrate function spaces, and corresponding regression functions, which have rates of convergence that are independent of dimension. One example of this is the projection pursuit regression function when the target function is limited to the class of projection pursuit regression functions with specific constraints [Chen, 1991a]. A second example is an interaction spline model, where the target function is sufficiently differentiable [Chen, 1991b]. Such theoretical results are useful because they can lead to practical implementations which are effective in very high dimensional spaces.

For further discussions on rate of convergence results, see Section 5.3 of Chapter 5.

2.4 Comparing Existing Algorithms to the SPORE

In this chapter, many different approaches to nonparametric regression have been discussed. In addition, we have discussed the concept of model variance reduction and how it is done in the literature. Finally, we have outlined some of the basic theoretical results in nonparametric regression. In this section, we compare SPORE to other published regression methodologies. First we outline which existing learning algorithms most closely resemble SPORE-type learning algorithms, and how they are fundamentally different. Then we discuss how the variance reduction methodology of SPORE relates to those of existing variance reduction techniques. Finally, this section concludes with a brief description of how the theoretical properties of SPORE differ from those of other published regression functions.

The two types of regression functions which are most closely related to the regression functions studied in this thesis (SPORE), are Cascade-Correlation [Fahlman and Lebiere, 1990], and GMDH [Ivankhnenko, 1971] type algorithms such as ASPN [Elder and Brown, 1995] and SONN [Tenorio and Lee, 1990]. All of these algorithms build successively on structures, using the outputs of previously constructed structural units as inputs to new units. However, from the conceptual standpoint of how units are added, and how their inputs are chosen, SPORE differs significantly from these regression functions. These regression functions attempt to build an "optimal" structure by searching for the "best" set of functional units, and/or the best set of inputs to each structural unit. As argued in Section 1.2.2 of Chapter 1, such searches are not computationally feasible in very high dimensional spaces. In the SPORE methodology, this philosophical perspective of searching for optimal (or close to optimal) structures is abandoned because our aim is specifically to address the very high dimensional regression problem. Instead, SPORE algorithms spend minimal computational effort on searches in function space and input space, concentrating more on constructing structures which produce robust approximations without the need for costly search strategies.

From the standpoint of how variance reduction is achieved in the SPORE methodology, we refer the reader to the specific example of SPORE-1 in Chapter 3, and the related discussion in Section 3.3.3 of that chapter. In comparison to other algorithms, SPORE-1 achieves effective variance reduction using a methodology which is similar to Bagging [Breiman, 1996a], but does so with only a single regression function, instead of an aggregate ensemble of regression functions. This is accomplished by constructing each structural unit using a bootstrap sample of the training data. The output of SPORE-1 is the weighted average (each unit is weighted proportionally to the inverse of the squared residual error it has on the training data) of each of these simple structural units. This produces an effective variance reduction methodology (here termed internal variance reduction) which differs significantly from other proposed averaging methods.

In Section 5.3 of Chapter 5, the rate of convergence (as a function of the number of sample points) results for the general SPORE-type learning algorithms are given. The rate of convergence is shown to be independent of dimension, but on different function spaces from those studied in the past. Instead of putting greater limits on the function type [Chen, 1991a] or requiring it to be more differentiable [Chen, 1991b], we assume a more general class of bounded continuous functions, which are at least once differentiable. We obtain a rate of convergence results independent of dimension by limiting the target function's domain of definition in specific ways. Because of this, regression functions based on SPORE have very different convergence properties when compared with other published results.

Chapter 3: The SPORE-1 Regression Function

The general SPORE methodology outlined in Chapter 1 can be implemented in a variety of ways. In this chapter, the simplest implementation of SPORE, termed SPORE-1, is defined and analyzed. In Section 1.3 of Chapter 1, an intuitive justification for the SPORE-1 structure was presented. In this chapter, we present a theoretical analysis of SPORE-1, with particular emphasis on rate of convergence and complexity properties. In Section 3.1 we examine the representation capabilities of the SPORE-1 regression function. In Section 3.2, the SPORE-1 learning algorithm, which is used to construct the regression function, is defined. In Section 3.3 an analysis of the theoretical properties of the SPORE-1 learning algorithm is given. Finally, a summary of the main results of this Chapter is given in Section 3.4.

3.1 SPORE-1: A Cascade of Two Dimensional Functions

The SPORE-1 structure is a cascade of two dimensional functions, with the output of each level of the cascade feeding directly to one of the inputs of the next level. This is diagrammatically represented in Figure 3.1. In the following, a complete description of this basic structure is given, followed by an analysis of the class of functions which can be represented by this SPORE-1 structure.

Let $R_L(x_1, \ldots, x_N)$ be an N dimensional SPORE-1 regression function constructed using an L level cascade of 2 dimensional functions. Thus, $y = R_L(\mathbf{x})$ estimates the dependent variable y given the independent variables $\mathbf{x} = (x_1, \ldots, x_N)$, and has the following form:

$$R_L(x_1, \ldots, x_N) = \alpha_1 \cdot g_1(x_{k_0}, x_{k_1}) + \alpha_2 \cdot g_2(g_1, x_{k_2}) + \sum_{l=3}^{L} \alpha_l \cdot g_l(g_{l-1}, x_{k_l}) \tag{3.1}$$

where L defines the number of levels in the cascade or, equivalently, the number of 2 dimensional functions $g_l(\cdot)$; α_l are real valued scaling factors; and the subscripts $k_0, \ldots, k_L \in \{1, \ldots, N\}$ serve to identify the input variables $(x_1, \ldots, x_N)$.

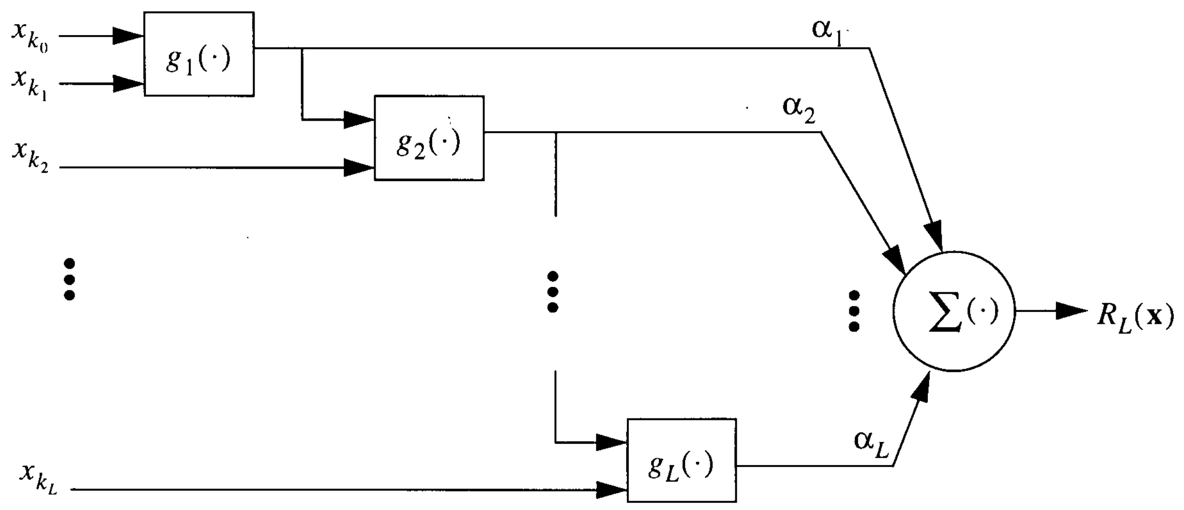


Figure 3.1 The SPORE-1 Regression

Given the above definition, the question we pose is what class of functions can be represented by the simple cascade given in (3.1)? The answer to this question depends on what class of 2 dimensional functions $g_l(\cdot)$ are used in the cascade. In the following, two classes of 2 dimensional functions are considered. First we consider the case of $g_l(\cdot)$ belonging to the class of bounded continuous 2 dimensional functions defined on a closed domain in $\Re^2$. In the second case $g_l(\cdot)$ is limited to the class of 2 dimensional polynomials of finite order greater than 1. The interesting conclusion is that in both instances, the cascade in (3.1) is shown to be a universal approximator: i.e. able of approximating any bounded continuous multidimensional function defined over a finite domain.

3.1.1 A Cascade of Bounded Continuous 2 Dimensional Functions

In the following theorem, it is assumed that the 2 dimensional functions $g_l(\cdot)$ belong to the class of 2 dimensional bounded continuous functions. Under these conditions, it is shown that any continuous, bounded N dimensional function, defined over a closed domain, can be represented using a cascade of $2N^2 + 3N$ $g_l(\cdot)$ functions. Since this is the broadest class of bounded continuous $g_l(\cdot)$ functions, this result defines a bound on the number of cascade levels necessary to exactly represent a bounded continuous multidimensional function. Other smaller classes of bounded continuous functions, such as polynomials, will require more functions.

THEOREM 1. *Let $f(x_1, \ldots, x_N)$ be a bounded continuous function defined on a closed* *domain in* $\Re^N$, $N > 2$. *Then, there exist* $2N^2 + 3N$ *bounded continuous* $g_l(\cdot)$ *functions which are at most 2 dimensional, and* $2N^2 + 3N$ *real valued numbers* α_l, *such that:*

$$f(x_1, \ldots, x_N) = \alpha_1 \cdot g_1(x_{k_0}, x_{k_1}) + \alpha_2 \cdot g_2(g_1, x_{k_2}) + \sum_{l=3}^{2N^2+3N-2} \alpha_l \cdot g_l(g_{l-1}, x_{k_l}) \tag{3.2}$$

Proof: The proof is based on Kolmogorov's superposition theorem [Kolmogorov, 1957], which states that any bounded continuous N dimensional function, $f(x_1, \ldots, x_N)$, defined on a closed domain, can be represented as:

$$f(x_1, \ldots, x_N) = \sum_{q=0}^{2N} \psi_q\left(\sum_{p=1}^{N} \phi_{qp}(x_p)\right) \tag{3.3}$$

where the 1 dimensional functions ψ_q and ϕ_{pq} are bounded and continuous (for more recent versions of this theorem see [Lorentz, 1986]). Thus, for our proof we must show that (3.2) and (3.3) are equivalent for $N > 2$. We show this by equating all $g_l(\cdot)$ functions in (3.2) to the appropriate functions in (3.3). First make the following assignment:

$$g_1 = \phi_{01}(x_1) + \phi_{02}(x_2), \qquad \alpha_1 = 0 \tag{3.4}$$

Then, for $l \in \{2, \ldots, N-1\}$ and $p = l + 1$, let

$$g_l = g_{l-1} + \phi_{0p}(x_p), \qquad \alpha_l = 0 \tag{3.5}$$

Thus, completing the definitions to level $l = N$, let

$$g_N = \psi_0(g_{N-1}), \qquad \alpha_N = 1 \tag{3.6}$$

Therefore, N levels are required to build the $q = 0$ part of (3.3). What remains are the $q = 1, \ldots, 2N$ parts to be constructed. Consider the $q = 1$ part. For level $l = N + 1$, let

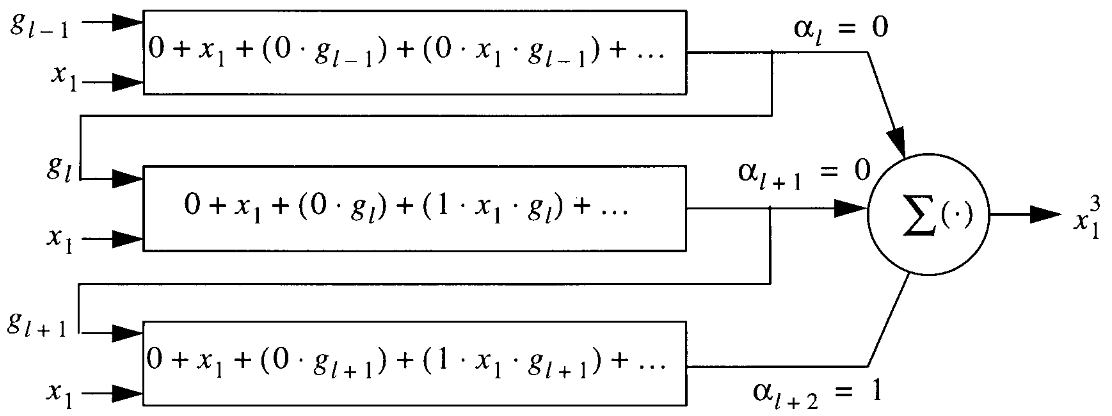
$$g_{N+1} = \phi_{11}(x_1), \qquad \alpha_{N+1} = 0 \tag{3.7}$$

Then, for $l \in \{N + 2, \ldots, 2N\}$ and $p = l - N$, let

$$g_l = g_{l-1} + \phi_{1p}(x_p), \qquad \alpha_l = 0 \tag{3.8}$$

Thus, completing the definitions to level $l = 2N + 1$, let

$$g_{2N+1} = \psi_1(g_{2N}), \qquad \alpha_{2N+1} = 1 \tag{3.9}$$



g_{l-1}
x_1
$0 + x_1 + (0 \cdot g_{l-1}) + (0 \cdot x_1 \cdot g_{l-1}) + \ldots$
$\alpha_l = 0$
g_l
x_1
$0 + x_1 + (0 \cdot g_l) + (1 \cdot x_1 \cdot g_l) + \ldots$
$\alpha_{l+1} = 0$
$\sum(\cdot)$
x_1^3
g_{l+1}
x_1
$0 + x_1 + (0 \cdot g_{l+1}) + (1 \cdot x_1 \cdot g_{l+1}) + \ldots$
$\alpha_{l+2} = 1$

Figure 3.2 A Cascade Representation of a Polynomial Term

Therefore, $2N + 1$ levels are required to build the $q = 0$ and $q = 1$ parts of (3.3). What remains are the $q = 2, \ldots, 2N$ parts to be constructed. The construction of these parts is identical to that of $q = 1$, which took $N + 1$ levels of (3.2). Thus, the total number of levels required is $2N(N+1) + N = 2N^2 + 3N$. This completes the proof of THEOREM 1. ***Q.E.D.***

THEOREM 1 establishes a bound on the sufficient number of levels required to exactly represent any bounded continuous function defined over a finite domain in 3 or more dimensions. However, the only restriction on the functions $g_l(\cdot)$ is that they be some 2 dimensional function. In addition, the proof of THEOREM 1 is based on Kolmogorov's superposition theorem, and it is a known result that the 1 dimensional functions used in Kolmogorov's theorem are often highly non-smooth [Vitushkin, 1954] [Vitushkin and Henkin, 1967] and therefore not easy to represent in practice.

3.1.2 A Cascade of Finite Order 2 Dimensional Polynomials

In this section the functions $g_l(\cdot)$ are further restricted to belong to the class of finite order polynomials. Thus THEOREM 2 deals with a more practical scenario in that the functions $g_l(\cdot)$ are of a class which makes them easier to use in practice.

THEOREM 2. *Let $f(x_1, \ldots, x_N)$ be a continuous function defined on a compact subset $A \subset \Re^N$. Let $g_l(\cdot)$ be a 2 dimensional polynomial of finite order greater than 1. Then, for each*

$\varepsilon > 0$, *there exists a finite number,* L, *of functions* $g_l(\cdot)$ *and real numbers* α_l, *such that*

$$|f(x_1, \ldots, x_N) - R_L(x_1, \ldots, x_N)| < \varepsilon \tag{3.10}$$

where $R_L(\mathbf{x})$ *is defined in* (3.1).

Proof: It is a known result that a continuous function $f(x_1, \ldots, x_N)$ defined over a compact subspace can be approximated by polynomials in $x_1, \ldots, x_N$ (see Theorem 6 on page 10 in [Lorentz, 1986] for a statement of the appropriate theorem). This representation has the form

$$\hat{f}(x_1, \ldots, x_N) = \sum_{i_1, \ldots, i_N \in \{0, 1, 2, \ldots\}} b_{i_1, \ldots, i_N} \cdot x_1^{i_1} \cdots x_N^{i_N} \tag{3.11}$$

where $b_{i_1, \ldots, i_N}$ are polynomial coefficients. Thus, we shall prove THEOREM 2 by showing that any polynomial in $x_1, \ldots, x_N$ can be represented by the cascade structure defined in (3.1), where the $g_l(\cdot)$ are 2 dimensional polynomials of finite order greater than 1. By this constraint, the simplest form of functions used in the cascade is:

$$g_l(u, v) = a_{00} + a_{10}u + a_{01}v + a_{11}uv + a_{20}u^2 + a_{02}v^2 \tag{3.12}$$

This and every other 2 dimensional polynomial of order greater than 1 must at least have the terms $a_{10}u$ and $a_{11}uv$. Given these two terms, it is simple to show that any term in (3.11) can be equated to the cascade structure (3.1). As an example, consider the polynomial term $b_{335} \cdot x_1^3 x_2^3 x_3^5$. We first construct the x_1^3 part of this term as outlined in Figure 3.2. Starting at any arbitrary level l, set $\alpha_l = 0$, $x_{k_l} = x_1$, $a_{10} = 1$, and all other parameters of $g_l(\cdot)$ to zero. Then, for levels $\rho = l + 1$ and $\rho = l + 2$ set $\alpha_\rho = 0$, $x_{k_\rho} = x_1$, $a_{11} = 1$, and all other parameters of $g_\rho(\cdot)$ to zero. This gives us the x_1^3 part of $b_{335} \cdot x_1^3 x_2^3 x_3^5$. Next, we construct the x_2^3 part as follows. For levels $\rho = l + 3$ to $\rho = l + 5$ set $\alpha_\rho = 0$, $x_{k_\rho} = x_2$, $a_{11} = 1$, and all other parameters of $g_\rho(\cdot)$ to zero. Next, for the x_3^5 part, for levels $\rho = l + 6$ to $\rho = l + 9$ set $\alpha_\rho = 0$, $x_{k_\rho} = x_3$, $a_{11} = 1$, and all other parameters of $g_\rho(\cdot)$ to zero. Finally, for level $\rho = l + 10$, set $\alpha_\rho = b_{335}$, $x_{k_\rho} = x_3$, $a_{11} = 1$, and all other parameters of $g_\rho(\cdot)$ to zero. This gives us the completed term $b_{335} \cdot x_1^3 x_2^3 x_3^5$. In a similar manner, we can construct any other term in the polynomial (3.11). Hence, given that we can construct any polynomial term using sections of the cascade structure in (3.1), and because (3.1) is a sum of such sections, it follows that any polynomial structure can be represented by the cascade structure defined in (3.1). This completes the proof of

THEOREM 2. ***Q.E.D.***

3.2 Learning Algorithm for the SPORE-1 Regression Function

In the previous section we have shown that the SPORE-1 regression function is a universal approximator. However, we have not shown how to construct the functions $g_l(\cdot)$ and real numbers α_l. One such algorithm is presented next.

3.2.1 The SPORE-1 Learning Algorithm

We assume that M_L learning input/output examples $(\mathbf{x}_q, y_q) \in \Gamma_L$ have been divided into 2 sets: a training set Γ_T containing M_T input/output pairs, and a validation set Γ_V containing M_V input/output pairs. In Chapter 4, various methodologies are described for dividing learning data into training and validation sets. There are 3 preset learning parameters, symbolized by $depth_1$, $depth_2$, and ε. The function of each of these parameters is explained in the following algorithmic description, and is further elaborated on below. A single cascade of $g_l(\cdot)$ functions is constructed in a series of sections. A section of a cascade between levels i and j, symbolized by ${}^{i}R_j$, is defined as follows:

$$ {}^{i}R_j = \sum_{l=i}^{j} \alpha_l \cdot g_l(\cdot) \tag{3.13}$$

When the section being constructed can no longer reduce mean squared error, the learning outputs y are replaced by residual errors due to ${}^{i}R_j$ and a new section is begun starting from the last error reducing function $g_j(\cdot)$. Throughout the remainder of the chapter we assume that the functions $g_l(\cdot)$ are 2 dimensional polynomials of finite order:

$$ g_l(u, v) = \sum_{i,\, j \in \{0, 1, 2, \dots\}}^{i + j \le \kappa} a_{ij} u^i v^j \tag{3.14}$$

where κ is the order of the polynomial, and a_{ij} are its coefficients. Stated in detail, the construction of the regression functions, $R_L(\mathbf{x})$, proceeds as follows (note that each SPORE-1 regression function has only one output, so for problems with multiple outputs, one regression function must be constructed for each output):

Algorithm 1: SPORE-1

STEP 1: *Initialize algorithm:* Initialize the 2 counters $i = 1$ and $p = 1$, where i and p are used as follows: i indexes the function $g_i(\cdot)$ at the start of the current section; p is a subscript indicating that section p of the cascade will be constructed in step 2. Throughout the algorithm the subscripts $k_0, \ldots, k_L$ (used to identify the independent variables) are randomly selected, one at a time, from the set $\{1, \ldots, N\}$, *without* replacement. When the set is empty, it is re-initialized to $\{1, \ldots, N\}$, and the process of selecting these subscripts continues.

STEP 2: *Construct new section:* Construct, in order, the functions $g_i(\cdot), g_{i+1}(\cdot), \ldots, g_j(\cdot)$, where $j - i > depth_l$, as follows:

1. Obtain a bootstrap sample set Γ_T^B (i.e. M_T randomly chosen samples with replacement) from the training set Γ_T.

2. Given Γ_T^B, using singular value decomposition, determine the model parameters $\{a_{\rho\sigma}\}$ of the function $g_i(\cdot)$ which minimize the following least squared error criteria:

$$\{a_{\rho\sigma}\} = \operatorname{argmin}_{\{a_{\rho\sigma}\}}\left[\sum_{q \in \Gamma_T^B} (g_i(\cdot) - y_q)^2\right] \tag{3.15}$$

The construction stops with function $g_j(\cdot)$ when the change in cascade error on the validation set Γ_V, over the past $depth_l + 1$ level additions, is less than some small, positive number ε. This is defined as follows:

$$1 - \left(\frac{MSE({}^iR_j)_{\Gamma_V}}{\frac{1}{depth_l}\left(\sum_{q = j-1-depth_l}^{j-1} MSE({}^iR_q)_{\Gamma_V}\right)}\right) \le \varepsilon \tag{3.16}$$

where ${}^iR_q = \sum_{l=i}^{q} \alpha_l \cdot g_l(\cdot)$ is the constructed cascade between levels i and q; and $MSE({}^iR_q)_{\Gamma_V}$ is the mean squared error of this cascade on the validation set Γ_V. With the addition of each new level, the scaling factors α_l, for $l = i, \ldots, j$, are recomputed as follows:

$$\alpha_l = \frac{(MSE(g_l(\cdot))_{\Gamma_T})^{-1}}{\sum_{q=i}^{j} (MSE(g_q(\cdot))_{\Gamma_T})^{-1}} \tag{3.17}$$

where $MSE(g_l(\cdot))_{\Gamma_T}$ is the mean squared error of the single function $g_l(\cdot)$ on the training set Γ_T. Hence, α_l is simply the inverse of the normalized mean squared error of level l, and its purpose is to give more weight to levels which contribute more significantly to error reduction.

STEP 3: *Prune section:* Next prune the current section back to the level which has the best mean squared error as follows. Find the *minimum* level $s \in \{i-1, i, \ldots, j\}$ which gives the least mean squared error of the cascade on the validation set Γ_V, in the following sense:

$$\text{choose } s < t \text{ s.t. } \forall(t \in \{i, \ldots, j\}),\ MSE({}^{i}R_s)_{\Gamma_V} \le MSE({}^{i}R_t)_{\Gamma_V} \tag{3.18}$$

Delete all functions $g_{s+1}(\cdot), g_{s+2}(\cdot), \ldots, g_j(\cdot)$. Set $j = s$ and, using (3.17), recalculate the scaling factors α_l, for $l = i, \ldots, j$. The current state of the cascade is now given by:

$$R_s(\mathbf{x}) = \sum_{l=1}^{s} \alpha_l \cdot g_l(\cdot) \tag{3.19}$$

Note that if $s = i-1$ then all of the functions just constructed in step 2 are deleted.

STEP 4: *Update learning outputs:* Update the output values, y_i, of both the validation set Γ_V and the training set Γ_T as follows:

$$\forall((\mathbf{x}, y) \in \Gamma_V \cup \Gamma_T) \Rightarrow y = y - \sum_{l=i}^{s} \alpha_l \cdot g_l(\cdot) \tag{3.20}$$

The sets Γ_V and Γ_T now contain the residual approximation errors after the cascade has been constructed to pruned level s. Let ρ_p be the variance of the output values, y_i, of the validation set Γ_V (as defined in step 1, the index p refers to the section just constructed). Thus, ρ_p is equivalent to the mean squared error of the approximation $R_s(\mathbf{x})$ on the validation set Γ_V. ρ_p is used in **STEP 5** to establish a stopping condition under which the construction of the cascade stops.

STEP 5: *Check stopping condition:* Go back to **STEP 2**, constructing at least $depth_2 + 1$ sections, until the following condition is met:

$$\left(1 - \frac{\rho_p}{\frac{1}{depth_2}\left(\sum_{q = p-1-depth_2}^{p-1} \rho_q\right)}\right) \le \varepsilon \qquad (3.21)$$

Therefore, the algorithm will terminate when the mean squared error fails to improve by a factor of ε, over the last $depth_2 + 1$ sections. Before executing **STEP 2**, set $i = s + 1$ and increment the section counter p to initialize the next section. Upon termination, set $L = s$, the total number of levels constructed. This completes the description of the algorithm.

As indicated above, the learning parameters $depth_1$, $depth_2$, and ε, are all used to define conditions under which the construction of the cascade stops. In particular, ε defines the relative decrease in error required to justify the addition of more $g_l(\cdot)$ functions to the cascade. The learning parameter $depth_1$ defines how many levels of the cascade are used to determine the termination conditions of **STEP 2**. Similarly, $depth_2$ and ε are used in **STEP 5** to define how many times **STEP 2** is executed before the algorithm is executed. The theoretical significance of these learning parameters is discussed in Section 3.3.1. In Chapter 4, the actual setting of these values is discussed.

3.3 Theoretical Analysis of the SPORE-1 Learning Algorithm

In this section we analyze some theoretical properties of the SPORE-1 learning algorithm. In Section 3.3.1 we give convergence and rate of convergence results for the algorithm. In Section 3.3.2 we give complexity results. In Section 3.3.3, we discuss the internal variance reduction property of SPORE-1. And in Section 3.3.4 we define the virtual space partitioning property of SPORE-1.

3.3.1 Convergence Results

We begin the theoretical analysis of SPORE-1 by analyzing its convergence properties. In THEOREM 3 we prove that, under appropriate regularity assumptions, the SPORE-1 algorithm monotonically converges to some finite approximation error. In THEOREM 4 we give sufficient conditions for the approximation error to converge to zero. In THEOREM 5 we extend this result by showing that the rate (as a function of the number of learning sample points) at which the learning algorithm converges to this approximation error, is independent of the dimension of the dimension of the target function. The proofs of THEOREM 3, THEOREM 4 and THEOREM 5 are based on 4 lemmas which we state and prove below.

The reader should note that in the following, the learning data Γ_L always refers to the *updated learning data* or residuals as defined in **STEP 4** of the learning algorithm.

In the first lemma, we show that the approximation error on any bootstrap sample used in the construction of a function $g_l(\cdot)$, must either decrease or stay the same; it can never increase.

LEMMA A. *For all $l \geq 1$, the following is true (note that the subscript q serves as an index for the summation $\sum$):*

$$\sum_{\Gamma_T^B}(g_{l+1}(g_l, x_{k_{l+1}}) - y_q)^2 \leq \sum_{\Gamma_T^B}(g_l(\cdot) - y_q)^2 \tag{3.22}$$

Proof: From **STEP 2** of the SPORE-1 learning algorithm, the parameters of $g_{l+1}(\cdot)$ are chosen as follows:

$$\{a_{ij}\} = \operatorname{argmin}_{\{a_{ij}\}}\left[\sum_{\Gamma_T^B}(g_{l+1}(u, v) - y_q)^2\right] \tag{3.23}$$

Given that $u = g_l$ and $v = x_{k_{l+1}}$ we could set coefficients of all terms with $x_{k_{l+1}}$ to zero if they increase error. Therefore, the error can never increase. Similarly, error will decrease if and only if $v = x_{k_{l+1}}$ contributes to error reduction within the parametric structure of $g_{l+1}(\cdot)$. Thus, LEMMA A is true by definition. ***Q.E.D.***

The implication of LEMMA A is that the mean squared error of any new level $MSE(g_{l+1}(\cdot))_{\Gamma_T^B}$, on any bootstrap sample Γ_T^B, is as least as small as the mean squared error of the previous level. Thus, even if $v = x_{k_{l+1}}$ does not contribute to the lowering of error at level $l+1$, the parametric form of $g_{l+1}(\cdot)$ ensures that for any bootstrap Γ_T^B, the previous best error is maintained.

Next we examine how the expected error, on the entire learning data set Γ_L, behaves with the addition of a new function $g_{l+1}(\cdot)$ to the cascade. This requires us to make some assumptions about the function which originally generated the learning data.

Assumption A: *We will assume that all the learning data Γ_L, and therefore all of the data in the training set Γ_T and the validation set Γ_V, are independently generated by some unknown function:*

$$y = f(\mathbf{x}) + \varepsilon(\mathbf{x}) \tag{3.24}$$

where $f(\mathbf{x})$ is bounded, continuous and defined over some closed domain $D \subseteq \Re^N$, and $\varepsilon(\mathbf{x})$ is some noise term with the property that $\forall(\mathbf{x} \in D)$, $E(\varepsilon(\mathbf{x})) = 0$. Furthermore, we assume that the probability density function which defines the distribution of $\varepsilon(\mathbf{x})$ is continuous, and has finite variance, $\forall(\mathbf{x} \in D)$. Note that one can equivalently define $f(\mathbf{x})$ as follows:

$$f(\mathbf{x}) = E[y|\mathbf{x}] = \int_{-\infty}^{\infty} yh(y|\mathbf{x})\varphi(dy) \tag{3.25}$$

where, $h(y|\mathbf{x})$ is a continuous probability density function with bounded variance, and $\varphi(dy)$ is some measure on $\Re$.

In the following lemma, we use **Assumption A** to show that the expected approximation error on the learning data either decreases or stays the same.

LEMMA B. *Given **Assumption A**, and for sufficiently large sample sizes M_T and M_V, then for all $l \geq 1$, the following is true:*

$$E\left[\sum_{\Gamma_L}(g_{l+1}(g_l, x_{k_{l+1}}) - y_q)^2\right] \leq E\left[\sum_{\Gamma_L}(g_l(\cdot) - y_q)^2\right] \tag{3.26}$$

Proof: From LEMMA A, we know that LEMMA B is true if the learning set Γ_L is equivalent to the bootstrap training set: i.e. $\Gamma_T^B \equiv \Gamma_L$. Hence, it is reasonable to assume that it will also be true if Γ_T^B is a "sufficiently good" representation of the distribution of points in the training set Γ_T and the validation set Γ_V. Thus, we prove this lemma by showing, given **Assumption A**, Γ_T^B becomes sufficiently close (in the probability distribution sense) to Γ_T and Γ_V for large M_T and M_V. Consider any arbitrary point $\mathbf{x}_1 \in D$. From the Central Limit Theorem, we know the following: if $y_1, \ldots, y_n$ are independent random observations from a population with probability function $h(y|\mathbf{x}_1)$ for which the variance $\sigma^2[y]$ is finite, the sample mean $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$ is approximately normally distributed when the sample size is reasonably large, with mean $E[y]$ and variance $(\sigma^2[y])/n$ [Nester et al., 1990]. Therefore, if we limit all learning sample points to be generated from any single point $\mathbf{x}_1 \in D$ and we make M_T and M_V sufficiently large, the following is true due to **Assumption A** and the Central Limit Theorem;

$$E\left[\sum_{\Gamma_L}(g_{l+1}(g_l, x_{k_{l+1}}) - y_q)^2\right] = E\left[\sum_{\Gamma_L}(g_l(\cdot) - y_q)^2\right] \tag{3.27}$$

This is true because, at any single point $\mathbf{x}_1 \in D$, the construction of the function $g_{l+1}(g_l, x_{k_{l+1}})$ as defined in (3.23) is equivalent to finding the mean value of the bias coefficient a_{00}; i.e. $g_{l+1}(g_l, x_{k_{l+1}})$ reduces to a single mean value which is equivalent to the mean of a one dimensional distribution. In addition, assuming that the variable $x_{k_{l+1}}$ does not contribute to reduction of error, then because $g_l(\cdot)$ and $f(\mathbf{x})$ are both bounded and continuous over a finite domain, and because the distribution of $\varepsilon(\mathbf{x})$ is also continuous, it must be the case that (3.27) is also true for any finite local region (within the domain D) about any arbitrary point $\mathbf{x}_1 \in D$. This is true because the continuity constraints on $g_{l+1}(\cdot)$ imply that as any point (u_1, v_1) approaches another point (u_2, v_2), then $g_{l+1}(u_1, v_1)$ must approach $g_{l+1}(u_2, v_2)$; hence, this is analogous to taking an elastic surface and smoothly bending it according to the constraints of (3.23). Furthermore, under the same conditions, because this is true for any arbitrary point $\mathbf{x}_1 \in D$, (3.27) must be true over the entire domain D (i.e. simply stretch the above elastic surface to include all of D). Thus, at minimum, the expected approximation error will remain unchanged. Similarly, due to the construction of $g_{l+1}(g_l, x_{k_{l+1}})$ as defined in (3.23), the condition

$$E\left[\sum_{\Gamma_L}(g_{l+1}(g_l, x_{k_{l+1}}) - y_q)^2\right] < E\left[\sum_{\Gamma_L}(g_l(\cdot) - y_q)^2\right] \tag{3.28}$$

is true if and only if the variable $x_{k_{l+1}}$ serves to reduce the approximation error within the context of the parametric structure of $g_{l+1}(g_l, x_{k_{l+1}})$. This completes the proof of LEMMA B. ***Q.E.D.***

Next we extend the result of the previous lemma to show that the expected approximation error, over the *entire* domain D of the target function, either decreases or stays the same.

LEMMA C. *Let $f_u(\mathbf{x})$ represent the updated error function after the computation of residuals in **STEP 4**. Thus before **STEP 4** is first executed $f_u(\mathbf{x}) = f(\mathbf{x})$, and the execution of **STEP 4**, computes:*

$$f_u(\mathbf{x}) = f(\mathbf{x}) - \sum_{l=i}^{s} \alpha_l \cdot g_l(\cdot), \tag{3.29}$$

and thereafter $f_u(\mathbf{x})$ is updated as follows:

$$f_u(\mathbf{x}) \Leftarrow f_u(\mathbf{x}) - \sum_{l=i}^{s} \alpha_l \cdot g_l(\cdot) \tag{3.30}$$

*Then, given **Assumption A**, and for sufficiently large sample sizes M_T and M_V, for every level $l > 1$, the following is true:*

$$E\left[\int_D (g_{l+1}(g_l, x_{k_{l+1}}) - f_u(\mathbf{x}))^2 dD\right] \le E\left[\int_D (g_l(\cdot) - f_u(\mathbf{x}))^2 dD\right] \tag{3.31}$$

Proof: From **Assumption A** we know that the samples y_q have mean $f_u(\mathbf{x})$. Furthermore, because $f(\mathbf{x})$ and $\sum_{l=i}^{s} \alpha_l \cdot g_l(\cdot)$ are bounded and continuous, then so is $f_u(\mathbf{x})$. The the proof of LEMMA C is identical to the proof of LEMMA B, with the exception that the approximation errors are now integrated over the entire domain D and not simply summed over the learning set Γ_L. ***Q.E.D.***

Next, using the above lemmas, we show that the SPORE-1 learning algorithm must converge to some finite approximation error. It is important to establish that SPORE-1 converges to some error function because this allows us to establish conditions under which the algorithm terminates. Algorithms which do not terminate are rarely useful in regression.

THEOREM 3. *Given **Assumption A**, and for sufficiently large sample sizes M_T and M_V, then for every $\varepsilon > 0$ ($\varepsilon \in \Re$), there exists a level l such that the following is true:*

$$E\left[\int_D (R_{l+p}(\mathbf{x}) - R_l(\mathbf{x}))^2 dD\right] \le \varepsilon \tag{3.32}$$

where $p \ge 1$ is any positive integer (note that this theorem implies a monotonically converging sequence).

Proof: Consider the construction of the function $R_l(\mathbf{x})$ during the first execution of **STEP 2**. Let

$$R_l(\mathbf{x}) = \sum_{p=1}^{l} \alpha_p \cdot g_p(\cdot) \tag{3.33}$$

and

$$R_{l+1}(\mathbf{x}) = \sum_{p=1}^{l+1} \alpha'_p \cdot g_p(\cdot) \tag{3.34}$$

Note that we have put a prime on α'_p to distinguish that they are different from α_p. Therefore,

$$R_{l+1}(\mathbf{x}) - R_l(\mathbf{x}) = \sum_{p=1}^{l+1} \alpha'_p \cdot g_p(\cdot) - \sum_{p=1}^{l} \alpha_p \cdot g_p(\cdot) \tag{3.35}$$

Simplifying we get:

$$R_{l+1}(\mathbf{x}) - R_l(\mathbf{x}) = \alpha'_{l+1} \cdot g_{l+1}(\cdot) - \sum_{p=1}^{l} (\alpha'_p - \alpha_p) \cdot g_p(\cdot) \tag{3.36}$$

where, in the limit of $M_T \to \infty$,

$$\alpha_p \to \frac{\left(\int_D (g_p(\cdot) - f(\mathbf{x}))^2 dD\right)^{-1}}{\sum_{q=i}^{l} \left(\int_D (g_q(\cdot) - f(\mathbf{x}))^2 dD\right)^{-1}} = \frac{\Phi_p}{\sum_{q=i}^{l} \Phi_q} \tag{3.37}$$

and

$$\alpha'_p \to \frac{\left(\int_D (g_p(\cdot) - f(\mathbf{x}))^2 dD\right)^{-1}}{\sum_{q=i}^{l+1} \left(\int_D (g_q(\cdot) - f(\mathbf{x}))^2 dD\right)^{-1}} = \frac{\Phi_p}{\sum_{q=i}^{l+1} \Phi_q} \tag{3.38}$$

where, for all $p > 0$

$$\Phi_p = \left(\int_D (g_p(\cdot) - f(\mathbf{x}))^2 dD\right)^{-1} \tag{3.39}$$

Therefore, we can write $\alpha'_p - \alpha_p$ as

$$\alpha'_p - \alpha_p = \frac{\Phi_p}{\sum_{q=i}^{l+1} \Phi_q} - \frac{\Phi_p}{\sum_{q=i}^{l} \Phi_q} = \frac{\Phi_p\left(\sum_{q=i}^{l} \Phi_q\right) - \Phi_p\left(\sum_{q=i}^{l+1} \Phi_q\right)}{\left(\sum_{q=i}^{l+1} \Phi_q\right)\left(\sum_{q=i}^{l} \Phi_q\right)} \tag{3.40}$$

Simplifying, we get

$$\alpha'_p - \alpha_p = \left(\frac{-\Phi_{l+1}}{\sum_{q=i}^{l+1} \Phi_q}\right)\left(\frac{\Phi_p}{\sum_{q=i}^{l} \Phi_q}\right) = -\alpha'_{l+1}\alpha_p \tag{3.41}$$

because

$$\alpha'_{l+1} = \frac{\Phi_{l+1}}{\sum_{q=i}^{l+1} \Phi_q} \tag{3.42}$$

Therefore, we can simplify (3.36) as follows: substituting $\alpha'_p - \alpha_p$ with $-\alpha'_{l+1}\alpha_p$

$$R_{l+1}(\mathbf{x}) - R_l(\mathbf{x}) = \alpha'_{l+1} \cdot g_{l+1}(\cdot) - \alpha'_{l+1} \sum_{p=1}^{l} \alpha_p \cdot g_p(\cdot) \tag{3.43}$$

Substituting in Equation (3.33) we get:

$$R_{l+1}(\mathbf{x}) - R_l(\mathbf{x}) = \alpha'_{l+1} \cdot g_{l+1}(\cdot) - \alpha'_{l+1} R_l(\mathbf{x}) \tag{3.44}$$

which simplifies to

$$R_{l+1}(\mathbf{x}) - R_l(\mathbf{x}) = \alpha'_{l+1}(g_{l+1}(\cdot) - R_l(\mathbf{x})) \tag{3.45}$$

Finally, from LEMMA C, and assuming that $E[\Phi_l] > 0$ for all l (note that if this assumption is false, then THEOREM 3 is true by default due to LEMMA C), as $l \to \infty$,

$$E[\Phi_{l+1} - \Phi_l] \to 0 \tag{3.46}$$

therefore, we have an infinite sum of a constant, giving infinity, or

$$\sum_{q=i}^{l+1} \Phi_q \to \infty . \tag{3.47}$$

and as a result

$$\alpha'_{l+1} = \left(\frac{\Phi_{l+1}}{\sum_{q=i}^{l+1} \Phi_q}\right) \to 0 . \tag{3.48}$$

Therefore, as $l \to \infty$, Equation (3.45) becomes:

$$E(R_{l+1}(\mathbf{x}) - R_l(\mathbf{x})) \to 0 . \tag{3.49}$$

Thus, THEOREM 3 is true during the first execution of execution of **STEP 2**, and because of the continuity arguments in LEMMA C, the same argument can be made during each subsequent execution of **STEP 2**. This completes the proof of THEOREM 3. ***Q.E.D.***

THEOREM 3 establishes that the SPORE-1 Learning Algorithm converges to some function $R_\infty(\mathbf{x})$ as $l \to \infty$. Thus, for any fixed sequence $k_0, \ldots, k_\infty$, the approximation error converges to the following:

$$\int_D (R_\infty(\mathbf{x}) - f(\mathbf{x}))^2 dD \tag{3.50}$$

In addition, THEOREM 3 allows us to define theoretically optimal values for the learning parameters $depth_1$, $depth_2$, and ε (see **STEP 2** and **STEP 5** of the SPORE-1 Learning Algorithm). First consider ε, which defines a cut-off on the minimum allowable decrease in approximation error. From the proof of THEOREM 3, we know that this rate of decrease monotonically converges to zero as the number of levels grows. Hence, by stopping the algorithm when some ε rate of decrease is reached, we know that any future expected decrease will not exceed this amount. Thus, in order to achieve the error given in (3.50), we must let $\varepsilon \to 0$. In addition, the true rate of potential error reduction is measured only when $p \to \infty$ in (3.32). Therefore, the parameters $depth_1$ and $depth_2$, which are related to p, should also be large in order to achieve (3.50).

In the next theorem, we establish sufficient conditions under which approximation error is reduced when a new level is added to the regression function. In order to do this, we need to make a further assumption as follows.

Assumption B: *For simplicity, we define the following:*

$$\Phi = E\left[\int_D (g_{l+1}(.) - f_u(\mathbf{x}))^2 dD\right] \tag{3.51}$$

$$\Psi = E\left[\int_D (g_{l+1}(.) - f_u(\mathbf{x}))(R_l(\mathbf{x}) - f(\mathbf{x}))dD\right] \tag{3.52}$$

$$\Omega = E\left[\int_D (R_l(\mathbf{x}) - f(\mathbf{x}))^2 dD\right] \tag{3.53}$$

Given the above definitions, the following 3 conditions are assumed to be true. First, the approximation error of $R_l(\mathbf{x})$ is greater than the correlation between the errors due to the new level function $g_{l+1}(\cdot)$ and the approximation error due to $R_l(\mathbf{x})$. This can be stated as follows ($f_u(\mathbf{x})$ is defined in LEMMA C*):*

$$\Omega > \Psi \tag{3.54}$$

In other words, this condition is satisfied when the errors due to $g_{l+1}(\cdot)$ are sufficiently uncorrelated with those due to $R_l(\mathbf{x})$. The second condition requires that the correlation between the errors due to the new level function $g_{l+1}(\cdot)$ and the approximation error due to $R_l(\mathbf{x})$ satisfy the following:

$$\Phi > 2\Psi - \Omega \tag{3.55}$$

Once again, this condition is satisfied when the errors due to $g_{l+1}(\cdot)$ are sufficiently uncorrelated with those due to $R_l(\mathbf{x})$. The final condition which we need to meet is that the scaling factor α'_{l+1} satisfies the following:

$$\alpha'_{l+1} \le \frac{-(\Psi - \Omega) - \sqrt{(\Psi - \Omega)^2 - (\Phi - 2\Psi + \Omega)(1 - \varepsilon)\Omega}}{(\Phi - 2\Psi + \Omega)} \tag{3.56}$$

where $\varepsilon = 1 - \delta$, and $\delta \in \Re$ is an arbitrarily small positive real number defining the amount of error reduction due to the addition of a new level to the regression function. If we let $l \to \infty$, and thus $\alpha'_{l+1} \to 0$ (see proof of THEOREM 3*), this condition is also satisfied when the errors due to $g_{l+1}(\cdot)$ are sufficiently uncorrelated with those due to $R_l(\mathbf{x})$. Hence all 3 conditions require that the approximation errors of a new level are uncorrelated with those of previous levels. As discussed in Section 3.3.3, the SPORE-1 learning algorithm encourages this condition to occur through the use of bootstrap training samples.*

Using **Assumption B**, we now state the following theorem.

THEOREM 4. *Given **Assumption A** and **Assumption B**, and for sufficiently large sample sizes M_T and M_V, then for every $l > 1$ the following is true:*

$$E\left[\int_D (R_{l+1}(\mathbf{x}) - f(\mathbf{x}))^2 dD\right] \le \varepsilon\left(E\left[\int_D (R_l(\mathbf{x}) - f(\mathbf{x}))^2 dD\right]\right) \tag{3.57}$$

*where ε is defined in **Assumption A**.*

Proof: From (3.44) we can write $R_{l+1}(\mathbf{x})$ as follows:

$$R_{l+1}(\mathbf{x}) = \alpha'_{l+1} \cdot g_{l+1}(\cdot) - \alpha'_{l+1} R_l(\mathbf{x}) + R_l(\mathbf{x}) \tag{3.58}$$

This can be written as:

$$R_{l+1}(\mathbf{x}) = \alpha'_{l+1} \cdot g_{l+1}(\cdot) + (1 - \alpha'_{l+1}) R_l(\mathbf{x}) \tag{3.59}$$

In addition, we can write $(R_{l+1}(\mathbf{x}) - f(\mathbf{x}))$ as:

$$(R_{l+1}(\mathbf{x}) - f(\mathbf{x})) = \alpha'_{l+1}(g_{l+1}(\cdot) - f_u(\mathbf{x})) + (1 - \alpha'_{l+1})(R_l(\mathbf{x}) - f(\mathbf{x})) \tag{3.60}$$

and $(R_{l+1}(\mathbf{x}) - f(\mathbf{x}))^2$ becomes:

$$\begin{gathered}(R_{l+1}(\mathbf{x}) - f(\mathbf{x}))^2 = (\alpha'_{l+1})^2 (g_{l+1}(\cdot) - f_u(\mathbf{x}))^2 + \\ 2\alpha'_{l+1}(1 - \alpha'_{l+1})(g_{l+1}(\cdot) - f_u(\mathbf{x}))(R_l(\mathbf{x}) - f(\mathbf{x})) + (1 - \alpha'_{l+1})^2 (R_l(\mathbf{x}) - f(\mathbf{x}))^2\end{gathered} \tag{3.61}$$

Using the above identities and the definitions given in (3.51), (3.52) and (3.53), rewriting (3.57) we get:

$$(\alpha'_{l+1})^2 \Phi + 2\alpha'_{l+1}(1 - \alpha'_{l+1})\Psi + (1 - \alpha'_{l+1})^2 \Omega \le \varepsilon\Omega \tag{3.62}$$

Solving for α'_{l+1}, we get

$$\alpha'_{l+1} \le \frac{-(\Psi - \Omega) - \sqrt{(\Psi - \Omega)^2 - (\Phi - 2\Psi + \Omega)(1 - \varepsilon)\Omega}}{(\Phi - 2\Psi + \Omega)} \tag{3.63}$$

The above is true given **Assumption B** is satisfied. This completes the proof of THEOREM 4. ***Q.E.D.***

In the last two theorems, we have established conditions under which the SPORE-1 learning algorithm converges. Next, we quantify the rate of convergence of this algorithm. More precisely, we measure how the approximation error decreases as the number of sample points used during the construction process increases. As discussed, in Section 2.3 of Chapter 2, this type of rate of convergence analysis is an important indication of an algorithms usefulness in high dimensional regression. In the following lemma and theorem, we establish the rate of convergence, to some finite approximation error given by (3.50), of the SPORE-1 Learning Algorithm as independent of the dimension of the function being approximated.

LEMMA D. *Let the optimal $g_l(\cdot)$ function be defined as*

$$^{opt}g_l(u, v) = \sum_{i,\, j \in \{0, 1, 2, \ldots\}}^{i + j \le \kappa} {}^{opt}a_{ij} \cdot u^i v^j \tag{3.64}$$

where the coefficients $\{^{opt}a_{ij}\}$ are chosen using an infinite *number of training sample points as follows:*

$$\{^{opt}a_{ij}\} = \operatorname{argmin}_{\{^{opt}a_{ij}\}}\left[\int_D (^{opt}g_l(\cdot) - f_u(\mathbf{x}))^2 dD\right] \tag{3.65}$$

and $f_u(\mathbf{x})$ is the residual error function defined in LEMMA C. Similarly, let the approximated $g_l(\cdot)$ function be defined as

$$g_l(u, v) = \sum_{i,\, j \in \{0, 1, 2, \ldots\}}^{i + j \le \kappa} a_{ij} \cdot u^i v^j \tag{3.66}$$

where the coefficients $\{a_{ij}\}$ are chosen using a finite *number of training sample points as follows:*

$$\{a_{ij}\} = \operatorname{argmin}_{\{a_{ij}\}}\left[\sum_{\Gamma_T^B}(g_l(u, v) - y_q)^2\right] \tag{3.67}$$

Given **Assumption A**, *and for sufficiently large sample size, the rate of convergence of the function $g_l(\cdot)$ to the optimal function $^{opt}g_l(u, v)$, as a function of the number of sample points M_T, is independent of dimension N of the target function $f(\mathbf{x})$ (we assume $N > 2$). More precisely, there exists a constant K, independent of N, such that the rate of convergence is given by:*

$$E\left[\int_D (g_l(\cdot) - [{}^{opt}g_l(\cdot)])^2 dD\right] \le K\left(\frac{(T_{g_l(\cdot)})^2}{M_T}\right) \tag{3.68}$$

where, $T_{g_l(\cdot)}$ is the number of terms in the polynomial $g_l(\cdot)$.

Proof: We first prove this theorem for the first level $l = 1$. The coefficients of ${}^{opt}g_l(u, v)$ are given, using an infinite sample size, by the following:

$$\{{}^{opt}a_{ij}\} = \operatorname{argmin}_{\{{}^{opt}a_{ij}\}}\left[\int_D ({}^{opt}g_1(x_{k_0}, x_{k_1}) - f(\mathbf{x}))^2 dD\right] \tag{3.69}$$

Similarly, the coefficients for $g_l(\cdot)$ are given using a finite samples size M_T, as follows:

$$\{a_{ij}\} = \operatorname{argmin}_{\{a_{ij}\}}\left[\sum_{\Gamma_T^B} (g_1(x_{k_0}, x_{k_1}) - y_q)^2\right] \tag{3.70}$$

Thus, given that $g_l(\cdot)$ converges to ${}^{opt}g_l(\cdot)$ as $M_T \to \infty$ (this is true because $g_l(\cdot)$ is continuous and so is $f(\mathbf{x})$), then we can view $g_l(\cdot)$ as being an approximation of ${}^{opt}g_l(\cdot)$. Given this view, the coefficients $\{a_{ij}\}$ are chosen based on sample outputs y_q which are generated using the following density function:

$$h_{g_l}(y|(x_{k_0}, x_{k_1})) = \frac{\int_{D'} ({}^{opt}g_1(x_{k_0}, x_{k_1}) - f(\mathbf{x}))dD'}{\int_{D'} dD'} \tag{3.71}$$

where $f(\mathbf{x})$ is defined in (3.25) of **Assumption A**, and D' is the domain D given that (x_{k_0}, x_{k_1}) is fixed (hence D' is $N-2$ dimensional space, spanning the $N-2$ dimensions of D which do not include the axes (x_{k_0}, x_{k_1})). Within this context, and by **Assumption A**, $h_{g_l}(y|(x_{k_0}, x_{k_1}))$ must be continuous with finite variance, and an equivalent representation for ${}^{opt}g_1(x_{k_0}, x_{k_1})$ is:

$${}^{opt}g_1(x_{k_0}, x_{k_1}) = E[y|(x_{k_0}, x_{k_1})] = \int_{-\infty}^{\infty} y h_{g_l}(y|(x_{k_0}, x_{k_1}))\varphi(dy) \tag{3.72}$$

Thus, we view the sample outputs y_q as being generated as follows:

$$y_q = {}^{opt}g_1(x_{k_0}, x_{k_1}) + \varepsilon(x_{k_0}, x_{k_1}) \tag{3.73}$$

where the density function of the *error* term $\varepsilon(x_{k_0}, x_{k_1})$ is continuous, and has finite variance

$\sigma_\varepsilon^2(x_{k_0}, x_{k_1})$ dependent on (x_{k_0}, x_{k_1}). Note that the variance $\sigma_\varepsilon^2(x_{k_0}, x_{k_1})$ is independent of dimension N because it is calculated using the distribution given in (3.71), which integrates and divides out $N-2$ dimensions of $f(\mathbf{x})$. Now, using the equality $\sigma^2(y) = E(y^2) - (E(y))^2$, we can write

$$E[(g_1(\cdot) - {}^{opt}g_1(\cdot))^2] = \sigma^2[g_1(\cdot) - {}^{opt}g_1(\cdot)] + (E[g_1(\cdot) - {}^{opt}g_1(\cdot)])^2 \tag{3.74}$$

From the Central Limit Theorem we know that $E[g_1(\cdot) - {}^{opt}g_1(\cdot)] = 0$ for each point (x_{k_0}, x_{k_1}). In addition, because ${}^{opt}g_1(x_{k_0}, x_{k_1})$ is constant for each fixed point (x_{k_0}, x_{k_1}), we can write (3.74) as follows (note that we are using the equality $\sigma^2(y+c) = \sigma^2(y)$ for c constant):

$$E[(g_1(\cdot) - {}^{opt}g_1(\cdot))^2] = \sigma^2[g_1(\cdot) - {}^{opt}g_1(\cdot)] = \sigma^2(g_1(\cdot)) \tag{3.75}$$

Next, if we symbolize the terms of $g_1(\cdot)$ as

$$g_1(\cdot) = \sum_{k=1}^{T_{g_1(\cdot)}} \theta_k \tag{3.76}$$

then, using the identity $\sigma^2(\sum_{i=1}^n c_i\theta_i) = \sum_{i=1}^n \sum_{j=1}^n c_i c_j \sigma(\theta_i, \theta_j)$, where $\sigma(\theta_i, \theta_j)$ is the covariance between θ_i and θ_j, we obtain

$$E[(g_1(\cdot) - {}^{opt}g_1(\cdot))^2] = \sigma^2\left(\sum_{k=1}^{T_{g_l(\cdot)}} \theta_k\right) = \sum_{i=1}^{T_{g_l(\cdot)}} \sum_{j=1}^{T_{g_l(\cdot)}} \sigma(\theta_i, \theta_j) \tag{3.77}$$

Now, for all terms $i = j = k$ where $k \in \{1, \ldots, T_{g_l(\cdot)}\}$, the covariance is given by the Central Limit Theorem as (i.e. variance of a distribution decreases as $\sigma^2(y)/n$ for n samples, if the original distribution had variance $\sigma^2(y)$):

$$\sigma(\theta_k, \theta_k) = \sigma^2(\theta_k) = \frac{\sigma_\varepsilon^2(x_{k_0}, x_{k_1})}{M_T} \tag{3.78}$$

where $\sigma_\varepsilon^2(x_{k_0}, x_{k_1})$ is the variance of the *noise* term defined in (3.73). Similarly, for all terms $i \neq j$ the covariance is:

$$\sigma(\theta_i, \theta_j) = \frac{\sigma_{ij}^2(x_{k_0}, x_{k_1})}{M_T} \tag{3.79}$$

where $\sigma_{ij}^2(x_{k_0}, x_{k_1})$ is some initial covariance between θ_i and θ_j (note that $\sigma_{ij}^2(x_{k_0}, x_{k_1})$ measures the covariance between terms of the 2 dimensional polynomial $g_1(\cdot)$, and is therefore independent of N). Thus, if we let $\sigma^2_{Max}(x_{k_0}, x_{k_1})$ be the maximum covariance of all of $\sigma^2_{ij}(x_{k_0}, x_{k_1})$ and $\sigma^2_{\varepsilon}(x_{k_0}, x_{k_1})$, we can write (3.77) as:

$$E[(g_1(\cdot) - {}^{opt}g_1(\cdot))^2] = \sum_{i=1}^{T_{g_l(\cdot)}} \sum_{j=1}^{T_{g_l(\cdot)}} \sigma(\theta_i, \theta_j) \le (T_{g_l(\cdot)})^2 \left(\frac{\sigma^2_{Max}(x_{k_0}, x_{k_1})}{M_T} \right) \tag{3.80}$$

Therefore, integrating ever D, we get

$$E\left[\int_D (g_1(\cdot) - [{}^{opt}g_1(\cdot)])^2 dD\right] \le \left(\frac{(T_{g_l(.)})^2}{M_T} \right) \left(\int_D \sigma^2_{Max}(x_{k_0}, x_{k_1}) dD \right) = K \left(\frac{(T_{g_l(.)})^2}{M_T} \right) \tag{3.81}$$

where $K = \int_D \sigma^2_{Max}(x_{k_0}, x_{k_1}) dD$, is independent of N. Thus, this proves LEMMA D for the function $g_1(\cdot)$. The proof for all other levels $l > 1$ follows the same logic and is not given here. This completes the proof of LEMMA D. ***Q.E.D.***

In LEMMA D we proved that the rate of convergence of each function $g_l(\cdot)$ is independent of dimension N of the target function. In the following theorem, we extend this to show that the rate of convergence of $R_l(\mathbf{x})$ is also independent of N.

THEOREM 5. *Let* ${}^{opt}R_{L_1}(\mathbf{x}) = \sum_{l=1}^{L_1} {}^{opt}\alpha_l \cdot {}^{opt}g_l(\cdot)$ *be the optimal constructed cascade (to level* L_1 *) after the first time that* ***STEP 2*** *of the SPORE-1 learning algorithm is executed; thus* ${}^{opt}R_{L_1}(\mathbf{x})$ *is constructed using an infinite number of samples points, and* ${}^{opt}\alpha_l$ *is defined as:*

$$ {}^{opt}\alpha_l = \frac{\left(\int_D ({}^{opt}g_l(\cdot) - f(\mathbf{x}))^2 dD \right)^{-1}}{\sum_{q=1}^{L_1} \left(\int_D ({}^{opt}g_q(\cdot) - f(\mathbf{x}))^2 dD \right)^{-1}}, \tag{3.82}$$

where ${}^{opt}g_l(\cdot)$ *is defined in* LEMMA D. *Similarly, let* $R_{L_1}(\mathbf{x}) = \sum_{l=1}^{L_1} \alpha_l \cdot g_l(\cdot)$ *be the constructed cascade (using the usual finite number of samples points) after the first time that* ***STEP 2*** *of the SPORE-1 learning algorithm is executed. Then, given* ***Assumption A*** *and for sufficiently large sample sizes* M_T, *the rate of convergence, as a function of the number of sample points* M_T, *of the function* $R_{L_1}(\mathbf{x})$ *to the optimal function* ${}^{opt}R_{L_1}(\mathbf{x})$, *is independent of dimension* N *of the target function* $f(\mathbf{x})$, *(we assume* $N > 2$ *). More precisely, there exists a constant* K, *independent of* N, *such that the rate of convergence is given by:*

$$E\left[\int_D (R_{L_1}(\mathbf{x}) - [{}^{opt}R_{L_1}(\mathbf{x})])^2 dD\right] \le K\left(\frac{(T_{g_l(\cdot)})^2}{M_T}\right) \tag{3.83}$$

where, $T_{g_l(\cdot)}$ is the number of terms in the polynomial $g_l(\cdot)$.

Proof: Using the identity $\sigma^2(y) = E(y^2) - (E(y))^2$, we write

$$E[(R_{L_1}(\mathbf{x}) - [{}^{opt}R_{L_1}(\mathbf{x})])^2] = \sigma^2(R_{L_1}(\mathbf{x}) - [{}^{opt}R_{L_1}(\mathbf{x})]) + (E(R_{L_1}(\mathbf{x}) - [{}^{opt}R_{L_1}(\mathbf{x})]))^2 \tag{3.84}$$

From the Central Limit Theorem we know that $E(R_{L_1}(\mathbf{x}) - [{}^{opt}R_{L_1}(\mathbf{x})]) = 0$ for each point $\mathbf{x}$. In addition, because ${}^{opt}R_{L_1}(\mathbf{x})$ is constant for each fixed point $\mathbf{x}$, we can write (3.74) as follows (note that we are using the equality $\sigma^2(y + c) = \sigma^2(y)$ for c constant):

$$E[(R_{L_1}(\mathbf{x}) - [{}^{opt}R_{L_1}(\mathbf{x})])^2] = \sigma^2(R_{L_1}(\mathbf{x}) - [{}^{opt}R_{L_1}(\mathbf{x})]) = \sigma^2(R_{L_1}(\mathbf{x})) \tag{3.85}$$

Next, given that $R_{L_1}(\mathbf{x}) = \sum_{l=1}^{L_1} \alpha_l \cdot g_l(\cdot)$, we can write the above as:

$$E[(R_{L_1}(\mathbf{x}) - [{}^{opt}R_{L_1}(\mathbf{x})])^2] = \sigma^2\left(\sum_{l=1}^{L_1} \alpha_l \cdot g_l(\cdot)\right) = \sum_{i=1}^{L_1}\sum_{j=1}^{L_1} \alpha_i\alpha_j\sigma(g_i, g_j) \tag{3.86}$$

For each point $\mathbf{x}$, let $\rho_{Max}(\mathbf{x})$ be the maximum covariance over all $\sigma(g_i, g_j)$. Then, given that $\sum_{i=1}^{L_1} \alpha_i = 1$ by definition, we can write the above as:

$$E[(R_{L_1}(\mathbf{x}) - [{}^{opt}R_{L_1}(\mathbf{x})])^2] \le \left(\sum_{i=1}^{L_1} \alpha_i\right)\left(\sum_{j=1}^{L_1} \alpha_j\right)\rho_{Max}(\mathbf{x}) = \rho_{Max}(\mathbf{x}) \tag{3.87}$$

From LEMMA D, we know that for $\sigma(g_i, g_i) = \sigma^2(g_i)$, the variance has the form:

$$\sigma^2(g_i) \le (T_{g_l(\cdot)})^2\left(\frac{\sigma^2_{Max}(\cdot)}{M_T}\right) \tag{3.88}$$

Thus, given that the functions $g_l(\cdot)$ are constructed one at a time using a least squared error fit, all other covariances $\sigma(g_i, g_j)$ must have the same form given in (3.88). Therefore, substituting $\rho_{Max}(\mathbf{x})$ with the maximum $\sigma(g_i, g_j)$, (3.87) becomes:

$$E[(R_{L_1}(\mathbf{x}) - [{}^{opt}R_{L_1}(\mathbf{x})])^2] \le \left(\frac{(T_{g_l(\cdot)})^2}{M_T}\right)\sigma^2_{Max}(\cdot) \tag{3.89}$$

Therefore, integrating ever D, we get

$$E\left[\int_D (R_{L_1}(\mathbf{x}) - [{}^{opt}R_{L_1}(\mathbf{x})])^2 dD\right] \le \left(\frac{(T_{g_l(.)})^2}{M_T}\right)\left(\int_D \sigma^2_{Max}(\cdot)dD\right) = K\left(\frac{(T_{g_l(.)})^2}{M_T}\right) \tag{3.90}$$

where $K = \int_D \sigma^2_{Max}(\cdot)dD$, is independent of N (see proof of LEMMA D). This completes the proof of THEOREM 5. ***Q.E.D.***

Therefore, THEOREM 5 shows that the rate of convergence of the error function, to the optimal error function is independent of dimension of the function being approximated. This result is important because it establishes that, for the SPORE-1 Learning Algorithm, the rate at which the constructed cascade approaches the best regression function is independent of the dimension of the function being approximated. Thus the number of sample points required to build an optimal SPORE-1 regression function, does not explode with dimension.

3.3.2 Complexity Results

An important property of a learning algorithm for the construction of high dimensional regression functions, is how its complexity grows with the number of input variables. Given that the number of parameters of each function $g_l(\cdot)$ is fixed, the computational cost of adding a level l is the same for all levels. In the following, we symbolize the computational complexity of adding a new level to the regression function, as C. The cost C is dominated by the computations required to determine the polynomial coefficients of $g_l(\cdot)$ (see **STEP 2** of the SPORE-1 Learning Algorithm). In the following theorem, we calculate the computational cost of the SPORE-1 learning algorithm.

THEOREM 6. *For any fixed sequence $k_0, \ldots, k_L$ of indices defining the order of independent variables (see **STEP 1** of the SPORE-1 Learning Algorithm), and any $\varepsilon > 0$, the computational cost to achieve the following convergence*

$$E\left[\int_D (R_{l+N}(\mathbf{x}) - R_l(\mathbf{x}))^2 dD\right] \le \varepsilon \tag{3.91}$$

is μNC, where N is the dimension of $f(\mathbf{x})$, C is the complexity of adding a level to the cascade, and $\mu > 0$ ($\mu \in \Re$) is independent of dimension N.

Proof: From THEOREM 3, we know that a level l satisfying the above condition must exist. Furthermore, due to THEOREM 3, there must be a sequence of positive real numbers $\varepsilon_0, \ldots, \varepsilon_k$

$$E\left[\int_D (R_{iN+N}(\mathbf{x}) - R_{iN}(\mathbf{x}))^2 dD\right] \le \varepsilon_i \tag{3.92}$$

such that $\varepsilon_i \le \varepsilon_{i+1}$ and $\varepsilon_k \le \varepsilon$. Thus k is the number of times one must cycle through the N independent variables before the desired convergence is achieved. Note that k depends on how the function $f(\mathbf{x})$ projects, at each specific level, onto the 2 dimensional functions $g_l(\cdot)$. Thus k depends on the distributions $h_{g_l}(y|\cdot)$ (see LEMMA D), and not on the dimension of $f(\mathbf{x})$. Therefore we complete the proof of THEOREM 6 by choosing $\mu > k$. ***Q.E.D.***

Thus, THEOREM 6 shows that the computational cost of building a SPORE-1 regression function is linear with respect to the number of predictor variables.

Next consider the effect of the number of sample points M_T used in the construction of the regression function, on computational complexity. The main effect of sample size is on the construction of each level of the regression function. Thus, the cost C of adding a single level is some function of the number of sample points: i.e. $C(M_T)$. The actual relationship will depend on how the coefficients of $g_l(\cdot)$ are calculated. In the SPORE-1 algorithm the preferred method is Singular Value Decomposition (SVD), and therefore $C(M_T)$ is the cost of doing 1 SVD calculation (see [Golub and Van Loan, 1993] for details).

3.3.3 Internal Variance Reduction

As outlined in Section 2.2 of Chapter 2, variance reduction is fundamental to constructing regression functions which have low approximation errors. One of the more effective means of variance reduction is to use many different regression functions to create a single prediction using a weighted average of all their outputs. The resulting regression function is referred to as an ensemble regression function. Thus, given P different N dimensional regression functions symbolized by $\hat{f}_i(\mathbf{x})$, the ensemble regression function $\hat{f}(\mathbf{x})$ is given by:

$$\hat{f}(\mathbf{x}) = \sum_{i=1}^{P} \beta_i \hat{f}_i(\mathbf{x}) \tag{3.93}$$

where the β_i are some positive scaling coefficients. Perrone [Perrone, 1993] has shown that optimal variance reduction is achieved when the β_i are calculated as follows:

$$\beta_i = \frac{\sum_j C_{ij}^{-1}}{\sum_k \sum_j C_{kj}^{-1}} \tag{3.94}$$

where C_{ij} is the correlation between the $\hat{f}_i(\mathbf{x})$ and $\hat{f}_j(\mathbf{x})$ regression functions. Furthermore, if residual errors of any two different regression functions $\hat{f}_i(\mathbf{x})$ and $\hat{f}_j(\mathbf{x})$ (i.e. $i \neq j$) are uncorrelated, (3.94) becomes:

$$\beta_i = \frac{[\sigma^2(\hat{f}_i - \hat{f})]^{-1}}{\sum_k [\sigma^2(\hat{f}_k - \hat{f})]^{-1}} \tag{3.95}$$

This calculation of β_i is equivalent to the calculation of the scaling factors α_l are computed in **STEP 2** of the SPORE-1 learning algorithm. Thus, the SPORE-1 regression function will exhibit optimal variance reduction if all $g_l(\cdot)$ have residual approximation errors which are uncorrelated. Although the learning algorithm does not guarantee that the approximation errors of all of the functions $g_l(\cdot)$ are uncorrelated, this condition is implicitly encouraged through use of different bootstrap samples and predictor variable inputs for the construction of each level. In THEOREM 4, we show that error reduction (or equivalently variance reduction) can occur even if two functions $g_i(\cdot)$ and $g_j(\cdot)$ have *some* correlated residual errors, as long as the conditions in **Assumption B** are satisfied.

Therefore, SPORE-1 exhibits variance reduction which is internal to its structure. It does not rely on a large ensemble of different high dimensional regression functions, using instead a weighted average of 2 dimensional functions within it's structure. We refer to this as the *internal variance reduction* property of the SPORE-1 learning algorithm.

In Chapter 4, we experimentally verify that this internal variance reduction property is effective in significantly reducing approximation errors.

3.3.4 Virtual Space Partitioning

Consider the pathological situation where

$$\{{}^{opt}a_{ij}\} = \operatorname{argmin}_{\{{}^{opt}a_{ij}\}}\left[\int_D ({}^{opt}g_l(\cdot) - f(\mathbf{x}))^2 dD\right] = \{0, \ldots, 0\} \tag{3.96}$$

Given this condition, we cannot expect reduction in approximation error to occur because the optimal solution is $g_l(\cdot) = 0$. In Chapter 5 this condition is avoided by partitioning the space D into a finite number of subdomains where this cannot happen (note that in Chapter 5 it is shown that these subdomains must exist). However, even without space partitioning, this pathological condition is not likely because the actual calculation of the coefficients of $g_l(\cdot)$ is done using a bootstrap sample of the training data. Due to replacement during sampling, on average each bootstrap sample contains roughly 63% of the training data [Efron, 1983]. Therefore, a bootstrap sample is essentially a random partitioning of the input space. We term this phenomenon *virtual space partitioning*.

In Chapter 4, we experimentally verify, that virtual space partitioning, causes the SPORE-1 regression function to converge to zero approximation error, even when condition (3.96) is valid. We verify this by applying SPORE-1 to the 10 bit parity problem. As outlined in Section 4.5 of Chapter 4, a major deficiency of most learning algorithms which build a regression function using lower dimensional functional units, is that they cannot represent parity functions that are of higher dimension than their largest dimensional function unit. In being able to approximate the 10 bit parity problem arbitrarily well using only 2 dimensional functional units, the virtual space partitioning property of the SPORE-1 algorithm serves to overcome the deficiencies present in other dimensionality reducing regression functions.

3.4 Summary

This chapter constitutes our first attempt at a theoretical verification of our *thesis* presented in Section 1.3 of Chapter 1. First, we have shown that the SPORE-1 structure is a universal approximator. Second, we have shown that under appropriate conditions, the rate of convergence of the SPORE-1 learning algorithm to some finite approximation error, is independent of the dimension of the target function. Thirdly, we have given sufficient conditions under which the approximation error converges to zero. And finally, we have shown that the complexity of constructing the SPORE-1 regression function is at most linear with the dimension of the target function.

Chapter 4: Experimental Results

In this chapter, the SPORE-1 learning algorithm is experimentally evaluated. In Section 4.1, the implementation details of SPORE-1 are described. In Section 4.2, the algorithm is evaluated on ten well known regression problems. In Section 4.3, the algorithm is evaluated on very high dimensional regression data. In Section 4.4, SPORE-1 is applied to 3 very high dimensional human-to-robot skill transfer problems, In Section 4.5, SPORE-1 is applied to the 10-bit parity problem. Finally, Section 4.6 concludes the chapter with a summary of experimental results.

4.1 Implementation Details of the SPORE-1 Regression Function

In this section we give the implementation details of the SPORE-1 algorithm described in Section 3.2 of Chapter 3. In all of the experiments described in this chapter, the same version of SPORE-1 is used. Thus, the same class of two dimensional functions $g_l(\cdot)$ are used, with identical settings for the learning parameters $depth_1$, $depth_2$, and ε. Even though the regression data presented in this chapter vary widely in dimension and number of learning samples, no fine tuning of the SPORE-1 algorithm was required. The SPORE-1 algorithm is completely automated, requiring no manual intervention.

The functions $g_l(\cdot)$ are modeled as third order 2 dimensional polynomials as follows:

$$\begin{aligned} g_l(u, v) &= a_{00} + a_{01}v + a_{10}u + a_{11}uv + a_{02}v^2 + \\ &\quad a_{20}u^2 + a_{12}uv^2 + a_{21}u^2v + a_{03}v^3 + a_{30}u^3 \end{aligned} \tag{4.1}$$

As indicated in Chapter 3, this choice for $g_l(\cdot)$ is not unique. In THEOREM 2 of Chapter 3 (Section 3.1.2) we showed that $g_l(\cdot)$ must belong to the class of 2 dimensional polynomials of order greater than one. Our original reason for not choosing second order polynomials was that third order polynomials are the smallest order polynomials which incorporate asymmetric terms (i.e. terms such as uv^2 which we assumed would allow for faster convergence) which second order polynomials do not. However, preliminary experiments (these results are not reported here) with second order polynomials suggest that they produce approximations which are comparable (with respect to mean squared error) to those produced by third order polynomials. In addition, second order polynomials often produced regression functions which were smaller than those produced by third order polynomials (however, this may not prove true if we exclude coefficients which are insignificantly small; we have not attempted such an analysis here). Our original reason for not choosing higher order polynomials is due to our goal of testing the SPORE methodology under the simplest implementation: i.e. very simply polynomial building blocks (only 10 coefficients). As with second order polynomials, preliminary experiments (these are not reported here) indicated that fourth, fifth and sixth order polynomials produce regression functions which are comparable (with respect to mean squared error) with those produced with third order polynomials. The main difference being that higher order polynomials produce regression functions which are larger (once again, this may not prove true if we exclude coefficients which are insignificantly small). Thus, even from a practical perspective, it appears that the choice of $g_l(\cdot)$ may be arbitrary, as long as they belong to the class of 2 dimensional polynomials of order greater than one.

The three learning parameters were assigned the following values: $depth_1 = 25$, $depth_2 = 6$, and $\varepsilon = 0.0001$ (see Section 3.2 of Chapter 3). These values were consistently found to be sufficiently close to the theoretically motivated values given in Section 3.3.1 of Chapter 3; i.e. theory suggests that large positive values should be assigned to $depth_1$ and $depth_2$, while $\varepsilon > 0$ should be made as small as possible (see **STEP 2** and **STEP 5** of the SPORE-1 Learning Algorithm in Section 3.2 of Chapter 3).

4.2 Evaluating SPORE-1 on Standard Regression Problems

In this section we evaluate the efficacy of the SPORE-1 algorithm by applying it to 10 well known regression problems from the literature (See Table 4.1) [Grudic and Lawrence, 1997]. These problems were chosen because they have been extensively studied with many different learning algorithms being applied to them. All of these regression examples are comparatively small, ranging in dimension from 4 input variables to 16, with learning sample sizes ranging from 80 to 15,000. Although these regression problems are *not* the type of large, very high dimensional problems that SPORE-1 was designed for (i.e. more than 100 input variables with about 40,000 learning examples), they are among the largest regression problems available in the literature. In addition, they allow us to directly compare the performance of SPORE-1 to many different learning algorithms.

Our comparisons are based on 3 different publications: Breiman [Breiman, 1996a], Rasmussen [Rasmussen, 1996], and Jordan and Jacobs [Jordan and Jacobs, 1994]. In the following, we give a brief description of the regression data used in each of these publications.

4.2.1 Regression Data Used by Breiman

Breiman studied 5 different regression problems in [Breiman, 1996a]. His goal was to determine how Bagging (see Section 2.2 of Chapter 2) affects the approximation error of CART (see Section 2.1.2 of Chapter 2). Since Bagging is a method of improving predictor accuracy through variance reduction, comparing SPORE-1 to the Bagged CART algorithm allows us to directly study the efficacy of the internal variance reduction property of SPORE-1 (see Section 3.3.3 of Chapter 3). Of the 5 regression data sets studied in [Breiman, 1996a], we used the Boston Housing, Friedman #1, Friedman #2, and Friedman #3 data. A brief description of these data sets is given below.

- Boston Hosing Data: This data has 12 predictor variables, which constitute various socio-economic indicators in the Boston area, and 1 output variable which is the median housing price in the corresponding area. There are a total of 506 examples in this data set, 51 of which are randomly selected to be used as a test set.
- Friedman #1: This is a simulated data set which appeared in [Friedman, 1991]. It has 10 independent predictor variables which are uniformly distributed over $0 \leq x_i \leq 1$, for all $i = 1, \ldots, 10$. The output is defined by the following equation:

$$y = 10\sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5 + \varepsilon \tag{4.2}$$

The noise term ε is Normally distributed with mean zero and a variance of 1. Half of the 10 prediction variables $(x_6, \ldots, x_{10})$ do not affect the output y, so this data is a good test of the ability of the SPORE-1 learning algorithm to ignore variables which do not contribute to reduced approximation error. This data has a total of 200 learning examples and 1000 test examples, all randomly generated using (4.2).

- Friedman #2 and Friedman #3: This is simulated data having 4 inputs and 1 output. The data was used in [Friedman, 1991] and simulates the impedance and phase shift in an alternating current circuit. The Friedman #2 is generated using the following:

$$y_2 = (x_1^2 + (x_2 x_3 - (1/(x_2 x_4)))^2)^{1/2} + \varepsilon_2 \tag{4.3}$$

The Friedman #3 is generated using the following:

$$y_3 = \operatorname{atan}\left(\frac{x_2 x_3 - (1/(x_2 x_4))}{x_1}\right) + \varepsilon_3 \tag{4.4}$$

The variables x_1, x_2, x_3, x_4 are uniformly distributed as follows:

$$\begin{gathered} 0 \le x_1 \le 100 \\ 20 \le \left(\frac{x_2}{2\pi}\right) \le 280 \\ 0 \le x_3 \le 1 \\ 1 \le x_4 \le 11 \end{gathered} \tag{4.5}$$

The noise terms ε_2 and ε_3 are Normally distributed with mean zero and a variance of σ_2^2 and σ_3^2 respectively. The variance is chosen so that the signal to noise ratios are 3 to 1; i.e. the ratios of variances are chosen to be $(\sigma_2^2/\sigma_{y_2}^2) = (\sigma_3^2/\sigma_{y_3}^2) = 1/3$, where $\sigma_{y_2}^2$ and $\sigma_{y_3}^2$ are the variance of the outputs of (4.3) and (4.4) respectively. The learning examples for these data sets had 200 randomly generated samples, while the test data had 1000 randomly generated examples.

In Table 4.1 we compare the performance of SPORE-1 on these data sets to the Bagged CART algorithm. We chose these data sets because the manner in which they were used was fully specified, allowing us to directly compare prediction error results (full specification of the use of the fifth data set, Ozone, was not available in [Breiman, 1996a]).

4.2.2 Regression Data Used by Rasmussen

Rasmussen [Rasmussen, 1996] studied five data sets from real-world regression examples. These included Auto Price data, CPU data, Boston Housing data, MPG data, and Servo Data. We give a brief description of these data sets below.

- Auto Price Data: This data set relates the list price of a 1985 automobile to various attributes such as it's size, weight and engine capacity. The data set has 10 real inputs and 1 output. There are 80 learning examples and 79 test examples.
- CPU Data: This data set relates CPU performance to various attributes such as maximum and minimum memory size. The data set has 6 real inputs and 1 output. There are 104 learning examples and 105 test examples.
- Boston Housing Data: This data set is a variation of the Boston Housing data described in the previous section. The data set has 12 real inputs, 1 binary input, and 1 output. There are 80 learning examples and 79 test examples.
- MPG Data: This data relates automobile fuel consumption to such attributes as car weight, model year, and horsepower. The data set has 3 real inputs, 6 binary inputs, and 1 output. There are 256 learning examples and 250 test examples.
- Servo Data: This data relates rise time of a servomechanism to gain settings and choice of mechanical linkages. The data set has 2 real inputs, 10 binary inputs, and 1 output. There are 80 learning examples and 79 test examples.

Rasmussen applied 5 different learning algorithms to this data: Monte Carlo, Gaussian Evidence, Backpropagation, MARS, and Gaussian Process. In his paper, Rasmussen was testing the efficacy of these algorithms as the number of learning data is increased, hence he tested them on various sizes of learning data. Since we are not concerned with data-limited regression in this thesis, we limited our comparisons to those results obtained on the *largest* learning data sizes. In Table 4.1, SPORE-1 is always compared to the algorithm which achieved the *lowest* errors on the corresponding data (this is symbolized by the brackets around the published error results). This gives a direct comparison of SPORE-1 with the best algorithm applied to this regression data.

4.2.3 Regression Data Used by Jordan and Jacobs

Finally, we compared SPORE-1 to the forward dynamics data studied by Jordan and Jacobs [Jordan and Jacobs, 1994]. This data was generated by a simulation of a 4 joint robot arm moving in 3 dimensional space. The goal is to predict the acceleration of the 4 joints, given as input the position, velocity, and applied torque of each joint (i.e. a total of 12 predictor variables). There are 15,000 training examples and 5,000 validation examples. This data is highly nonlinear and is a good test of a learning algorithm's ability to predict nonlinear phenomena. Jordan and

Jacobs applied 7 learning algorithms to this data: linear, Backpropagation, two versions of the Hierarchical Mixtures of Experts (HME) algorithm, two versions of CART, and the MARS algorithm. In Table 4.1 we compare SPORE-1 to the Backporpagation algorithm, which achieved the lowest approximation error of algorithms studied by Jordan and Jacobs.

4.2.4 Experimental Results

Next, we detail our experimental results on the above regression data. All experimental results reported in Table 4.1 are obtained under the *exact* experimental conditions reported in the literature. This allows a direct comparison of the performance of SPORE-1 to other algorithms. All learning times reported in this section are for SPORE-1 programmed in C and running on a Pentium Pro 150 using LINUX.

The data sets studied by Breiman [Breiman, 1996a] and Rasmussen [Rasmussen, 1996] have relatively few training examples, and therefore we constructed the SPORE-1 approximations using 10 fold cross-validation (10 fold cross-validation is one effective method of getting good approximations using a small number of data points). The 10 fold cross-validation procedure used is the following: the *learning data* set was divided into 10 approximately equally distributed sets, and then 10 regression functions were constructed using, in turn, 9 of these sets as *training sets*, and the remaining set as a *validation set* (see Section 3.2.1 of Chapter 3 for definitions of *learning, training* and *validation* data). For each *test data* point (*test data* was *not* used during learning), the outputs of the 10 regression functions were averaged to produce the final approximation output, for which error results are reported. To test reproducibility, 100 independent approximations (independent with respect to random sequences of input variables and bootstrap samples as defined in Section 3.2 of Chapter 3) were generated using the Learning Data: Table 4.1 reports the best *test set* error, along with the average and standard deviation (s.d.) over the 100 independent runs.

For the data sets studied by Breiman, we followed the same experimental procedure used by Breiman [Breiman, 1996a]. Briefly, this procedure is summarized as follows. For each of the 4 data sets, 100 trials were set up using 100 learning and 100 test sets created randomly according to the guidelines outlined in Section 4.2.1 and in [Breiman, 1996a]. For each of the 100 learning sets a regression function was constructed (using 10 fold cross-validation as described above) and evaluated on the corresponding test set. Experimental results for the 100 experiments are given in

Table 4.1. The previously published error for the Breiman [Breiman, 1996a] data refers to the average bagged error reported. From Table 4.1 it is evident that the internal variance reduction property of SPORE-1 is at least as effective as Bagged regression trees in reducing approximation error (see Section 3.3.3 of Chapter 3). Note that the large standard deviation (s.d.) in the errors is due to the fact that each of the 100 learning and training sets were different. For all 4 data sets, the average SPORE-1 learning time ranged from about 1 to 10 minutes per approximation (this is the time required to generate one 10-fold approximation, or equivalently, ten SPORE-1 regression functions). The average size of a single cascade was about 90 K-bytes (i.e. about 8500 model parameters).

For the data sets studied by Rasmussen [Rasmussen, 1996], we report results for the largest learning sets only (we used 80 learning examples of auto price data, 104 of cpu data, 256 of housing data, 192 of mpg data, and 88 of servo data as outlined in Section 4.2.2). For each of these 5 fixed data sets, the SPORE-1 approximation was constructed using 10 fold cross validation as described above. The previously published error given in Table 4.1 under the Rasmussen [Rasmussen, 1996] data sets is the best reported error (indicated by brackets) of the 5 algorithms evaluated, and was obtained from the graphs presented in the paper. We generated 100 trials (independent approximations) for each data set. Because all learning and test data were the same in each trial, this was done to determine what effect the stochastic aspect of the SPORE-1 algorithm has on its performance. As defined in Section 3.2 of Chapter 3, the ordering of the independent variables is random, and the construction of the 2 dimensional functions $g_l(\cdot)$ is done using a (random) bootstrap sample of the training data. The best and average relative mean squared *test set* error, and its standard deviation, are reported in Table 4.1. From Table 4.1, it is evident that although there is some variation in error performance from run to run, the stochastic effect of the algorithm is mostly negligible. As with the Breiman data, the average SPORE-1 learning time ranged from about 1 to 10 minutes per approximation. The average size of a single cascade was about 90 K-bytes (i.e. about 8500 model parameters).

Finally, for the forward dynamics data studied by Jordon and Jacobs [Jordan and Jacobs, 1994], our evaluation was done using the experimental setup described in their paper for the on-line back-propagation algorithm: learning was done using 15,000 training examples, and learning stopped when the error could no longer be reduced on 5000 validation examples. The SPORE-1 approximation for this data consisted of a single cascade per output: due to the large

Table 4.1 Performance of SPORE-1 of Published Data

Published Source	Data	Pub. Error	*SPORE-1 Error (100/10 Runs)*			
			best	ave.	s.d.	Ave. Size of Cascade
Breiman, 1996a	Housing	11.6	1.95	9.4	8.4	90 K-byte (8500 model parameters)
	Friedman 1	6.1	1.77	2.88	0.43	
	Friedman 2	22,100	16,733	19,603	1,413	
	Friedman 3	0.0242	0.014	0.0189	0.0024	
Rasmussen, 1996	Auto Price	(0.29)	0.21	0.29	0.03	
	CPU	(0.17)	0.086	0.14	0.025	
	House	(0.15)	0.128	0.145	0.007	
	MPG	(0.10)	0.008	0.1	0.004	
	Servo	(0.15)	0.198	0.25	0.02	
Jordan and Jacobs, 1994	For. Dyn.	(0.09)	0.0378	0.041	0.009	1 M-byte (94,000 parameters)

data size, 10 fold cross validation was not necessary. In order to estimate the reproducibility of SPORE-1 on this data, we constructed 10 independent approximations. The best and average relative error on the validation set (in accordance with [Jordan and Jacobs, 1994]), and the standard deviation over these 10 independent experiments (time constraints did not allow us to do 100 experiments), is reported in Table 4.1. The previously published error, shown in Table 4.1, is the best relative error (on the validation data) of the 7 algorithms studied in [Jordan and Jacobs, 1994]. The forward dynamics data consists of 12 inputs and 4 outputs, which requires 4 SPORE-1 cascades (1 for each output). The algorithm required about 10 hours of computation to build all 4 cascades, and the average size of each cascade was about 1 M-byte (i.e. about 94,000 model parameters). From Table 4.1 we conclude that the SPORE-1 errors on all 10 runs are consistently low, thus the stochastic effect of the learning algorithm had negligible effect on approximation error.

From Table 4.1, it is evident that the SPORE-1 algorithm demonstrated as good or better error results on all but 1 (the servo data) of the data sets. However, this result should be interpreted with caution. The specific goal of SPORE-1 is to build a regression function which best represents the learning data in the mean squared error sense. The SPORE-1 learning algorithm continues to add levels until the mean squared error can no longer be reduced: this is beneficial if one wants the "best" approximation, but detrimental if one wants a representation of fixed size. Most algorithms referred to in Table 4.1 are parametric and therefore of fixed size. Two exceptions are MARS and CART. However, even comparing these to SPORE-1 should be done with caution: both MARS and CART can be used to determine the significance of various inputs and how they interact with one another. Currently SPORE-1 has no such data interpretation features.

Nevertheless, we cannot ignore the fact that SPORE-1 consistently produced regression functions which had low prediction errors. Thus, even though SPORE-1 was not designed for small, low dimensional problems, it should be considered a strong candidate for such regression problems if prediction error is important.

4.3 Evaluating SPORE-1 on Synthetic High Dimensional Problems

In this section, our goal is to study the SPORE-1 algorithm when applied to very high dimensional regression data, under various noise conditions [Grudic and Lawrence, 1997]. All learning times reported in this section are for SPORE-1 running on a Pentium Pro 150 using LINUX.

Since no large, high dimensional regression data examples were found in the literature, we applied SPORE-1 to fifteen artificial data sets ranging from 100 to 1,600 input dimensions. For five of these data sets no noise was present, while the remaining ten data sets had noise. For the first five of these "noisy" data sets, noise was present only in the output, while the remaining five data sets had both input and output noise. Input noise was generated by simply allowing only half of all inputs to contribute to the output (thus half of the inputs used in constructing the approximation had no effect on output). This allows us to determine how effectively the SPORE-1 learning algorithm is able to account for variables which have no predictive power.

Output noise was generated using a normal distribution with the standard deviation selected to give a signal to noise ratio of 3 to 1, thus implying that the true underlying function accounts for 90% of the variance in learning and test data. Specifically, we generated the regression data as follows:

$$y = f(x_1, \ldots, x_N) + \varepsilon \tag{4.6}$$

where $f(x_1, \ldots, x_N)$ is the target function, and ε is a Gaussian noise term with mean zero and variance σ_ε^2. Under output noise conditions, we chose σ_ε^2 such that $(\sigma_f^2/\sigma_\varepsilon^2) = (3)^2$, where σ_f^2 is the variance of $f(x_1, \ldots, x_N)$. Thus, because ε and $f(x_1, \ldots, x_N)$ are independent, we know that

$$\frac{\sigma_\varepsilon^2}{\sigma_y^2} = \frac{\sigma_\varepsilon^2}{\sigma_f^2 + \sigma_\varepsilon^2} = \frac{1}{(\sigma_f^2/\sigma_\varepsilon^2) + 1} = \frac{1}{9+1} = 0.1 \tag{4.7}$$

where σ_y^2 is the variance of the output y. Therefore, given that *relative error* is defined as

$$\frac{\sigma^2(\hat{f} - f)}{\sigma_y^2} \tag{4.8}$$

where $\hat{f}$ is the constructed regression function, theoretically the best relative error achievable by any approximation is 0.1.

The target function $f(x_1, \ldots, x_N)$ is generated as follows:

$$f(x_1, \ldots, x_N) = \frac{\cos\left(\frac{1}{N}\sum_{i=1}^{N} \cos(r_i(x_i))\right)}{1 + \cos\left(\frac{1}{N}\sum_{i=1}^{N} \cos(s_i(x_i))\right)} \tag{4.9}$$

where N is the input dimension of the data, while $r_i(x_i)$ and $s_i(x_i)$ are continuous 1 dimensional functions which are nonlinear and all different from one another. Equation (4.9) allows us to generate highly nonlinear data of any dimension, while at the same time controlling the *complexity* of the data via the appropriate selection of functions $r_i(x_i)$ and $s_i(x_i)$. By *data complexity,* we are referring to the number of random sample points required to build sufficient non-parametric models of the data, with respect to mean squared error. The more complex the generating function, the more data points are required for non-parametric modeling.

For the purposes of this thesis, we have chosen functions $r_i(x_i)$ and $s_i(x_i)$ such that 40,000 random sample points are sufficient in order to effectively model the data. Our choice of functions $r_i(x_i)$ and $s_i(x_i)$ was done as follows. The functions $r_i(x_i)$ and $s_i(x_i)$ are modeled using linear splines. The spline knot locations were randomly chosen using a uniform random distribution. The number of knots was the same for each function. In addition, each of the predictor variables x_i were modeled as 2 dimensional parametric functions of the form

$x_i = \psi_i(\omega_1, \omega_2)$, where the functions $\psi_i(\omega_1, \omega_2)$ are different for each predictor variable x_i. The parametric functions $\psi_i(\omega_1, \omega_2)$ are modeled as 2 dimensional linear splines, with knot positions randomly chosen using a uniform random number generator (the number of knots is the same in each of the functions $\psi_i(\omega_1, \omega_2)$, but their locations are all different). Hence, the complexity of the target function $f(x_1, \ldots, x_N)$ in (4.9) is directly controlled by the number of knots in functions $r_i(x_i)$, $s_i(x_i)$ and $\psi_i(\omega_1, \omega_2)$, and increasing the number of knots increases the number of points required to achieve effective approximations of the target function. One should note that, even though the target function is essentially 2 dimensional in the parameter space (ω_1, ω_2), this information is not used by SPORE-1; i.e. regression data is provided in the form of N dimensional predictor variables x_i, and one output variable y. Therefore a learning algorithm sees the resulting regression problem as N dimensional.

In order to pick a target function complexity such that 40,000 sample points are required in order to generate an effective regression model, we followed the following procedure. The number of knots used to represent the functions $r_i(x_i)$, $s_i(x_i)$ and $\psi_i(\omega_1, \omega_2)$, was increased until a nearest neighbor approximation (implemented using a simple K-Nearest Neighbor algorithm with $K = 1$; see Chapter 2 for details), *in the 2 dimensional parameter space* (ω_1, ω_2) exceeded a relative mean squared error of 0.01 (relative mean squared error is defined, in the usual sense, as the mean squared error on the test data, divided by the variance of the test data). In building this K-Nearest Neighbor approximation, 40,000 random training samples were used, and the relative error was calculated using 5000 randomly picked points. A nearest neighbor approximation gives a good indication of maximum approximation error, because the 40,000 learning points densely sample the 2 dimensional parameter space (ω_1, ω_2), where the nearest neighbor calculations are done (a nearest neighbor approximation done in the N dimensional predictor variables x_i, would not be nearly as effective, for reasons outlined in Chapter 2). This procedure gives us a good estimate of the maximum approximation error that the SPORE-1 algorithm should get; the relative mean squared error of a SPORE-1 approximation, or any effective approximation method, should not exceed 0.01 on any of the high dimensional data presented in this section.

The simulation results for the 15 high dimensional data sets are presented in Table 4.2. For each data set there are 40,000 learning examples and 40,000 testing examples. Each learning set is further divided into 30,000 training examples and 10,000 validation examples: these are used to construct one cascade which forms the approximation for that data set. The three rightmost columns of Table 4.2 contain the test set average relative mean squared error and corresponding standard deviations over 10 independent runs. From Table 4.2 it is evident that the relative error is small when no noise is present, and falls well below the expected maximum error value of 0.01. With the addition of noise the relative error approaches the theoretical limit (due to the 3 to 1 signal to noise ratio) of 0.1 (see explanation given above). Learning time for each data set with no noise was approximately 7 hours and produced cascades which were between 2 and 6 M-bytes in size (i.e. between approximately 190,000 and 570,000 model parameters). In contrast, learning time for each data set containing noise was about 1 hour and produced approximations of between 200 and 600 K-bytes in size (i.e. between approximately 19,000 and 57,000 model parameters). The drop in learning time and approximation size results because the SPORE-1 learning algorithm stops adding levels when error on the validation set can no longer be reduced. When the data has output noise, this condition is reached sooner than when there is no noise. The reason for this is that interpolation between sample points is no longer possible when noise is present, once a given level of approximation error (i.e. 0.1) is reached.

Table 4.2 Evaluation of SPORE-1 on High Dimensional Data Sets

Dimension	No Noise Data	Output Noise Data	Input/Output Noise Data
100	0.00139 s.d. 0.00006	0.1049 s.d. 0.0003	0.1048 s.d. 0.0004
200	0.0018 s.d. 0.0002	0.1064 s.d. 0.0003	0.1055 s.d. 0.0002
400	0.0009 s.d. 0.0002	0.1041 s.d. 0.0002	0.1064 s.d. 0.0003
800	0.0044 s.d. 0.0006	0.1111 s.d. 0.0003	0.1038 s.d. 0.0003
1,600	0.0011 s.d. 0.0003	0.1087 s.d. 0.0003	0.1042 s.d. 0.0003

The results in Table 4.2 demonstrate that SPORE-1 can potentially be used on large, high dimensional regression problems. It is interesting to observe that the dimension of the data did not adversely affect either learning time or the size of the approximation. This is made evident in Figure 4.1 where we plot approximation size as a function data dimension. The "error bars" indicate the variation in approximation size for each specific data type. These experimental results support the theory developed in Chapter 3, where we showed that the rate of convergence of SPORE-1 is independent of dimension, and that complexity is at most linear with dimension. A

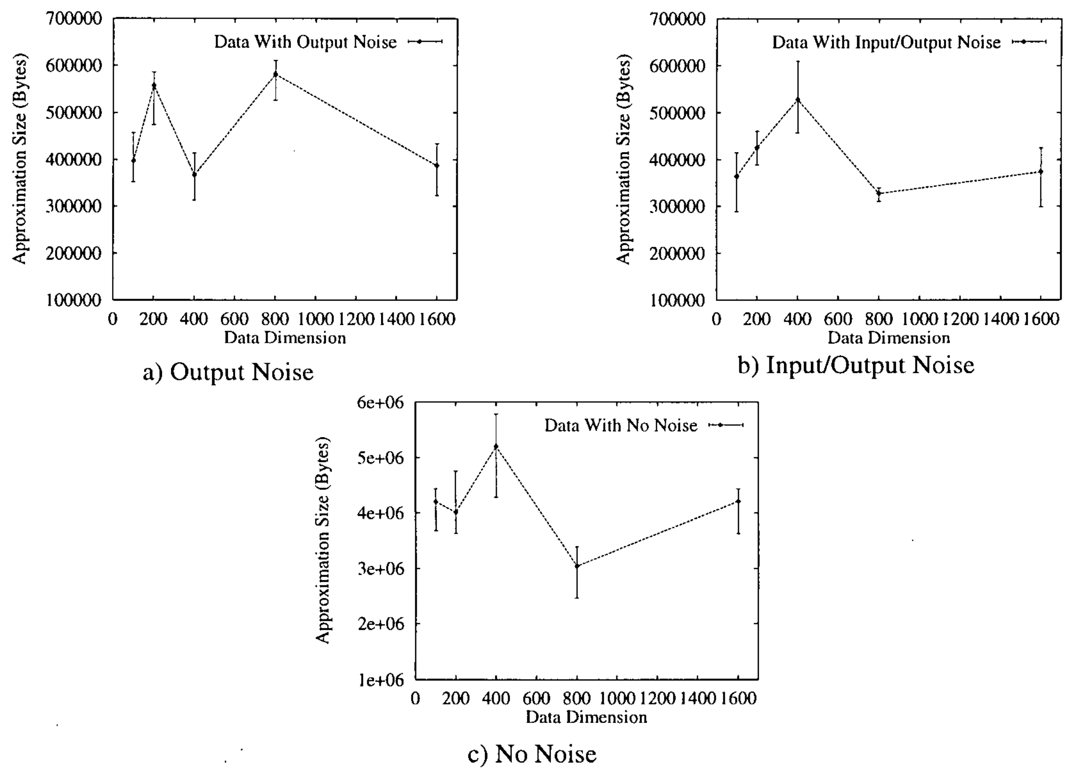


Figure 4.1 Increase in Approximation Size with Data Dimension

second interesting observation is that SPORE-1 is effective in dealing with both input and output noise, even on regression functions which are very high dimensional.

4.4 Human-to-Robot Skill Transfer Experiments

We chose human-to-robot skill transfer as a real world test of the SPORE-1 learning algorithm. Our motivation for this is that human-to-robot skill transfer is an example of a regression problem which is very high dimensional (1024 inputs), and exhibits both input and output noise characteristics [Grudic and Lawrence, 1995] [Grudic and Lawrence, 1996].

We approach the problem of programming robots that can work in unstructured environments from the point of view of human-to-robot skill transfer. In this paradigm, the human transfers his or her skills to a robot by giving the robot a finite number of examples of how the desired manipulation task is done. The key to the success of this type of robot programming is that the robot must be able to generalize the desired task, given only the human experts examples, even when the unstructured environments of interest change after learning has occurred.

Other researchers have had success with similar approaches to the robot programming problem. We mention only a few examples here. Perhaps the most related work is that of Pomerleau [Pomerleau, 1993b] [Pomerleau, 1993a] where he successfully uses a 3 layer perceptron network to control the CMU ALVINN autonomous driving system as it drives along a road. Pomerleau's approach differs from that presented here in that 1) we use a nonparametric approximation approach and 2) the human expert supplying the training data uses only the sensor inputs being fed to the approximation (Pomerleau does not use this restriction). However, as with the approach presented here, Pomerleau uses a large number of sensor inputs (960) to effectively generalize the desired task.

Other related work includes that of Kosuge *et. al.* [Kosuge et al., 1991] where *a priori* knowledge of the task is incorporated with human generated examples to transfer a skill. Other researchers use neural networks to encode the human's skills: for the deburring task Shimokura and Liu [Shimokura and Liu, 1991], and for compliance control Asada [Asada, 1990]. Lui and Asada [Lui and Asada, 1992] use process dynamics for transferring manipulative skills. Yang *et. al.* [Yang et al., 1994] use a multidimensional Hidden Markov Model to represent the transferred skill. Rouse *et. al.* [Rouse et al., 1989] also consider symbol processing models as useful tools for capturing human skills and knowledge.

The approach to human-to-robot skill transfer proposed in this section assumes the following:

> *The human operator can accomplish the desired manipulation task by using only the robot's sensors to obtain information about the world, and using only the robot's actuators to manipulate world objects. Thus, the human has only the robot's perspective on gathering world information and manipulating world objects, and no other. This implies that if the human can accomplish the task, then the robot should also be able to accomplish it.*

This assumption has at least one important implication for the choice of a learning system, or approximation framework, for generating the sensor to actuator mappings. Humans are usually only experts at manipulation tasks which they can anthropomorphize to their natural frame of reference. In other words, we are accustomed to particular human sensory and action modalities, and as the information for the task becomes more sparse, the task becomes more difficult. As a result, when a human expert is demonstrating a task which is sufficiently complex, it is likely that he or she is using a large number of sensor inputs of various types, including both visual and

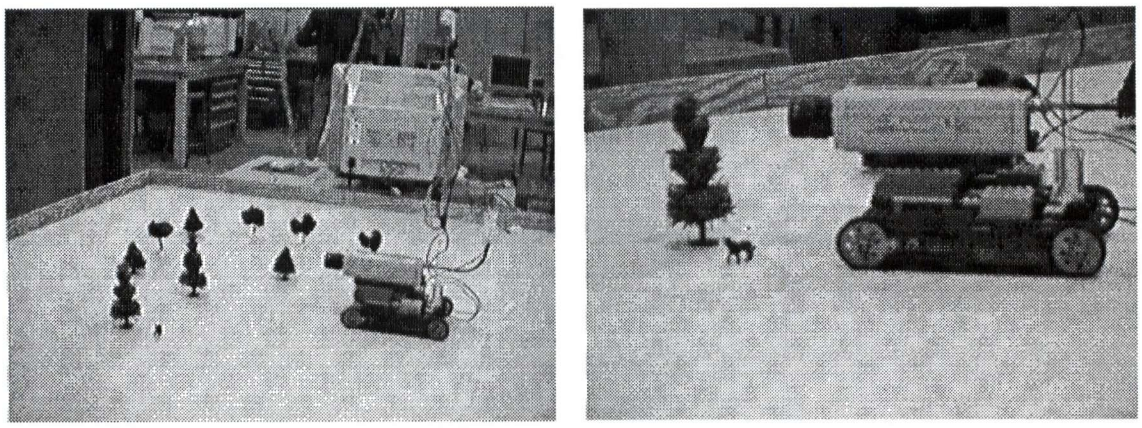


Figure 4.2 The Human to Robot Skill Transfer Test Bed

tactile. Thus, the type of human-to-robot skill transfer proposed here requires an approximation framework which is highly flexible with respect to the number and variety of sensor inputs, and yet is practical to implement.

In this section, we describe a human-to-robot skill transfer testbed which we have implemented, as well as outline our first skill transfer experiments done in an unstructured environment using the SPORE-1 algorithm. The skill which we have chosen to demonstrate is loosely based on a typical "object locate and approach task", and was chosen due to its inherently unstructured characteristics. However, the framework we are in the process of formulating should be generalizable to a large variety of human skills.

4.4.1 The Experimental Testbed

A small scale proof-of-concept testbed for human-to-robot skill transfer has been implemented (see Figure 4.2), consisting of a tracked mobile robot which is approximately 35 centimeters long and 23 centimeters wide. The robot is equipped with 1) left and right tracks, each with independent proportional forward/backward motion similar to that found in forestry excavators; 2) a gripper (not shown in Figure 4.2) to allow it to grasp objects; and 3) an NTSC color camera. The leftmost image in Figure 4.2 shows the robot's workspace and the rightmost image shows a close-up of the mobile robot. Initially this robot is intended to operate in a flat 2.5 meter by 1.8 meter workspace which has randomly placed objects. The intended task of the robot is to manipulate the target objects in a manner defined by a human operator who first accomplishes the desired task a number of times via teleoperation. The human operator observes images received from the camera located on the robot (at a frame rate of approximately 20 to 30 frames per second), and operates two joysticks which control the robot's left/right forward/backward motion, as well as the open/close gripper motion. A SPARCstation 20, equipped with a frame grabber, is used to display the on-board camera image to the human operator as he or she demonstrates the desired task. The learning and subsequent autonomous control of the robot are also performed by the SPARCstation 20.

The testbed currently uses various small objects including lego models and model "trees", as objects which must be located and approached. There are 2 lego objects, a red and blue stripped tower, and a red and blue stripped tower with holes. The "trees" are currently of three different types (tall spruce, short spruce, and short birch), and our goal is to teach the robot to differentiate between them. This testbed is considered unstructured because 1) although there are only three different types of trees and two lego models, no two objects are identical either in height, color distribution, or in texture, making manual encoding of a model difficult (especially for the model trees); and 2) the image background of the objects is a robotics lab which is continually changing as equipment is moved around and people move throughout the lab.

4.4.2 Experimental Results

The first skill which has been successfully transferred from human to robot is described in Figure 4.3. This skill is loosely related to the "tree tending skill," which requires the selection of a certain size and species of tree for inspection. The skill involves turning the robot counterclockwise in place until at least one tall spruce is within the camera's field of view, and then maneuvering the robot into a position where the tree can be inspected. Figure 4.3 shows several typical camera views which the operator sees, as the task of locating and moving towards a tall spruce is performed. Starting at the leftmost image of Figure 4.3, the robot is turning counterclockwise searching for a tree; in the third image a tall spruce has been found and the robot begins to move towards it; in the sixth image the spruce is within inspecting distance and the robot turns off to the left in search of another tall spruce. Note that this first task does not involve avoiding obstacles or actually stopping to inspect the tree, it simply requires the robot to find and move towards a tall

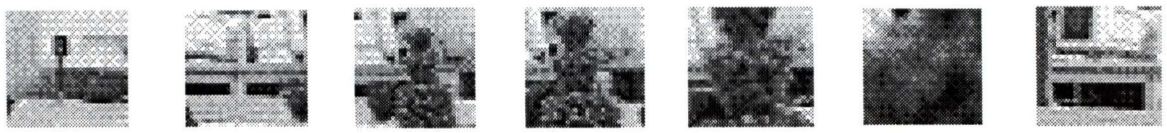


Figure 4.3 A "Tall Spruce Tree"

spruce tree in the visually cluttered environment of the robotics lab.

The image which the human operator, as well as the robot, observes while executing the desired task, consists of 1024 pixels derived by sampling a 32 pixel by 32 pixel region in the center of the original colour NTSC image (see Figure 4.3 for gray-scale examples). The sampled pixel resolution is 8 bits, 3 bits for red, 3 bits for green, and 2 bits for blue. There is no image processing done prior to feeding these pixel values to the SPORE approximation; what the human sees is exactly what is passed to the approximator. In order to generalize the desired task the robot must approximate two mappings, one to control the robot's forward and backward motion, and one to control the robot's left and right motion. Both of these mappings have 1024 inputs and 1 output. As usual, two SPORE-1 cascades are generated, one for each dependent variable.

The training data generated by the human operator consisted of demonstrating the above skill a total of 4 times, from 4 random starting positions of the robot, and 4 random positions of the spruce. Due to image resolution limits, the distance between the robot and the "tree" was always 1.5 meters or less. It took approximately 2 minutes (equivalent to 2180 camera frames or input/output examples) for the human operator to do these 4 demonstrations. The SPORE-1 regression function was constructed by randomly subdividng these 2180 samples into 218 validation samples and 1962 training samples. The resulting approximation took about 5 minutes of SPARCstation 20 CPU time to generate, and took about 250 K-bytes of memory space (i.e. approximately 24,000 model parameters). The length of time to evaluate the resulting approximation for one instance of sensor inputs is 191 milliseconds.

Next we asked the following question: given only a few examples of the desired skill, and the raw sensor inputs used by the human, can the SPORE-1 regression algorithm effectively generalize the desired skill. In the case of the above experiment, the human operator generated only 4 examples of the desired task, and the SPORE approximation was fed raw pixel data with no intensity compensation or any other type of signal preprocessing. In addition, one can see from Figure 4.3 that the image generated by the frame grabber inside the SPARCstation 20 is extremely noisy. Nonetheless, when we ran 30 experiments where the robot used the SPORE approximation

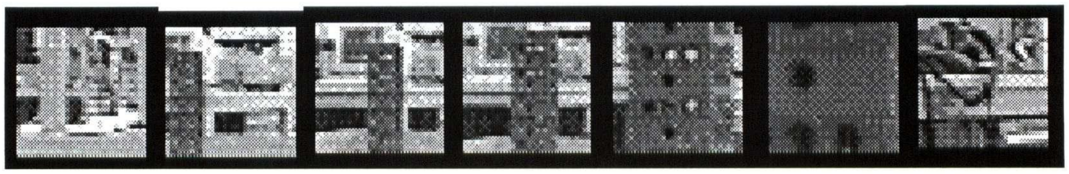


Figure 4.4 A Red and Blue Stripped Tower

to autonomously accomplish the transferred skill, the robot never failed to locate and move towards the tree. These 30 experiments were done with random placements of tree and robot (as with the training set, the spruce was always within 1.5 meters of the robot). It is interesting to note that the robot does not approach, for instance, upside-down spruce trees, and does not seem to be solely color sensitive; the shape of the tree and its texture are significant as well. We may therefore conclude that to the extent that the robot acquired the desired skill, it was able to execute it within an unstructured environment.

Next, we repeated the same experiments with the two lego objects. Figure 4.4 and Figure 4.5 show grey-scale camera frame examples of approach sequences to two objects we have experimented with. The object in Figure 4.4 is a red and blue striped tower, and the object in Figure 4.5 is also a red and blue striped tower but has a number of holes through it, which allow background scenes to be observed. For the object in Figure 4.4, the teleoperator generated 4632 camera images during 4 approaches. Similarly, for the object in Figure 4.5, the teleoperator generated 4805 images during 4 example approaches. The variation in camera frame counts is simply a result of the random placement of the vehicle and object. The SPORE-1 approximation was generated for both objects, using a random selection of 10% of the learning points as validation set, and rest as a training set. The generation of the 4 approach examples took the teleoperator approximately 5 minutes to generate for each object, while the off-line learning of SPORE-1 required approximately 4 additional minutes on the SPARCstation 20, for each object. The size of the approximation generated for the object in Figure 4.4 is 489 K-bytes (i.e. approximately 44,000 model parameters), while the size of the approximation generated for the object in Figure 4.5 is 552 K-bytes (i.e. approximately 52,000 model parameters). The time required to evaluate either approximation (once it has been constructed) for an observed image is less than 120 milliseconds

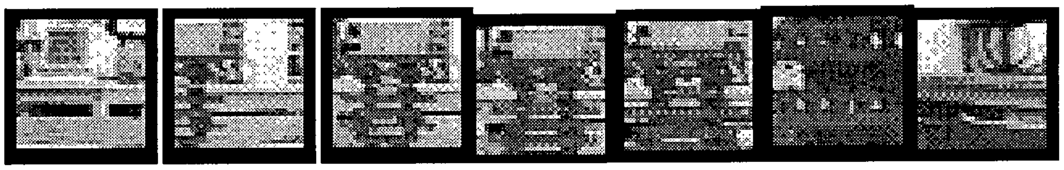


Figure 4.5 A Red and Blue Stripped Tower with Holes

for either object (run on a SPARCstation 20).

Once more, in order to evaluate the ability of the robot to autonomously execute an approach sequence toward an object, we ran 30 experiments for each of the two objects described above. Each experiment involved placing the vehicle and object in random positions, and then seeing whether the robot would autonomously be able to located and approach the object. For the object shown in Figure 4.4 the robot successfully executed the task 29 out of 30 trials, while for the object shown in Figure 4.5, the robot was successful in approaching the object 27 out of 30 trials (only one object was present at each trial). Note that the initial distance between the vehicle and target object was always less than 1.5 meters, because at distances greater than this, the object is not easily distinguishable, even by a human operator.

From these experimental results, we can conclude that the SPORE-1 approximation was successful in generalizing all of the above locate and approach tasks, given relatively few noisy examples and raw pixel inputs. This is promising for the many and difficult robot programming problem facing robotics researchers.

4.5 The 10 Bit Parity Problem

One of the main criticisms of algorithms which construct a high dimensional regression functions using the smaller dimensional functional units, is that they cannot model interactions which have an intrinsic dimension that is greater than the dimension of the highest dimensional functional unit. For example, the classic implementation GMDH [Ivankhnenko, 1971] has 2 dimensional functional units, and therefore can at most model the 2-bit parity problem GMDH [Elder IV and Brown, 1995]. Similarly, ASPN can at most model the 3-bit parity problem because its functional units are at most 3 dimensional [Elder IV and Brown, 1995]. Note that the N-bit parity problem is one of determining whether an odd or even number of the N bits in a word is high: the target function being, for example, set to 1 if an even number of bits are high, and -1 otherwise.

Using this same train of thought, one may conclude that because SPORE-1 uses 2 dimensional functional units, it should not be able to model parity problems for words of 3 or more bits. This, however, is not the case. As discussed in Section 3.3.4 of Chapter 3, the virtual space partitioning property of SPORE-1 can allow the algorithm to converge to zero error, even when the following pathological condition occurs;

$$\{{}^{opt}a_{ij}\} = \operatorname{argmin}_{\{{}^{opt}a_{ij}\}}\left[\int_D ({}^{opt}g_l(u, v) - f(\mathbf{x}))^2 dD\right] = \{0, \ldots, 0\} \tag{4.10}$$

The above condition occurs in a parity problem of more than 2 bits because the target function value is equally to be likely -1 or 1, for each point (u, v) in the training set. Thus, given that the training set consists of an exhaustive set of parity points, and the training data is applied without modification (i.e. a bootstrap sample is *not* used), the function ${}^{opt}g_l(u, v)$ which always gives the best squared error fit to the training data is the zero function $g_l(u, v) = 0$.

In order to demonstrate the ability of SPORE-1 to represent 3 or more bit parity functions, we applied it to the 10-bit parity problem [Grudic and Lawrence, 1997]. The training data and the validation data were made to be identical, with each containing the exhaustive 1024 examples of the 10-bit parity function. Our goal was to determine how well SPORE-1 could fit these learning examples. The results achieved were that the relative mean squared error on the learning data was 9.8×10^{-9}, indicating that, to within the floating point precision of the CPU, the SPORE-1 algorithm was able to *exactly* model the 10-bit parity function. However, the rate at which SPORE-1 converged to this exact solution was rather slow: it took about 2 hours and the final approximation was over a megabyte in size (i.e. approximately 94,000 model parameters). If, on the other hand we stopped the learning algorithm when the parity problem was actually solved (i.e. when the maximum approximation error is less than 1, meaning that every instance of the learning data has been appropriately mapped), the learning time is only 10 minutes, and the size of the approximation is about 40 K-bytes (i.e. approximately 3,800 model parameters).

4.6 Summary

This chapter is our first attempt at an experimental verification of our *thesis* presented in Section 1.3 of Chapter 1. The SPORE-1 algorithm is our most extreme example of the general SPORE methodology. The structural units are small and all the same (i.e third order two dimensional polynomials), and no input selection is ever done. We have demonstrated its successful performance on problems ranging from relatively small 6 dimensional regression problems having only 104 learning examples, to 1,600 dimensional regression problems with 40,000 learning examples. Its performance compared favorably to published results on well known data sets in the literature. We introduced new very high dimensional data and demonstrated that SPORE-1's construction times were independent of dimension. A real world example of human-to-robot skill transfer was also successful despite the raw image pixels used as inputs. Finally, we demonstrated that SPORE-1 does not have the same limitations on parity type problems, as other systems which build regression functions using small dimensional functions which are added one at a time.

Chapter 5: The General SPORE Methodology

In Chapter 3 the SPORE-1 learning algorithm was presented, and its basic theoretical characteristics were explored. In this chapter the most general form of the SPORE methodology is presented and the theoretical characteristics of learning algorithms which construct SPORE-type regression functions are presented. In particular, this chapter deals with the rate of convergence and computational complexity properties of the general SPORE learning algorithms, with particular emphasis on how these depend on the dimension of the function being approximated.

5.1 The Complete SPORE Definition

The most general form of the SPORE structure (see Figure 5.1) has C cascades of D dimensional functions. In Figure 5.1, the D dimensional functions are symbolized by $g_{i,j}$, where i serves to identify one of the C cascades, and j identifies the level of the cascade where the function $g_{i,j}$ is located. In the general case, there are $C \geq 1$ cascades of $D \geq 2$ dimensional functions, and the dimension of the functions $g_{i,j}$ need not all be the same. As a specific example, the SPORE-1 structure in Chapter 3 has $C = 1$ and $D = 2$ for all $g_{i,j}$ functions. The columns in Figure 5.1 symbolize cascades of functions, and the rows symbolize levels in each of the cascades. Thus, the SPORE structure depicted in Figure 5.1 has C cascades, with the first cascade having L_1 levels, and the final cascade having L_C levels.

There are 2 types of inputs to the functions $g_{i,j}$. The first type are symbolized by $\boldsymbol{X}_{i,j} = (x_{k_1}, \ldots, x_{k_{(i,j)}})$, where $(x_{k_1}, \ldots, x_{k_{(i,j)}})$ are some subset (perhaps randomly selected) of all of the model input variables (or independent variables). The second type are symbolized by $\boldsymbol{G}_{i,j} = (g_{r_1,q_1}, \ldots, g_{r_{(i,j)},q_{(i,j)}})$, where $(g_{r_1,q_1}, \ldots, g_{r_{(i,j)},q_{(i,j)}})$ are some subset of outputs of functions which have already been added to the structure (note that g_{r_1,q_1} refers to the output of the function from level q_1 of the r_1 cascade).

The final regression function, symbolized by $\hat{f}$ in Figure 5.1, is some weighted sum of all

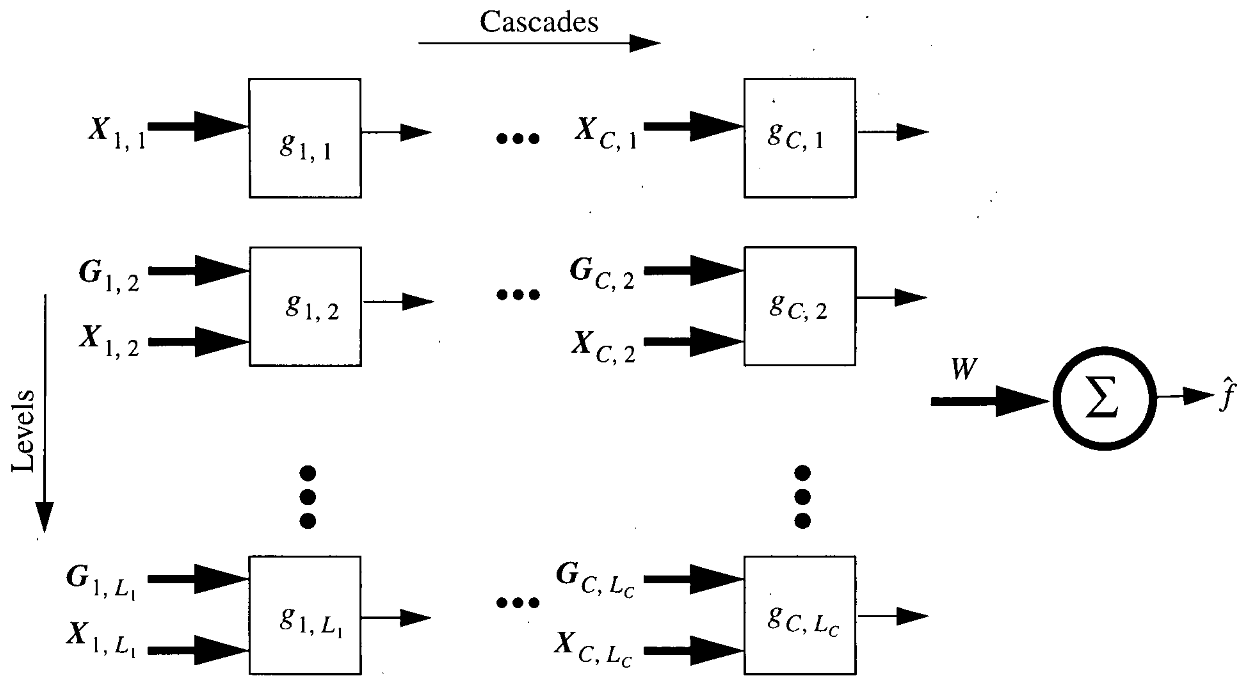


Figure 5.1 The SPORE Structure

constructed functions $g_{i,j}$. This is symbolized by the vector:

$$W = (w_{1,1} \cdot g_{1,1}, w_{1,2} \cdot g_{1,2}, \ldots, w_{C,L_C} \cdot g_{C,L_C}) \tag{5.1}$$

Given this definition of the general SPORE structure outlined in Figure 5.1, we next give a description of how the a general implementation of a SPORE learning algorithm would construct this structure.

The specific structure of the functions $g_{i,j}$ is either 1) fixed throughout SPORE structure as with SPORE-1; 2) chosen randomly, either with or without replacement, from a set of possible candidate functions (if without replacement, then reinitializing the list after all have been chosen), or 3) chosen in a specific order from a predefined sequence. Similarly, the inputs to each $g_{i,j}$ function is either 1) chosen in a specific order (going back to the first one in the list after all have been chosen); or 2) chosen randomly, with or without replacement, from a list of all possible inputs (if without replacement, then reinitializing the list after all have been chosen). This methodology for selecting inputs and function types implies that no time consuming search occurs during the process of constructing the structure.

In all instances of SPORE, the function $g_{1,1}$ is constructed first, its inputs being some $D_{1,1} \geq 2$ dimensional projection of the N dimensional learning data. The parameters of the function $g_{1,1}$ are determined using some subset (perhaps even all) of points in a *training set* containing examples of inputs and outputs. Once the initial function $g_{1,1}$ is constructed, the remaining functions are constructed, one at a time, with the only stipulation being that no function $g_{i,j}$ is constructed until all of its inputs have been constructed. As with the initial function $g_{1,1}$, the parameters of each function $g_{i,j}$ are determined using some subset (or perhaps all) of the points in a *training set*, projected onto the inputs $(\boldsymbol{X}_{i,j}, \boldsymbol{G}_{i,j})$ of $g_{i,j}$.

The weights in the vector W are either all the same, or chosen using some criteria such as w_{ij} being proportional to the inverse of the approximation error of $g_{i,j}$ on the training data (note that this is how the weights are chosen in the SPORE-1 learning algorithm).

During construction the desired learning outputs are periodically replaced by the residual errors resulting from subtracting out the approximation due to the structure already constructed. This usually happens when the approximation errors can no longer be reduced by fitting the parameters of the current $g_{i,j}$ function to the current residual errors in the data set. Further additions to the structure are then added to reduce these residual errors.

In the general case, the SPORE construction terminates when the approximation error, on some *validation set*, has been reduced to some acceptable level. An exception to this rule is when the algorithm allows for the subdivision of the input space of the target function into disjoint subdomains. In which case new SPORE structures are then constructed in each resulting subdomain.

The reader should note that the above description conforms with the general SPORE characteristics outlined in Section 1.3.1 of Chapter 1.

5.2 Theoretical Analysis of Rates of Convergence For Two Specific SPORE Structures

In this chapter we study the rate of convergence properties of two specific examples of SPORE: the SPORE-2 algorithm and an extension of the SPORE-1 algorithm defined in Chapter 3. As discussed in Section 2.3 of Chapter 2, there are two types of rate of convergence results. The first assumes an infinite number of learning examples and measures how approximation error decreases as new structures (or parameters) are added to the regression model. The second assumes a finite number of learning samples and measures how approximation error decreases as the learning sample size increases. In this section we address the second type of convergence result.

In Section 5.2.1, we describe and theoretically analyze the SPORE-2 regression function. Specifically, our goal is to define function spaces for which the rate of convergence of SPORE-2 is independent of dimension. In such function spaces, it is potentially possible to avoid both the computational complexity and large sample size aspects of the curse of dimensionality. Our aim is to analyze theoretically motivated regression functions which point the way to real implementations that are feasible. The theoretical constructions presented should be viewed with this in mind.

In Section 5.2.2, we extend the SPORE-1 regression function described in Chapter 3 to include the partitioning of the input space into disjoint subdomains. This subdivision of the input space guarantees that the approximation error to converges to zero, while simultaneously preserving the rate of convergence and complexity properties of the SPORE-1 learning algorithm within each of these subdomains.

5.2.1 The SPORE-2 Regression Function

As the primary goal of this thesis is to study very high dimensional nonparametric regression, in this section we explore function spaces where such large problems are potentially tractable. In the previous two chapters we have demonstrated the value of building cascades of lower dimensional functions as approximations to high dimensional functions. In this section we extend this basic idea to more complex cascaded structures, and include space partitioning as a method to ensure that the approximation error always converges zero. As is outlined below, an additional benefit of space partitioning is that the rate of convergence of the regression function is independent of dimension, within each resulting subdomain.

The SPORE-2 regression model consists of a partitioner and a structure. The structure consists of a cascade of 2 dimensional functions, and the partitioner is a method of subdividing the domain of the target function into regions were the cascade is an effective regression function. The algorithm for building the SPORE-2 regression function is based first on an algorithm which decomposes a continuous 3 dimensional function $\mathcal{U}(x, y, z)$ into a cascaded 2 dimensional function having the structural form $\mathcal{W}(\mathcal{V}(x, y), z)$. Such a decomposition of 3 dimensional function into two 2 dimensional functions is possible because we partition the domain of $\mathcal{U}(x, y, z)$ into a finite number of regions where it can be represented to arbitrary accuracy by $\mathcal{W}(\mathcal{V}(x, y), z)$. This partitioning is the subject of the following theorem.

THEOREM I: *Let* $\mathcal{U}(x, y, z)$ *be a bounded continuous 3 dimensional function defined*

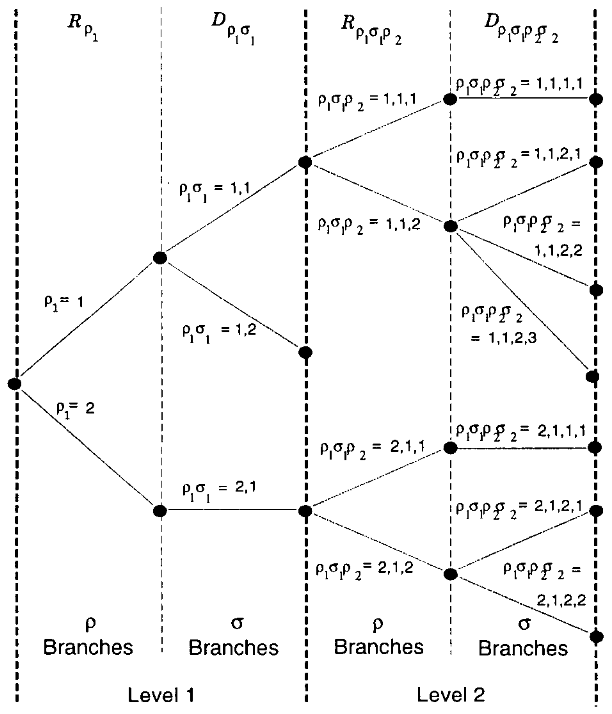

R_{ρ_1}
$D_{\rho_1\sigma_1}$
$R_{\rho_1\sigma_1\rho_2}$
$D_{\rho_1\sigma_1\rho_2\sigma_2}$
$\rho_1 = 1$
$\rho_1 = 2$
$\rho_1\sigma_1 = 1,1$
$\rho_1\sigma_1 = 1,2$
$\rho_1\sigma_1 = 2,1$
$\rho_1\sigma_1\rho_2 = 1,1,1$
$\rho_1\sigma_1\rho_2 = 1,1,2$
$\rho_1\sigma_1\rho_2 = 2,1,1$
$\rho_1\sigma_1\rho_2 = 2,1,2$
$\rho_1\sigma_1\rho_2\sigma_2 = 1,1,1,1$
$\rho_1\sigma_1\rho_2\sigma_2 = 1,1,2,1$
$\rho_1\sigma_1\rho_2\sigma_2 = 1,1,2,2$
$\rho_1\sigma_1\rho_2\sigma_2 = 1,1,2,3$
$\rho_1\sigma_1\rho_2\sigma_2 = 2,1,1,1$
$\rho_1\sigma_1\rho_2\sigma_2 = 2,1,2,1$
$\rho_1\sigma_1\rho_2\sigma_2 = 2,1,2,2$
ρ Branches
σ Branches
ρ Branches
σ Branches
Level 1
Level 2

Figure 5.2 A Tree-Like Subdivision of Input Space

over a finite dense domain in $U \subset \Re^3$. *Then for every arbitrary small real* $\varepsilon > 0$, *there exist a finite number of dense disjoint domains* U_i, *where* $\bigcup_i U_i \equiv U$, *and corresponding bounded continuous 2 dimensional functions* $\mathcal{W}_i$ *and* $\mathcal{V}_i$, *such that for any* $(x, y, z) \in U_i$ *the following is true:*

$$\left|\mathcal{W}_i(\mathcal{V}_i(x, y), z) - \mathcal{U}(x, y, z)\right| < \varepsilon \qquad (5.2)$$

The proof of THEOREM I is given in Appendix A, where it is restated in a more precise, but equivalent, way. The proof is based on four algorithmic constructions which are presented in Appendix D. These constructions are titled Construction I through Construction IV, and they collectively serve to decompose the function $\mathcal{U}(x, y, z)$ into the functions $\mathcal{W}_i(\mathcal{V}_i(x, y), z)$, as well as define and build the disjoint domains U_i.

Using the 3 dimensional decomposition and domain partitioning outlined in THEOREM I, we can begin to define the domain partitioning nature of the complete N dimensional SPORE-2 algorithm. The SPORE-2 partitioner works by subdividing the input space of the target function into a *tree*-like subdivision. This subdivision is represented using the tree notation $\rho_1\sigma_1\ldots\rho_l\sigma_l$, where l represents the level of the tree, and the symbols ρ and σ refer to branch numbers at various levels of the tree. Thus, each branch of the tree (or equivalently path through a tree), defines a subdomain of the target function. An example of such a subdivision of space is given in Figure 5.2. Because each tree branch defines a unique subdomain, we speak of a point $(x_0, \ldots, x_{N-1})$ being an element of the subdomain corresponding to a tree branch with index $\rho_1\sigma_1\ldots\rho_l\sigma_l$.

Given this description of domain subdivision, we can define the structure built by the SPORE-2 regression algorithm within each of these subdomains. The partitioner works by subdividing the domain of the function being estimated into a set of disjoint subdomains, such that in each subdomain, one can estimate the function using a superposition, or summation, of a cascade of 2 dimensional functions. The 2 dimensional functions are of two types symbolized by $g_{\rho_1\sigma_1\ldots\rho_l\sigma_l}$ and $h_{\rho_1\sigma_1\ldots\rho_l\sigma_l}$, where $\rho_1\sigma_1\ldots\rho_l\sigma_l$ is the tree subdivision of the input space defined above. Thus, we can view the functions $g_{\rho_1\sigma_1\ldots\rho_l\sigma_l}$ and $h_{\rho_1\sigma_1\ldots\rho_l\sigma_l}$ as acting on the subdomain defined by tree branch $\rho_1\sigma_1\ldots\rho_l\sigma_l$.

The SPORE-2 structure is defined as follows. Let $\hat{\theta}_n(x_0, \ldots, x_{N-1})$ be the SPORE-2 estimate of some unknown function $\theta(x_0, \ldots, x_{N-1})$, which is based on n random samples of $\theta(x_0, \ldots, x_{N-1})$. Then, if $(x_0, \ldots, x_{N-1})$ is an element of the subdomain corresponding to a tree branch with index $\rho_1\sigma_1\ldots\rho_{l_{max}}\sigma_{l_{max}}$, the SPORE-2 regression function $\hat{\theta}_n(x_0, \ldots, x_{N-1})$ has the following expansion:

$$\hat{\theta}_n(x_0, \ldots, x_{N-1}) = g_{\rho_1\sigma_1}(h_{\rho_1\sigma_1}((x_0, x_1), x_2)) + g_{\rho_1\sigma_1\rho_2\sigma_2}(h_{\rho_1\sigma_1\rho_2\sigma_2}((h_{\rho_1\sigma_1}, x_2), x_3)) + \sum_{l=3}^{l_{max}} g_{\rho_1\sigma_1\ldots\sigma_l}(h_{\rho_1\sigma_1\ldots\sigma_l}((h_{\rho_1\sigma_1\ldots\sigma_{l-1}}, x_{(l \bmod N)}), x_{((l+1) \bmod N)})) \tag{5.3}$$

where,

1. $A \bmod N$ is the integer remainder of $A \div N$, and is used to cycle through the indices of the independent variables $(x_0, \ldots, x_{N-1})$ (hence the order of input variables is predefined);

2. the subscript notation $\rho_1\sigma_1\ldots\rho_{l_{max}}\sigma_{l_{max}}$ is described above (see Appendix B for a more complete specification) and serves to identify the two dimensional functions $g_{\rho_1\sigma_1\ldots\sigma_l}$ and $h_{\rho_1\sigma_1\ldots\sigma_l}$ which act on the specific subdomain of $\theta(x_0, \ldots, x_{N-1})$ referred to by branch $\rho_1\sigma_1\ldots\rho_l\sigma_l$;

3. and, l_{max} is an integer defining the maximum number of terms (or levels of the tree structure) required in order to achieve some approximation error bound

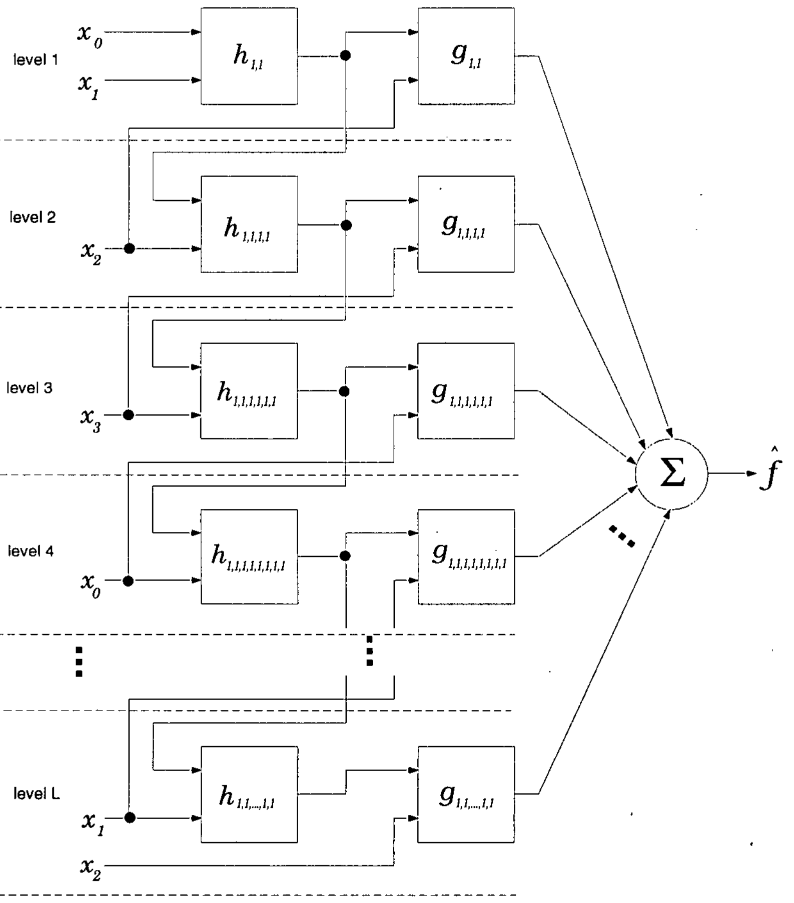


Figure 5.3 A 4 Dimensional Example of SPORE-2

$$\max_{x \in D}\left|\hat{\theta}_n(x) - \theta(x)\right| < \varepsilon \tag{5.4}$$

where ε is an arbitrarily small (for sufficiently large n) positive real number, and D is the domain of the unknown function $\theta(x_0, \dots, x_{N-1})$.

A 4 dimensional example of a SPORE-2 structure is shown in Figure 5.3, where the tree branch is defined by $\rho_l = 1$ and $\sigma_l = 1$, for all $l = 1, \dots, l_{max}$. Note that this branch is identified by the top most branch in Figure 5.2.

Given the above description of SPORE-2, we now state the second theoretical result of this section. In THEOREM II, we prove that the SPORE-2 algorithm produces an approximation which is a universal approximator.

THEOREM II: *Let $\theta(x_0, \dots, x_{N-1})$ be a bounded continuous N dimensional function defined over a finite dense domain in $D \subset \Re^N$. Then, for every arbitrarily small real $\varepsilon > 0$, there* *exist a finite number of dense disjoint domains* S_k, *where* $\bigcup_k S_k \equiv D$, *and each domain* S_i *corresponds to some branch* $\rho_1\sigma_1\ldots\rho_{l_{max}}\sigma_{l_{max}}$, *and corresponding bounded continuous 2 dimensional* functions $g_{\rho_1\sigma_1\ldots\sigma_l}$ and $h_{\rho_1\sigma_1\ldots\sigma_l}$, *such that, for any* $(x_0, \ldots, x_{N-1}) \in S_k$ *the following is true:*

$$\left|\hat{\theta}_n(\boldsymbol{x}) - \theta(\boldsymbol{x})\right| < \varepsilon \tag{5.5}$$

where $\hat{\theta}_n(\boldsymbol{x})$ *is defined in (5.3), and we let* $n \to \infty$.

The proof of THEOREM II is given in Appendix B, where it is restated in a more precise, but equivalent, way. The proof uses the result of THEOREM I, and is based on five algorithmic constructions which are presented in Appendix D. These constructions are titled Construction I through Construction V, and they collectively serve to decompose the function $\theta(\boldsymbol{x})$ into the functions $\hat{\theta}_n(\boldsymbol{x})$, as well as define and build the disjoint domains S_k. Note that Construction I through Construction IV are constructions which are also used in the proof of THEOREM I.

The proof of THEOREM II defines a sufficient algorithm for constructing the universal approximation $\hat{\theta}_n(\boldsymbol{x})$, given that the number of samples points, n, is infinite. In THEOREM III we extend this result by assuming a finite learning sample size n, analyzing how the approximation error decrease as the sample size increases. We state this theorem next.

THEOREM III: *Let* $\hat{\theta}_n(\boldsymbol{x})$, $\theta(\boldsymbol{x})$, *and* S_k *be as defined in THEOREM II, with the further assumptions that* $\theta(\boldsymbol{x})$ *is* p*-times differentiable (* $p \geq 1$ *), and is the mean value of some probability function as follows:*

$$\theta(\boldsymbol{x}) = \int y h(y|\boldsymbol{x})\phi(dy) \tag{5.6}$$

where $h(y|\boldsymbol{x})$ *is some probability function, and* $\phi(dy)$ *is some measure on* $\Re$. *Assume the regularity conditions 1-3 (establishing optimal rates of convergence), defined by Stone [Stone, 1982], are true for the target function* $\theta(\boldsymbol{x})$. *Let* n *be the number of samples of the target function* $\theta(\boldsymbol{x})$ *within a specific subdomain* S_k. *Then, as* n *increases, the rate of convergence of the regression function* $\hat{\theta}_n(\boldsymbol{x})$ *to the target function* $\theta(\boldsymbol{x})$, *on the subdomain* S_k, *is given by:*

$$\left\|\hat{\theta}_n(\boldsymbol{x}) - \theta(\boldsymbol{x})\right\|_\infty \leq O((n)^{-1}\log(n))^r \tag{5.7}$$

where $r = p/(2p+3)$, *and* $\|\cdot\|_\infty$ *is the* L^∞ *norm.*

The proof of THEOREM III is given in Appendix C, where it is restated in a more precise, but equivalent, way. The three regularity conditions of Stone [Stone, 1982] which we use in the statement of the theorem, simply impose certain restrictions on the probability distributions associated with the target function $\theta(\boldsymbol{x})$ (see [Stone, 1982] for details). The first condition imposes constraints on the density function $h(y|\boldsymbol{x})$, and is required to verify that the sequence $((n)^{-1}\log(n))^r$ is a lower convergence sequence (i.e. the *best* convergence sequence). The second condition imposes further constraints on $h(y|\boldsymbol{x})$ to ensure that the rate defined by the sequence $((n)^{-1}\log(n))^r$ is achievable. Examples of density functions which satisfy these two conditions are the Normal distribution and the Bernoulli distribution. The final condition used by Stone puts constraints on the asymptotic distribution of the independent variables, $\boldsymbol{x}$, of the training samples. This condition is required to ensure that the rate of convergence is both optimal an achievable. An example domain $D \subset \Re^N$ which satisfies this condition would be any polyhedral domain in $\Re^N$ where the distribution of points $\boldsymbol{x}$ is uniform, where the learning samples are independent and identically distributed. A more general example of a sufficient domain $D \subset \Re^N$ can be found in [Stone, 1982].

THEOREM III establishes that the closed domain of any function, which is bounded and at least once differentiable (and under appropriate regularity conditions of Stone [Stone, 1982]), is densely covered by a finite number of subdomains where the rate of convergence of our regression model is independent of dimension. This is an important result because it is not intuitively obvious. Previously, Stone [Stone, 1982] showed that the optimal rate of convergence, *over the entire domain* of such functions, does depend on dimension. Specifically, for an N dimensional function, the rate of convergence established by Stone is $\|\hat{\theta}_n(\boldsymbol{x}) - \theta(\boldsymbol{x})\|_\infty \le O((n)^{-1}\log(n))^v$, where $v = p/(2p+N)$. Knowing that every such function has a finite number of subdomains where there exists at least one type of regression function (for example the SPORE-2 regression function studied in this section) which has a rate of convergence independent of dimension, has important practical implications if these domains can be identified. For example, suppose that we have a regression problem which has a rate of convergence that depends adversely on dimension over its whole domain. Furthermore, suppose that we can divide the domain of this regression problem into two regions, with each region having a rate of convergence independent of dimension. Then our original "hard" regression problem has been converted into two "easier" regression problems.

The constructions given in Appendix D of this thesis provide a sufficient algorithm for finding domains which have rates of convergence independent of dimension. These theoretical constructions can serve as a basis for practical algorithms for very high dimensional regression, and in fact are the motivation behind the implementation of the SPORE-1 algorithm.

A further implication of THEOREM III of Appendix C is that it demonstrates that it is possible to measure the complexity of regression in terms of the minimum number of subdomains which have rate of convergence independent of dimension, rather than the dimension of the regression problem. The more subdomains, the more complex the regression problem. This measure of complexity may have important implications for the practical implementation of very high dimensional regression. This is especially true if we can establish the complexity of a regression problem using only the given training examples. Our hope is that the constructions contained in Appendix D of this thesis can serve as the basis for such a data based complexity measure. As described above, these constructions collectively serve to decompose the function $\theta(\boldsymbol{x})$ into the functions $\hat{\theta}_n(\boldsymbol{x})$, while at the same time building the disjoint domains S_k which have the desired rate of convergence properties. Since these constructions are based on learning examples of the target function $\theta(\boldsymbol{x})$ (see proof of THEOREM III in Appendix C), they constitute a sufficient algorithm for such a data based complexity measure. However, the practicality of such a complexity measure has yet to be demonstrated.

5.2.2 The SPORE-1 Regression Function with Domain Partitioning

The rate of convergence result for the SPORE-2 algorithm is based on two key observations. First, the SPORE-2 algorithm works by constructing the approximation one functional unit at a time, and the rate of convergence of each of these individual functional units is independent of dimension. Secondly, the domain of the function being approximated is subdivided to ensure that as each functional unit is constructed, the approximation error over the particular subdivision acted on by the functional unit, must decrease. The result is that within each subdomain, the approximation error converges to zero, and the rate at which it does so is independent of dimension.

Given the definition of the SPORE-1 algorithm in Chapter 3, it is not difficult to envision a modification to the SPORE-1 algorithm which will also give it a rate of convergence to zero error which is independent of dimension. First, in THEOREM 5 of Chapter 3, we showed that the rate of convergence, to a fixed approximation error, of SPORE-1 is independent of the dimension of the function being approximated. Therefore, if we designate each single execution of the SPORE-1 algorithm to be a single equivalent functional unit, then we can achieve a *rate of convergence to zero error*, by subdividing the input space such that the next time SPORE-1 is run on any subdivision, the resulting approximation error over that subdivision will decrease. One method of choosing such a subdivision is to do the following binary split of the input space. Let $R_L(\mathbf{x})$ be the constructed SPORE-1 regression function which acts on the domain D. Then, create 2 subdomains, D_1 and D_2, such that $D_1 \cup D_2 \equiv D$, as follows:

$$\begin{aligned} \mathbf{x} \in D_1 &\text{ iff } ((R_L(\mathbf{x}) - f(\mathbf{x})) > 0) \\ \mathbf{x} \in D_2 &\text{ iff } ((R_L(\mathbf{x}) - f(\mathbf{x})) \le 0) \end{aligned} \tag{5.8}$$

(Note that D_1 and D_2 may not be topologically connected, whereas the subdivisions of most implementations of regression trees [Breiman et al., 1984] are. This may or may not be a significant advantage depending on how hard it is to construct such domains in practice. This is a topic for future research) The subdivision in (5.8) ensures that the maximum expected approximation error, within each subdomain D_i ($i \in \{1, 2\}$), is given by

$$E\left[\int_{D_i} ((R_L(\mathbf{x}) - f(\mathbf{x})) - A_i)^2 dD_i\right] < E\left[\int_{D_i} (R_L(\mathbf{x}) - f(\mathbf{x}))^2 dD_i\right] \tag{5.9}$$

where $A_i \in \Re$ is a constant defined as follows:

$$A_i = \operatorname{argmin}_{A_i \in \Re}\left[\int_{D_i} ((R_L(\mathbf{x}) - f(\mathbf{x})) - A_i)^2 dD_i\right] \tag{5.10}$$

(Note that A_i is simply the average value of $(R_L(\mathbf{x}) - f(\mathbf{x}))$ over each region D_i). Hence, such a subdivision ensures the reduction of error after each execution of the SPORE-1 algorithm. Using THEOREM 5 of Chapter 3, this gives us the following rate of convergence (assuming we subdivide after each execution of SPORE-1),

$$E\left[\int_{D_{\eta(m)}} (R_{L_1}(\cdot) - f(\mathbf{x}))^2 dD_{\eta(m)}\right] \le K\left(\frac{(T_{g_i(\cdot)})^2}{M_{T_{D_{\eta(m)}}}}\right) + \varepsilon \tag{5.11}$$

where

1. ε is any arbitrarily small positive real number;
2. $D_{\eta(m)}$ is any resulting subdomain after m domain subdivisions;

3. and $M_{T_{D_{\eta(m)}}}$ is the number of training sample points in domain $D_{\eta(m)}$.

Thus, the rate of convergence to any arbitrarily small error ε is independent of the dimension of $f(\mathbf{x})$.

5.3 Rate of Convergence and Computational Complexity

In this chapter we have given two specific examples of the SPORE structure. We have shown that both examples have a rate of convergence which is independent of the dimension of the function being approximated as a result of two important characteristics. The first is that the regression function is constructed one functional unit at a time, and the rate of convergence of each of these individual functional units is independent of dimension. The second is that the domain of the function being approximated is subdivided to ensure that as each functional unit is constructed, the overall approximation error decreases. Thus, as long as these two characteristics are maintained, any resulting SPORE implementation will have a rate of convergence independent of dimension.

In THEOREM 6 of Chapter 3, we showed that the computational complexity of the SPORE-1 algorithm is linear with respect to the dimension of the function being approximated. This is a powerful practical result because it makes regression functions which have thousands of inputs, at least potentially feasible from a computational standpoint. This is experimentally supported by the high dimensional regression examples given in Chapter 4.

The SPORE-1 algorithm achieves a computational complexity which is linear with dimension because the drop in approximation error, due to the addition of a new level, does not depend on the dimension of the function being approximated. Instead, it depends on the probability distribution of the function space of $f(\mathbf{x})$ around the 2 dimensional space that the error function is being projected on; i.e. the 2 dimensional space defined by the independent variables of the function $g_l(\cdot)$ (see proof of THEOREM 6 in Chapter 3). Thus, if the variance of this probability distribution is high, this is an indication that $f(\mathbf{x})$ maps poorly onto $g_l(\cdot)$ and many levels in the SPORE-1 structure may be required to get a good fit. Thus, the error drop achieved by cycling through all independent variables of $f(\mathbf{x})$ is not a dependent on the dimension of $f(\mathbf{x})$, but rather on the way it projects onto the 2 dimensional input space of $g_l(\cdot)$. Hence the linear growth in computational complexity.

Following this argument, we can show that, within each subdomain defined in Section

5.2.1, the computational complexity of the SPORE-2 algorithm is also linear with dimension. Thus, the total computational cost of the SPORE-2 algorithm is dependent on the number of subdomains required (which may in fact depend on dimension). Similar arguments are valid for the computational cost analysis of any SPORE structure. Thus, we conclude that within each subdivision of a general SPORE-type structure, the computational cost of the algorithm is linear with respect to the number of input variables.

Finally, we ask the following question: under what conditions is the subdivisions of space not necessary in order to achieve convergence to zero error? For the SPORE-1 algorithm, this question is answered in THEOREM 4 of Chapter 3, where we showed that as long as the approximation errors due to the addition of a level to the regression function are sufficiently uncorrelated with respect to current approximation errors, approximation errors will continue to converge to zero. Another interpretation of this is simply that the new level is contributing to overall error reduction. An analysis of the SPORE-2 constructions gives an equally trivial interpretation of what constitutes continued error reduction without subdivision. In order for SPORE-2 to reduce error *without* space subdivision, the projection of the current error function onto its two dimensional functions must always contribute to overall error reduction. The only condition under which error reduction *will not* occur, is if the projection of the current error function onto the appropriate 2 dimensional space produces functions $g_{\rho_1\sigma_1\ldots\sigma_l}$ and $h_{\rho_1\sigma_1\ldots\sigma_l}$ which are identically zero (see Appendix B for details). Similarly, the SPORE-1 learning algorithm will also fail to converge if $g_l(\cdot) = 0$. Therefore, for both algorithms, subdivision is required when the residual error produce two dimensional functions which are identically zero. As described in Section 3.3.4 of Chapter 3, SPORE-1 uses bootstrap samples to make this pathological condition unlikely. Perhaps it is also possible to use bootstrap samples to achieve similar results with the SPORE-2 algorithm, and more generally with other SPORE-type algorithms. This is an open theoretical question.

5.4 Summary

In this chapter we have shown that the general SPORE structure addresses the two major difficulties associated with high dimensional nonparametric regression. Both of these can be attributed to the curse of dimensionality [Bellman, 1961]). The first difficulty is associated with the potentially intractable number of sample points required to approximate high dimensional functions. The second is associated with the computational complexity inherent in building very high dimensional regression functions. Both difficulties are addressed in the SPORE structure via the sequential addition of small functional units in a way which decreases the residual approximation error. We have shown under which regularity conditions (Stone [Stone, 1982]) this leads to a rate of convergence (as a function of the number of training sample points) independent of dimension. In addition, we have demonstrated that under appropriate conditions, the computational complexity of the general SPORE structure, is linear with respect to the dimension of the function being approximated.

Thus, this chapter, building upon our results in Chapter 3, constitutes our second argument for a theoretical verification of our *thesis* presented in Section 1.3 of Chapter 1. The theoretical analysis presented in this chapter is an attempt to fully understand the fundamental characteristics of a learning algorithm. Many practical implementations of learning algorithms are not based on a firm theoretical understanding of their theoretical characteristics, often resulting in *ad hoc* heuristics which have no firm foundations. Perhaps one of the most significant differences between the SPORE methodology and the other learning methodologies which it loosely resembles (see Section 2.4 of Chapter 2), results from the fact that we have specifically used a theoretical analysis of the characteristics of very high dimensional regression to define our methodology.

Chapter 6: Conclusion

The focus of this thesis has been the construction of models (nonparametric regression functions) which have a large number of inputs, and where little *a priori* knowledge is available about what constitutes a good choice for model structure. Examples of large high dimensional nonparametric regression problems can be found in such diverse fields as meteorology, economics, and robotics. However, general practical methods for constructing these high dimensional models have yet to be widely proposed in the literature.

It has been shown in this thesis that very high dimensional nonparametric regression can be effectively accomplished using a sufficient number of small, low dimensional parametric building blocks, which are fitted to the regression data one at a time. Furthermore, we have demonstrated that both variables and building blocks can be added in a random order to the approximation, thus little or no computational effort is required to determine what to fit next. Projection onto low dimensional parametric building blocks was shown by proof and example to produce stable regression functions from relatively few sample points. It was demonstrated that the random selection of building blocks and their inputs leads to algorithms that are computationally feasible in very high dimensional spaces.

This document is a theoretical and experimental evaluation of a general methodology (SPORE) which exhibits the above characteristics. The most significant result of this thesis is that this simple algorithmic structure leads to regression functions which are effective approximators in very high dimensional spaces. The successful application of nonparametric regression to high dimensional problems such as the 1,600 dimensional problem defined in Section 4.3 of Chapter 4, has not been demonstrated in the past, without using some form of dimensionality reduction requiring *a priori* knowledge. The main reason for this is that previously published techniques have been designed with lower dimensional problems in mind, resulting in algorithms which utilize search strategies in input space, thus making them computationally infeasible in very high dimensional spaces.

The main contributions of this thesis are the following:

I. A new methodology is proposed (SPORE) for constructing very high dimensional regression functions. The SPORE approach to nonparametric regression differs from conventional approaches. Emphasis is shifted from searching through a large parameter space for an optimal structure, to an algorithmic approach which projects learning data onto low dimensional structures, without resorting to computationally expensive searches. This makes SPORE particularly suited to the large regression problems considered in this thesis.

II. A theoretical analysis of SPORE demonstrated that it is a universal approximator. Furthermore, we showed that domain of the target function can be subdivided into a finite number of regions where the rate of convergence (as a function of the number of learning examples) to any bounded continuous target function (at least once differentiable) is independent of the number of input variables (i.e. dimension). This theoretical result is significant because it shows a rate of convergence which has not previously been demonstrated for high dimensional nonparametric regression problems. In addition, we showed that the SPORE structure has a computational complexity which is linear with dimension in each of these subdomains. This is also a significant theoretical result because previously proposed multidimensional nonparametric regression algorithms are not computationally practical in higher dimensional spaces.

III. The simplest example of SPORE, called SPORE-1, has been implemented and evaluated. The SPORE-1 learning algorithm is completely automated, requiring no user intervention. Theoretical analysis of the SPORE-1 learning algorithm gives conditions under which the approximation error converges to zero (without domain partitioning), with a rate of convergence which is independent of the number of input variables (i.e. dimension). Experimental evaluation of SPORE-1 on regression problems which have been previously studied in the literature, shows that SPORE-1 compares favorably, and often outperforms, other published regression algorithms. Out of ten regression examples, SPORE-1 did as well or better than published results on all but one of these examples. Evaluation of SPORE-1 on the 10-bit parity problem showed that it can be applied to problems which have flat low dimensional projections, and therefore is not subject to the same limitations as other methods which construct approximations using low dimensional structural units added sequentially. Evaluation of SPORE-1 on very high dimensional data (100 to 1600 inputs) showed that SPORE-1 performs well under noise conditions in both input variables and output values, and that the dimension of the data had no direct effect on learning times or the size of the approximation. Application of SPORE-1 to real-world human-to-robot skill transfer problems, using raw image pixels, showed that SPORE-1 is effective on very high dimensional (1024 inputs), noisy, real world problems. These human-to-robot skill transfer experiments demonstrated a promising (and theoretically understood) solution for a wide variety of complicated learning/programming tasks, which could not previously be addressed in this manner.

6.1 Future Work

SPORE defines a general methodology for constructing regression functions. The SPORE-1 algorithm, having only a simple 2 dimensional cascaded structure with only one type of functional unit, is the most basic example of this general methodology. Our intention is to implement and empirically evaluate other examples of SPORE. In particular, we intend to implement a SPORE algorithm which has many types of functional units, where each functional unit can accept inputs from more than one existing unit. Such a structure would exploit more aspects of the general SPORE methodology, while still maintaining the computational capabilities of the SPORE-1 algorithm.

In addition, the SPORE structures presented in this thesis have been designed as long, cascaded chains which can only be evaluated in a serial manner. This may not be a suitable computational structure for applications which require fast calculations (e.g. real-time control systems). For such applications, computationally parallel SPORE structures may be more appropriate. The theoretical and empirical evaluation of a parallel SPORE algorithm is a good topic for further investigation.

The main focus of this thesis has been building nonparametric regression functions for the purpose of predicting future outputs given new inputs. However, traditionally many nonparametric regression techniques have been used as analytical tools to determine which inputs are important and how they interact. It would be interesting to investigate the possibility of using SPORE as such an analytical tool. If this were possible, it would allow analysis of input spaces for problems which have many input variables.

An additional topic for further study is a more complete analysis of the theoretical properties of the SPORE structure. This thesis gives rate of convergence results as a function of the number of learning variables, and gives conditions under which these rates are valid. Although empirical and theoretical evidence suggests that these conditions are not overly restrictive, there are as yet no theoretical results which analyze how useful or interesting this class of functions is. In addition, no rate of convergence results, as a function of the number of parameters in the SPORE regression function, are given in this thesis. Both of these theoretical questions are topics for future research.

A specific theoretical investigation which this thesis has not addressed is the full computational cost of a SPORE learning algorithm which uses domain subdivision. We know that the computational cost of a SPORE construction is linear with dimension, in each of the resulting subdivisions. However, we don't know how many subdivisions are required for a specific class of target functions. Empirical evidence presented in this thesis suggests that subdivision of the input space is not necessary (at least for the regression examples studied here), however, this may not be true in the general case. In fact, regression examples may exist where an exponential growth in the number of subdomains may be needed, as the dimension of the target function increases. This is clearly a topic for future investigation.

Finally, in this thesis, we demonstrated that it is possible to do very high dimensional human-to-robot skill transfer if the operator demonstrates the task using only the robot's sensors and actuators. This is a promising result which we have only started to explore. Many more experiments need to be done in order to fully evaluate the efficacy of this approach to robot programming.

Bibliography

Alfeld, P. (1989). Scattered data interpolation in three of more variables. In Lyche, T. and Schumaker, L. L., editors, *Mathematical Methods in Computer Aided Geometric Design*, pages 1–33. Academic Press, Boston, MA.

Asada, H. (1990). Teaching and learning of compliance using neural nets: Representation and generation of nonlinear compliance. In *Proceedings of the 1990 IEEE International Conference on Robotics and Automation*, pages 1237–1244. IEEE.

Barron, A. R. (1993). Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Trans. Inform. Theory*, 39(3):930–945.

Barron, A. R. (1994). Approximation and estimation bounds for artificial neural networks. *Machine Learning*, 14:115–133.

Bellman, R. E. (1961). *Adaptive Control Processes*. Princeton University Press.

Breiman, L. (1993). Hinging hyperplanes for regression, classification, and function approximation. *IEEE Trans. Inform. Theory*, 39(3):999–1013.

Breiman, L. (1996a). Bagging predictors. *Machine Learning*, 24:123.

Breiman, L. (1996b). Heuristics of instability and stabilization in model selection. *Ann. Statist.*, 24(6):2350–2383.

Breiman, L. (1996c). Stacked regressions. *Machine Learning*, 24:49.

Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (1984). *Classification and Regression Trees*. Wadsworth International Group, Belmont, California.

Broomhead, D. S. and Lowe, D. (1988). Multivariable functional interpolation and adaptive networks. *Complex Systems*, 2:321–355.

Casdagli, M. (1989). Nonlinear prediction of chaotic time-series. *Physica D*, 35:335–356.

Chen, H. (1991a). Estimation of a projection-pursuit type regression model. *Ann. Statist.*, 19(1):142–157.

Chen, Z. (1991b). Interactive spline models and their convergence rates. *Ann. Statist.*, 19(4):1855–1868.

Cybenko, G. (1989). Approximations by superpositions of a sigmoidal function. *Math. Contr. Sygnals, Syst.*, 2:303–314.

Edelman, S. and Poggio, T. (1990). Bringing the grandmother back into the picture: A memory-based view of object recognition. *MIT A.I. Memo No. 1181*.

Efron, B. (1983). Estimating the error rate of a prediction rule: Some improvements on cross-validation. *Journal of the American Statistical Association*, 78:316–331.

Elder IV, J. F. and Brown, D. E. (1995). Induction and polynomial networks. In *IEEE Int. Conf. on Sys. Man and Cyb.*, pages 874–879.

Fahlman, S. E. and Lebiere, C. (1990). The cascade-correlation learning architecture. In *Advances in Neural Information Processing Systems 2*, pages 524–532.

Farlow, S. J. (1984). The gmdh algorithm. In Farlow, S. J., editor, *Self-Organizing Methods in Modeling: GMDH Type Algorithms*, pages 1–24. Marcel Dekker, Inc., New York, NY.

Friedman, J. H. (1991). Multivariate adaptive regression splines. *Ann. Statist.*, 19:1–141.

Friedman, J. H. (1994a). Flexible metric nearest neighbor classification. *Technical Report, Stanford University, Department of Statistics*.

Friedman, J. H. (1994b). An overview of predictive learning and function approximation. In Cherkassky, V., Friedman, J. H., and Wechsler, H., editors, *From Statistics to Neural Networks*, pages 1–61. Springer-Verlag.

Friedman, J. H. and Stuetzle, W. (1981). Projection pursuit regression. *J. Amer. Statist. Assoc.*, 76(376):1–141.

Girosi, F. (1994). Regularization theory, radial basis functions and networks. In Cherkassky, V., Friedman, J. H., and Wechsler, H., editors, *From Statistics to Neural Networks*, pages 167–187. Springer-Verlag.

Golub, G. H. and Van Loan, C. F. (1993). *Matrix Computations*. The John Hopkins University Press, Baltimore and London.

Grudic, G. Z. and Lawrence, P. D. (1995). Human-to-robot skill transfer via teleoperation. In *IEEE Int. Conf. on Sys. Man and Cyb.*.

Grudic, G. Z. and Lawrence, P. D. (1996). Human-to-robot skill transfer using the spore approximation. In *IEEE Int. Conf. on Rob. and Aut.*.

Grudic, G. Z. and Lawrence, P. D. (1997). Is nonparametric learning practical in very high dimensional spaces? In *To Appear: Fifteenth International Joint Conference on Artificial Intelligence (IJCAI-97)*, NAGOYA, Aichi, Japan.

Hardy, R. L. (1971). Multiquadric equations of topology and other irregular surfaces. *J. Geophys. Res.*, 76:1905–1915.

Hastie, T. J. and Tibshirani, R. J. (1986). Generalized additive models. *Statistical Science*, 1(3):297–318.

Hastie, T. J. and Tibshirani, R. J. (1990). *Generalized Additive Models*. Chapman and Hall.

Hastie, T. J. and Tibshirani, R. J. (1994). Nonparametric regression and classification. In Cherkassky, V., Friedman, J. H., and Wechsler, H., editors, *From Statistics to Neural Networks*, pages 1–61. Springer-Verlag.

Hastie, T. J. and Tibshirani, R. J. (1996). Discriminant adaptive nearest neighbor classification and regression. In *Advances in Neural Information Processing Systems 8*, pages 409–415. MIT Press, Cambridge MA.

Haykin, S. S. (1994). *Neural networks : a comprehensive foundation*. Macmillan, New York.

Hornik, K., Stinchcombe, M., and White, H. (1989). Multi-layer feedforward networks are universal approximators. *Neural Networks*, 2:359–366.

Huber, P. J. (1985). Projection pursuit. *Ann. Statist.*, 13:435–475.

Igelnik, B. and Pao, Y.-H. (1995). Stochastic choice of basis functions in adaptive function approximation and the functional-link net. *IEEE Transactions on Neural Networks*, 6(6):1320–1329.

Ivankhnenko, A. G. (1971). Polynomial theory of complex systems. *IEEE Trans. Systems, Man, and Cybernetics*, SMC-12:364–378.

Jackson, I. R. H. (1988). Convergence properties of radial basis functions. *Constructive Approximation*, 4:243–264.

Jones, L. K. (1992). A simple lemma on greedy approximation in hilbert space and convergence rates for projection pursuit regression and neural network training. *Ann. Statist.*, 20(1):608–613.

Jordan, M. I. and Jacobs, R. A. (1994). Hierarchical mixtures of experts and the em algorithm. *Neural Computation*, 6:181–214.

Juditsky, A., Hjalmarsson, H., Benveniste, A., Delyon, B., Ljung, L., Sjoberg, J., and Zhang, Q. (1995). Nonlinear black-box modeling in system identification: Mathematical foundations. *Automatica: Special Issue on Trends in System Identification*, 31(12):1725–1750.

Kolmogorov, A. N. (1957). On the representation of continuous functions of many variables by superposition of functions of one variable and addition. *Dokl. Akad. Nauk SSSR*, 114:953–956.

Kosuge, K., Fukuda, T., and Asada, H. (1991). Acquisition of human skills for robotic systems. In *Proceedings of the 1991 IEEE International Symposium on Intelligent Control*, pages 469–474. IEEE.

Kwok, T.-Y. and Yeung, D.-Y. (1997). Constructive algorithms for structural learning in feedforward neural networks for regression problems. *IEEE Transactions on Neural Networks*, 8(3):630–645.

Lorentz, G. G. (1986). *Approximation of Functions*. Chelsea Publishing Co., New York.

Lowe, D. (1995). Similarity metric learning for a variable-kernel classifier. *Neural Computation*, 7(1):72–85.

Lui, S. and Asada, H. (1992). Transferring manipulation skills to robots: Representation and acquisition of manipulation skills using process dynamics. *J. Dynamic Syst. Measur. Contr.*, 114(2):220–228.

Malitz, J. (1987). *Introduction to Mathematical Logic*. Springer-Verlag, New York, NY.

Mel, B. W. and Kock, C. (1990). Sigma-pi learning: On radial basis functions and cortical associative learning. In *Advances in Neural Information Processing Systems 2*, pages 474–481.

Michie, D., Siegelhalter, D. J., and Taylor, C. C. (1994). *Machine Learning, Neural and Statistical Classification*. Ellis Horwood, New York, NY.

Nester, J., Wasserman, W., and Kutner, M. H. (1990). *Applied Linear Statistical Models: Regression, Analysis of Variance, and Experimental Design*. IRWIN.

Parmanto, B., Munro, P. W., and Doyle, H. R. (1996). Improving committee diagnosis with resampling techniques. In Touretzky, D. S., Mozer, M. C., and Hasselmo, M. E., editors, *Advances in Neural Information Processing Systems 8*. MIT Press, Cambridge MA.

Perrone, M. P. (1993). *Improving Regression Estimation: Averaging Methods for Variance Reduction with Extensions to General Convex Measure Optimization*. PhD thesis, Department of Physics, Brown University.

Poggio, T. and Girosi, F. (1989). A theory of networks for approximation and learning. *MIT A.I. Memo No. 1140*.

Poggio, T. and Girosi, F. (1990). Extensions of a theory of networks for approximation and learning: Dimensionality reduction and clustering. *MIT A.I. Memo No. 1167*.

Pomerleau, D. A. (1993a). Input reconstruction reliability estimation. In *NIPS5*, pages 279–286. Morgan Kaufmann Pub.

Pomerleau, D. A. (1993b). *Neural Network Perception for Mobile Robot Guidance*. Kluwer Academic Publishers, Boston/Dordrecht/London.

Press, W. H., Flannery, B. P., Teukolsky, S. A., and Vetterling, W. T. (1988). *Numerical Recipes in C*. Cambridge University Press, New York NY.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA.

Rasmussen, C. A. (1996). A practical monte carlo implementation of bayesian learning. In Touretzky, D. S., Mozer, M. C., and Hasselmo, M. E., editors, *Advances in Neural Information Processing Systems 8*. MIT Press, Cambridge MA.

Reed, R. (1993). Pruning algorithms - a survey. *IEEE Transactions on Neural Networks*, 4(5):740–747.

Renals, S. and Rohwer, R. (1989). Phoneme classification experiments using radial basis functions. In *Proceedings of the International Joint Conference on Neural Networks*, pages I–461–I–467.

Rouse, W. B., Hammer, J. M., and Lewis, C. M. (1989). On capturing human skills and knowledge: Algorithmic approaches to model identification. *IEEE Trans. Syst. Man Cybernet.*, 19(3):558–573.

Saha, A. and Keeler, J. D. (1990). Algorithms for better representation and faster learning in radial basis function networks. In *Advances in Neural Information Processing Systems 2*, pages 482–489.

Sanger, T. D. (1991). A tree-structured adaptive network for function approximation in high-dimensional spaces. *IEEE Transactions on Neural Networks*, 2(2).

Shimokura, K. and Liu, S. (1991). Programming deburring robots based on human demonstration with direct burr size measurements. In *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, pages 469–474. IEEE.

Sjoberg, J., Zhang, Q., Ljung, L., Benveniste, A., Delyon, B., Glorennec, P. Y., Hjalmarsson, H., and Juditsky, A. (1995). Nonlinear black-box modeling in system identification: A unified overview. *Automatica: Special Issue on Trends in System Identification*, 31(12):1691–1724.

Stone, C. J. (1982). Optimal global rates of convergence for nonparametric regression. *Ann. Statist.*, 10:1040–1053.

Tenorio, M. F. and Lee, W.-T. (1990). Self-organizing network for optimum supervised learning. *IEEE Trans. on Neur. Net.*, 1(1):100–110.

Vitushkin, A. G. (1954). On hilbert's thirteenth problem. *Dokl. Akad. Nauk. SSSR*, 95:701–704.

Vitushkin, A. G. and Henkin, G. M. (1967). Linear superposition of functions. *Russian Math. Surveys*, 22:77–125.

Widrow, B., McCool, J. M., Larimore, M. G., and Johnson, C. R. (1976). Stationary and non-stationary learning characteristics of the LMS adaptive filter. *Proc. IEEE*, 64(8):1151–1162.

Wolpert, D. (1992). Stacked generalization. *Neural Networks*, 5.

Yang, J., Xu, Y., and Chen, C. S. (1994). Hidden markov model approach to skill learning and its application to telerobotics. *IEEE Trans. Robotics and Automat.*, 10(5):421–631.

Yao, X. and Lui, Y. (1997). A new evolutionary system for evolving artificial neural networks. *IEEE Transactions on Neural Networks*, 8(3):694–713.

Zhang, Q. (1997). Using wavelet network in nonparametric estimation. *IEEE Transactions on Neural Networks*, 8(2):227–236.

Appendix A. Approximating 3 Dimensional Functions as a Cascade

THEOREM I: *Let $\mathcal{U}(x,y,z)$ be a bounded continuous 3 dimensional function defined on the dense domain $(x,y,z) \in U$, where $U \subset \Re^3$ and U has finite non-zero volume. Then there exists, for all $b \in \{1,2,\ldots,b_{\max}\}$ where $b_{\max} > 0$ ($b_{\max} \in \mathbb{I}$),*

a. *a set of bounded continuous 2 dimensional functions $\mathcal{V}_b(x,y)$ defined on $(x,y) \in S_b$, where $S_b \subset \Re^2$ and $\forall i,j \in \{1,2,\ldots,b_{max}\}[(i \neq j) \Rightarrow S_i \cap S_j \equiv \emptyset]$; and,*

b. *a second set of bounded continuous 2 dimensional functions $\mathcal{W}_b(\mathcal{V}_b, z)$ defined on $(\mathcal{V}_b, z) \in Q_b \subset \Re^2$,*

such that, for all $\epsilon > 0$ ($\epsilon \in \Re$), the following is true:

$$\forall (x,y,z) \in U \ \Big[\exists b \in \{1,2,\ldots,b_{\max}\} \\ \big[((x,y) \in S_b) \wedge (|\mathcal{W}_b(\mathcal{V}_b(x,y),z) - \mathcal{U}(x,y,z)| < \epsilon)\big]\Big]. \tag{1A}$$

The following are some basic definitions which are used throughout the proof of THEOREM I. Let $\delta > 0$ be an arbitrarily small real number. Create a uniformly spaced grid of points in $\Re^3$ at all points $(l \cdot \delta, m \cdot \delta, n \cdot \delta)$, such that $l, m, n \in \mathbb{I}$. Define the set $\mathbb{U}$ to be:

$$\mathbb{U} = \{(l,m,n) | (l,m,n \in \mathbb{I}) \wedge ((l \cdot \delta, m \cdot \delta, n \cdot \delta) \in U)\}, \tag{2A}$$

where the set U is defined in THEOREM I. Let $l_1, m_1 \in \mathbb{I}$ be fixed such that $\exists n[(l_1, m_1, n) \in \mathbb{U}]$. Then, $\mathcal{U}(l_1 \cdot \delta, m_1 \cdot \delta, n \cdot \delta)$ (the function $\mathcal{U}(x,y,z)$ is defined in THEOREM I) is a 1 dimensional function defined for all n such that $(l_1, m_1, n) \in \mathbb{U}$. Let $l_1, m_1, l_2, m_2 \in \mathbb{I}$ be fixed such that

$\exists n[((l_1, m_1, n) \in \mathbb{U}) \wedge ((l_2, m_2, n) \in \mathbb{U})]$. Now one can define a norm, symbolized by $\|\cdot\|_{\max}$, between the two 1 dimensional functions $\mathcal{U}(l_1 \cdot \delta, m_1 \cdot \delta, n \cdot \delta)$ and $\mathcal{U}(l_2 \cdot \delta, m_2 \cdot \delta, n \cdot \delta)$ as:

$$\begin{aligned} &\|\mathcal{U}(l_1 \cdot \delta, m_1 \cdot \delta, n \cdot \delta) - \mathcal{U}(l_2 \cdot \delta, m_2 \cdot \delta, n \cdot \delta)\|_{\max} = \\ &\max_{n \in \mathbb{N}} |\mathcal{U}(l_1 \cdot \delta, m_1 \cdot \delta, n \cdot \delta) - \mathcal{U}(l_2 \cdot \delta, m_2 \cdot \delta, n \cdot \delta)|, \end{aligned} \tag{3A}$$

where the set $\mathbb{N}$ is defined as:

$$\mathbb{N} = \{n | (n \in \mathbb{I}) \wedge ((l_1, m_1, n) \in \mathbb{U}) \wedge ((l_2, m_2, n) \in \mathbb{U})\}. \tag{4A}$$

Note that the norm

$$\|\mathcal{U}(l_1 \cdot \delta, m_1 \cdot \delta, n \cdot \delta) - \mathcal{U}(l_2 \cdot \delta, m_2 \cdot \delta, n \cdot \delta)\|_{\max} \tag{5A}$$

is defined if and only if $\exists n[((l_1, m_1, n) \in \mathbb{U}) \wedge ((l_2, m_2, n) \in \mathbb{U})]$.

The concept of an *S-ordering* of the 1 dimensional functions $\mathcal{U}(l \cdot \delta, m \cdot \delta, n \cdot \delta)$ (defined $\forall n$ such that $(l, m, n) \in \mathbb{U}$) with respect to the norm $\|\cdot\|_{\max}$, is defined next (note that an *S-ordering* is similar to a *well ordering* defined in [Malitz, 1987, pp. 21–23], with the addition of some other conditions given below). Define the set $\mathbb{T}$ to be:

$$\mathbb{T} = \{(l, m) | (l, m \in \mathbb{I}) \wedge (\exists n[(l, m, n) \in \mathbb{U}])\}. \tag{6A}$$

Let $\mathbb{S} \subseteq \mathbb{T}$ represent a set of 1 dimensional functions $\mathcal{U}(l \cdot \delta, m \cdot \delta, n \cdot \delta)$, where $(l, m) \in \mathbb{S}$, which are S-ordered. Let the set $\mathbb{V}^1 \subseteq \mathbb{S}$ be defined such that $\forall (l^1, m^1) \in \mathbb{V}^1$ the 1 dimensional functions $\mathcal{U}(l^1 \cdot \delta, m^1 \cdot \delta, n \cdot \delta)$ are first in the S-ordering. Let the set $\mathbb{V}^p \subset \mathbb{S}$ be defined such that $\forall (l^p, m^p) \in \mathbb{V}^p$ the 1 dimensional functions $\mathcal{U}(l^p \cdot \delta, m^p \cdot \delta, n \cdot \delta)$ are in order p (where

$p \in \mathbb{I}\,[p > 1]$) of the S-ordering. Let the set $\mathbb{V}^{p_m} \subset \mathbb{S}$ (where $p_m \in \mathbb{I}\,[p_m > 1]$) be defined such that $\forall (l^{p_m}, m^{p_m}) \in \mathbb{V}^{p_m}$ the 1 dimensional functions $\mathcal{U}(l^{p_m} \cdot \delta, m^{p_m} \cdot \delta, n \cdot \delta)$ are last in the S-ordering (because $\mathbb{T}$ is a set which has a finite number of elements, $\mathbb{S} \subseteq \mathbb{T}$ also has a finite number of elements and therefore there must exist a set $\mathbb{V}^{p_m} \subset \mathbb{S}$ which is last in the S-ordering). Then the S-ordering of the functions $\mathcal{U}(l \cdot \delta, m \cdot \delta, n \cdot \delta)$, where $(l, m) \in \mathbb{S}$, is defined such that the set $\mathbb{S}$, and the sets $\mathbb{V}^p$ ($\forall p \in \{1, \ldots, p_m\}$), must satisfy the following conditions:

Cond a. $\forall (l_1, m_1), (l_2, m_2) \in \mathbb{S}\big[\exists n \in \mathbb{I}[((l_1, m_1, n) \in \mathbb{U}) \wedge ((l_2, m_2, n) \in \mathbb{U})]\big]$.

Cond b. $\forall p \in \{1, 2, \ldots p_m\}$, if $(l_1, m_1) \in \mathbb{V}^p$ and $(l_2, m_2) \in \mathbb{V}^p$ then

$$\|\mathcal{U}(l_1 \cdot \delta, m_1 \cdot \delta, n \cdot \delta) - \mathcal{U}(l_2 \cdot \delta, m_2 \cdot \delta, n \cdot \delta)\|_{\max} = 0. \tag{7A}$$

Also, $\forall p \in \{1, 2, \ldots p_m\}$, if $(l_1, m_1) \in \mathbb{V}^p$ and $(l_3, m_3) \in \mathbb{S}\ [(l_3, m_3) \notin \mathbb{V}^p]$ then

$$\|\mathcal{U}(l_1 \cdot \delta, m_1 \cdot \delta, n \cdot \delta) - \mathcal{U}(l_3 \cdot \delta, m_3 \cdot \delta, n \cdot \delta)\|_{\max} \neq 0. \tag{8A}$$

Cond c. $\forall p_1 \in \{1, 2, \ldots, p_m - 2\}$, $\forall p_2 \in \{p_1 + 1, \ldots, p_m - 1\}$, $\forall p_3 \in \{p_2 + 1, \ldots, p_m\}$, if $(l_1, m_1) \in \mathbb{V}^{p_1}$, $(l_2, m_2) \in \mathbb{V}^{p_2}$, and $(l_3, m_3) \in \mathbb{V}^{p_3}$ then

$$\begin{aligned} &\|\mathcal{U}(l_3 \cdot \delta, m_3 \cdot \delta, n \cdot \delta) - \mathcal{U}(l_1 \cdot \delta, m_1 \cdot \delta, n \cdot \delta)\|_{\max} > \\ &\|\mathcal{U}(l_2 \cdot \delta, m_2 \cdot \delta, n \cdot \delta) - \mathcal{U}(l_1 \cdot \delta, m_1 \cdot \delta, n \cdot \delta)\|_{\max}. \end{aligned} \tag{9A}$$

Cond d. Choose $\delta > 0$ ($\delta \in \Re$) such that $\forall (x_1, y_1, z_1), (x_2, y_2, z_2) \in U$, if $\|(x_1, y_1, z_1) - (x_2, y_2, z_2)\| < \delta \cdot \sqrt{2}$, then (note that ϵ is defined in THEOREM I)

$$|\mathcal{U}(x_1, y_1, z_1) - \mathcal{U}(x_2, y_2, z_2)| < \frac{\epsilon}{5}. \tag{10A}$$

It is always possible to choose such a δ because $\mathcal{U}(x, y, z)$ is bounded and uniformly continuous on U. Given the above δ, $\forall p \in \{1, 2, \ldots, p_m - 1\}$, if $(l_1, m_1) \in \mathbb{V}^p$, and $(l_2, m_2) \in \mathbb{V}^{p+1}$ then the following condition must hold:

$$\|\mathcal{U}(l_1 \cdot \delta, m_1 \cdot \delta, n \cdot \delta) - \mathcal{U}(l_2 \cdot \delta, m_2 \cdot \delta, n \cdot \delta)\|_{\max} < \frac{\epsilon}{5}. \qquad (11A)$$

Proof of THEOREM I: The proof of THEOREM I is based on constructions of the functions $\mathcal{V}_b(x, y)$ and $\mathcal{W}_b(\mathcal{V}_b, z)$, and the domains S_b (for all $b \in \{1, 2, \ldots, b_{\max}\}$). These constructions are given in CONSTRUCTION I, CONSTRUCTION II, CONSTRUCTION III, and CONSTRUCTION IV of Appendix D.

In CONSTRUCTION I the discrete functions $\mathtt{V}_b(x, y)$ and $\mathtt{W}_b(\mathtt{V}_b, z)$, along with the discrete sets $\mathbb{S}_b$ (for all $b \in \{1, 2, \ldots, b_{\max}\}$) are constructed. In CONSTRUCTION II the discrete sets $\mathbb{S}_b$ (for all $b \in \{1, 2, \ldots, b_{\max}\}$) are used to construct the domains S_b. In CONSTRUCTION III the discrete functions $\mathtt{V}_b(x, y)$ (for all $b \in \{1, 2, \ldots, b_{\max}\}$) are used to construct the continuous functions $\mathcal{V}_b(x, y)$, and finally, in CONSTRUCTION IV the discrete functions $\mathtt{W}_b(\mathtt{V}_b, z)$ (for all $b \in \{1, 2, \ldots, b_{\max}\}$) are used to construct the continuous functions $\mathcal{W}_b(\mathcal{V}_b(x, y), z)$.

One should first note that a maximum value for $b \in \{1, 2, \ldots, b_{\max}\}$ must exist because, in **STEP 3** of CONSTRUCTION I, the number of elements in the set $\mathbb{T}_1$ is finite, and because in **STEP 9** of CONSTRUCTION I, $\mathbb{T}_b$ is updated using the equation $\mathbb{T}_b = \mathbb{T}_{b-1} - \mathbb{S}_{b-1}$, where $\mathbb{S}_{b-1}$ is always a non-empty S-ordering set (non-empty because, by the definition of an S-ordering given on page 108, if $\mathbb{T}_{b-1}$ is non-empty, then there is always at least one element in $\mathbb{T}_{b-1}$ which satisfies the conditions of an S-ordering and is therefore a member of the set $\mathbb{S}_{b-1}$).

Because CONSTRUCTION II chooses a value for $b \in \{1, 2, \ldots, b_{\max}\}$ which minimizes the function

$$\rho = \min_{(l,m) \in \mathbb{S}_b} \|(x, y) - (l \cdot \delta, m \cdot \delta)\|, \tag{12A}$$

it must be the case that $\forall (x, y, z) \in U$, $\exists b \in \{1, 2, \ldots, b_{\max}\}$ such that $(x, y) \in S_b$. Thus the constructed approximation spans the domain of definition of the function being approximated. Also, because the value of b chosen is one which is minimum in magnitude, it must also be true that $\forall i, j \in \{1, 2, \ldots, b_{\max}\}[(i \neq j) \Rightarrow S_i \cap S_j \equiv \emptyset]$ (i.e. $\forall (x, y, z) \in U$, b is unique and therefore S_b is also unique).

Given that $(x, y, z) \in U$ and $(x, y) \in S_b$, in CONSTRUCTION IV the function $\mathcal{W}_b(\mathcal{V}_b(x, y), z)$ is calculated using the equation:

$$\begin{aligned} \mathcal{W}_b(\mathcal{V}_b(x,y), z) =& (\mathtt{W}_b(\mathtt{V}_{b_1}, n_1 \cdot \delta) - \mathtt{W}_b(\mathtt{V}_{b_0}, n_0 \cdot \delta)) \cdot \frac{s_V(\mathcal{V}_b(x,y) - \mathtt{V}_{b_0})}{\delta_V} + \\ & (\mathtt{W}_b(\mathtt{V}_{b_2}, n_2 \cdot \delta) - \mathtt{W}_b(\mathtt{V}_{b_0}, n_0 \cdot \delta)) \cdot \frac{s_z(z - n_0 \cdot \delta)}{\delta} + \mathtt{W}_b(\mathtt{V}_{b_0}, n_0 \cdot \delta), \end{aligned} \tag{13A}$$

where the function $\mathcal{V}_b(x, y)$ is calculated, from CONSTRUCTION III, using the equation:

$$\begin{aligned} \mathcal{V}_b(x,y) =& (\mathtt{V}_b(l_1 \cdot \delta, m_1 \cdot \delta) - \mathtt{V}_b(l_0 \cdot \delta, m_0 \cdot \delta)) \cdot \frac{s_x(x - l_0 \cdot \delta)}{\delta} + \\ & (\mathtt{V}_b(l_2 \cdot \delta, m_2 \cdot \delta) - \mathtt{V}_b(l_0 \cdot \delta, m_0 \cdot \delta)) \cdot \frac{s_y(y - m_0 \cdot \delta)}{\delta} + \mathtt{V}_b(l_0 \cdot \delta, m_0 \cdot \delta). \end{aligned} \tag{14A}$$

One should note that the function $\mathcal{V}_b(x, y)$ is continuous and bounded on $(x, y) \in S_b$, and the function $\mathcal{W}_b(\mathcal{V}_b(x, y), z)$ is continuous and bounded on $(\mathcal{V}_b, z) \in Q_b$, because:

1. $\mathcal{U}(x, y, z)$ is continuous and bounded on $(x, y, z) \in U$;
2. Condition d of an S-ordering given on page 108;

3. **STEP 6** of CONSTRUCTION I;
4. The tessellations grids used in CONSTRUCTION III and CONSTRUCTION IV; and,
5. **STEP 2.a** and **STEP 2.b** of both CONSTRUCTION III and CONSTRUCTION IV.

Inserting equation (14A) into equation (13A), one gets:

$$\begin{aligned}\mathcal{W}_b(\mathcal{V}_b(x,y),z) = &(\mathtt{W}_b(\mathtt{V}_{b_1}, n_1\cdot\delta) - \mathtt{W}_b(\mathtt{V}_{b_0}, n_0\cdot\delta))\cdot s_V \cdot \\ &\left[\frac{(\mathtt{V}_b(l_1\cdot\delta, m_1\cdot\delta) - \mathtt{V}_b(l_0\cdot\delta, m_0\cdot\delta))}{\delta_V}\cdot\frac{s_x(x-l_0\cdot\delta)}{\delta} + \right. \\ &\frac{(\mathtt{V}_b(l_2\cdot\delta, m_2\cdot\delta) - \mathtt{V}_b(l_0\cdot\delta, m_0\cdot\delta))}{\delta_V}\cdot\frac{s_y(y-m_0\cdot\delta)}{\delta} + \\ &\left.\frac{(\mathtt{V}_b(l_0\cdot\delta, m_0\cdot\delta) - \mathtt{V}_{b_0})}{\delta_V}\right] + \\ &(\mathtt{W}_b(\mathtt{V}_{b_2}, n_2\cdot\delta) - \mathtt{W}_b(\mathtt{V}_{b_0}, n_0\cdot\delta))\cdot\frac{s_z(z-n_0\cdot\delta)}{\delta} + \mathtt{W}_b(\mathtt{V}_{b_0}, n_0\cdot\delta).\end{aligned} \tag{15A}$$

Therefore,

$$\begin{aligned}|\mathcal{W}_b(\mathcal{V}_b(x,y),z) - \mathcal{U}(x,y,z)| = &\left| (\mathtt{W}_b(\mathtt{V}_{b_1}, n_1\cdot\delta) - \mathtt{W}_b(\mathtt{V}_{b_0}, n_0\cdot\delta))\cdot s_V \cdot\right. \\ &\left[\frac{(\mathtt{V}_b(l_1\cdot\delta, m_1\cdot\delta) - \mathtt{V}_b(l_0\cdot\delta, m_0\cdot\delta))}{\delta_V}\cdot\frac{s_x(x-l_0\cdot\delta)}{\delta} + \right. \\ &\frac{(\mathtt{V}_b(l_2\cdot\delta, m_2\cdot\delta) - \mathtt{V}_b(l_0\cdot\delta, m_0\cdot\delta))}{\delta_V}\cdot\frac{s_y(y-m_0\cdot\delta)}{\delta} + \\ &\left.\frac{(\mathtt{V}_b(l_0\cdot\delta, m_0\cdot\delta) - \mathtt{V}_{b_0})}{\delta_V}\right] + \\ &(\mathtt{W}_b(\mathtt{V}_{b_2}, n_2\cdot\delta) - \mathtt{W}_b(\mathtt{V}_{b_0}, n_0\cdot\delta))\cdot\frac{s_z(z-n_0\cdot\delta)}{\delta} + \\ &\left.\mathtt{W}_b(\mathtt{V}_{b_0}, n_0\cdot\delta) - \mathcal{U}(x,y,z)\right|.\end{aligned} \tag{16A}$$

Using the Schwartz inequality and taking out the s_V, s_x, s_y, and s_z terms (this can be done because, from CONSTRUCTION III and CONSTRUCTION IV, $|s_V| = |s_x| = |s_y| = |s_z| = 1$), one gets:

$$\begin{aligned}|\mathcal{W}_b(\mathcal{V}_b(x,y),z) - \mathcal{U}(x,y,z)| \leq & \left| (\mathtt{W}_b(\mathtt{V}_{b_1}, n_1 \cdot \delta) - \mathtt{W}_b(\mathtt{V}_{b_0}, n_0 \cdot \delta)) \cdot \right. \\ & \left. \frac{(\mathtt{V}_b(l_1 \cdot \delta, m_1 \cdot \delta) - \mathtt{V}_b(l_0 \cdot \delta, m_0 \cdot \delta))}{\delta_V} \cdot \frac{(x - l_0 \cdot \delta)}{\delta} \right| + \\ & \left| (\mathtt{W}_b(\mathtt{V}_{b_1}, n_1 \cdot \delta) - \mathtt{W}_b(\mathtt{V}_{b_0}, n_0 \cdot \delta)) \cdot \right. \\ & \left. \frac{(\mathtt{V}_b(l_2 \cdot \delta, m_2 \cdot \delta) - \mathtt{V}_b(l_0 \cdot \delta, m_0 \cdot \delta))}{\delta_V} \cdot \frac{(y - m_0 \cdot \delta)}{\delta} \right| + \\ & \left| (\mathtt{W}_b(\mathtt{V}_{b_1}, n_1 \cdot \delta) - \mathtt{W}_b(\mathtt{V}_{b_0}, n_0 \cdot \delta)) \cdot \right. \\ & \left. \frac{(\mathtt{V}_b(l_0 \cdot \delta, m_0 \cdot \delta) - \mathtt{V}_{b_0})}{\delta_V} \right| + \\ & \left| (\mathtt{W}_b(\mathtt{V}_{b_2}, n_2 \cdot \delta) - \mathtt{W}_b(\mathtt{V}_{b_0}, n_0 \cdot \delta)) \cdot \frac{(z - n_0 \cdot \delta)}{\delta} \right| + \\ & \left| \mathtt{W}_b(\mathtt{V}_{b_0}, n_0 \cdot \delta) - \mathcal{U}(x,y,z) \right| . \end{aligned} \tag{17A}$$

Because of:

1. Condition d of an S-ordering given on page 108;
2. **STEP 6** of CONSTRUCTION I; and,
3. **STEP 2.a** of both CONSTRUCTION III and CONSTRUCTION IV,

the following inequalities are true:

$$\begin{aligned}|\mathtt{W}_b(\mathtt{V}_{b_1}, n_1 \cdot \delta) - \mathtt{W}_b(\mathtt{V}_{b_0}, n_0 \cdot \delta)| < \frac{\epsilon}{5}, \\ |\mathtt{W}_b(\mathtt{V}_{b_2}, n_2 \cdot \delta) - \mathtt{W}_b(\mathtt{V}_{b_0}, n_0 \cdot \delta)| < \frac{\epsilon}{5}.\end{aligned} \tag{18A}$$

Because of:

1. **STEP 2** of CONSTRUCTION I; and,

2. **STEP 1** of both CONSTRUCTION III and CONSTRUCTION IV,

the following inequalities are true:

$$\left|\frac{(x - l_0 \cdot \delta)}{\delta}\right| \leq 1,$$
$$\left|\frac{(y - m_0 \cdot \delta)}{\delta}\right| \leq 1, \qquad (19A)$$
$$\left|\frac{(z - n_0 \cdot \delta)}{\delta}\right| \leq 1.$$

Because of:

1. **STEP 1** of CONSTRUCTION I; and,
2. **STEP 2** of CONSTRUCTION I; and,
3. **STEP 1** of both CONSTRUCTION III and CONSTRUCTION IV,

the following inequality is true:

$$|\mathtt{W}_b(\mathtt{V}_{b_0}, n_0 \cdot \delta) - \mathcal{U}(x, y, z)| < \frac{\epsilon}{5}. \qquad (20A)$$

And finally, because of:

1. Condition d of an S-ordering given on page 108;
2. **STEP 6.iii** of CONSTRUCTION I;
3. **STEP 2.a** of both CONSTRUCTION III and CONSTRUCTION IV; and,
4. **STEP 2** of CONSTRUCTION IV,

the following inequalities are true:

$$\left|\frac{(\mathtt{V}_b(l_1 \cdot \delta, m_1 \cdot \delta) - \mathtt{V}_b(l_0 \cdot \delta, m_0 \cdot \delta))}{\delta_V}\right| \leq 1,$$
$$\left|\frac{(\mathtt{V}_b(l_2 \cdot \delta, m_2 \cdot \delta) - \mathtt{V}_b(l_0 \cdot \delta, m_0 \cdot \delta))}{\delta_V}\right| \leq 1, \qquad (21A)$$
$$\left|\frac{(\mathtt{V}_b(l_0 \cdot \delta, m_0 \cdot \delta) - \mathtt{V}_{b_0})}{\delta_V}\right| \leq 1.$$

Inserting the above inequalities into equation (17A), one gets:

$$|\mathcal{W}_b(\mathcal{V}_b(x,y),z) - \mathcal{U}(x,y,z)| < \frac{\epsilon}{5} + \frac{\epsilon}{5} + \frac{\epsilon}{5} + \frac{\epsilon}{5} + \frac{\epsilon}{5} = \epsilon. \qquad (22A)$$

This completes the proof of THEOREM I $\square$.

Appendix B. Constructing the SPORE-2 Structure

In the proposed structure the approximation of $f(x_0,\ldots,x_{N-1})$ is given by the following expansion (see Figure 1 for a 4 dimensional example of this):

If $(x_0,\ldots,x_{N-1})$ is an element of the subdomain corresponding to tree branch $\rho_1\sigma_1\ldots\rho_{l_{max}}\sigma_{l_{max}}$, then

$$\begin{aligned}\hat{f}(x_0,\ldots,x_{N-1}) =& g_{\rho_1\sigma_1}(h_{\rho_1\sigma_1}(x_0,x_1),x_2)+\\ & g_{\rho_1\sigma_1\rho_2\sigma_2}\Big(h_{\rho_1\sigma_1\rho_2\sigma_2}(h_{\rho_1\sigma_1},x_2),x_{\left((3) \text{ mod } (N)\right)}\Big)+\\ & \sum_{l=3}^{l_{max}} g_{\rho_1\sigma_1\ldots\sigma_l}\Big(h_{\rho_1\sigma_1\ldots\sigma_l}\Big(h_{\rho_1\sigma_1\ldots\sigma_{(l-1)}},x_{\left(l \text{ mod } (N)\right)}\Big),x_{\left((l+1) \text{ mod } (N)\right)}\Big),\end{aligned} \tag{23A}$$

where,

a. A mod $N =$ integer remainder of $(A \div N)$, and is used as an index to cycle through the independent variables of $f(x_0,\ldots,x_{N-1})$;

b. the subscript notation $\rho_1\sigma_1\ldots\rho_l\sigma_l$ ($l \in \{1,\ldots,l_{max}\}$) is standard tree notation (see complete definition below) and serves to identify the two dimensional functions $g_{\rho_1\sigma_1\ldots\rho_l\sigma_l}$ and $h_{\rho_1\sigma_1\ldots\rho_l\sigma_l}$ which act on the specific subdomain of$f(x_0,\ldots,x_{N-1})$ referred to by branch $\rho_1\sigma_1\ldots\rho_l\sigma_l$;

c. and, l_{max} is an integer defining the maximum number of terms (or levels of the tree structure) required in order to achieve some maximum approximation error

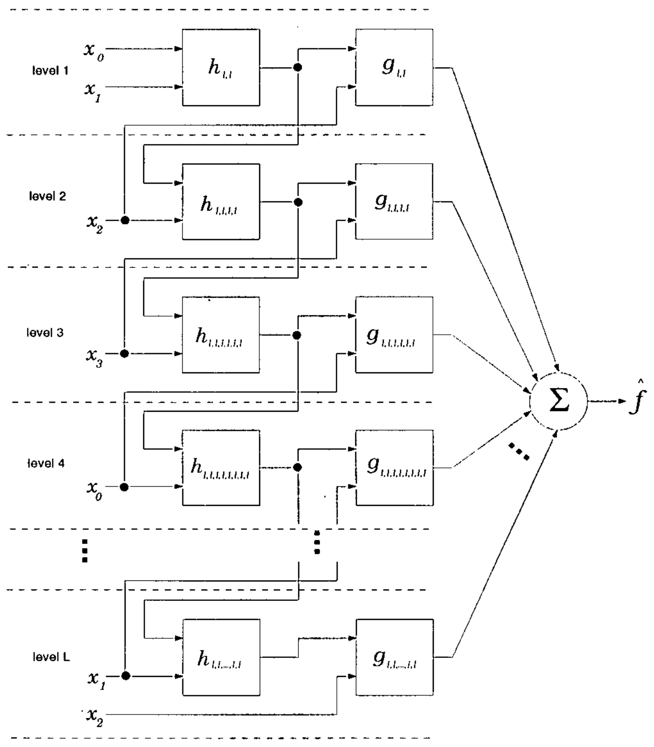
$$\max_{\mathbf{x}\in\mathcal{D}}\left|\hat{f}(\mathbf{x})-f(\mathbf{x})\right|<\epsilon \tag{24A}$$

where ϵ is an arbitrary small positive real number, and$\mathcal{D}$ is the domain of the function $f(\mathbf{x})$.

The most basic structural characteristic of the proposed N dimensional model is its tree structure. This structure is here referred to as the *approximation tree.* An example of the proposed model approximation tree structure is given in Figure 2. Two things are represented via this tree structure: 1) variables of the function being approximated are selected and acted upon by functions at each tree node; and, 2) the domains spanned by these variables are being divided up at each node in such a way that each point in the domain of the function being approximated represents one and only one path through the tree. These characteristics are explained in detail below.

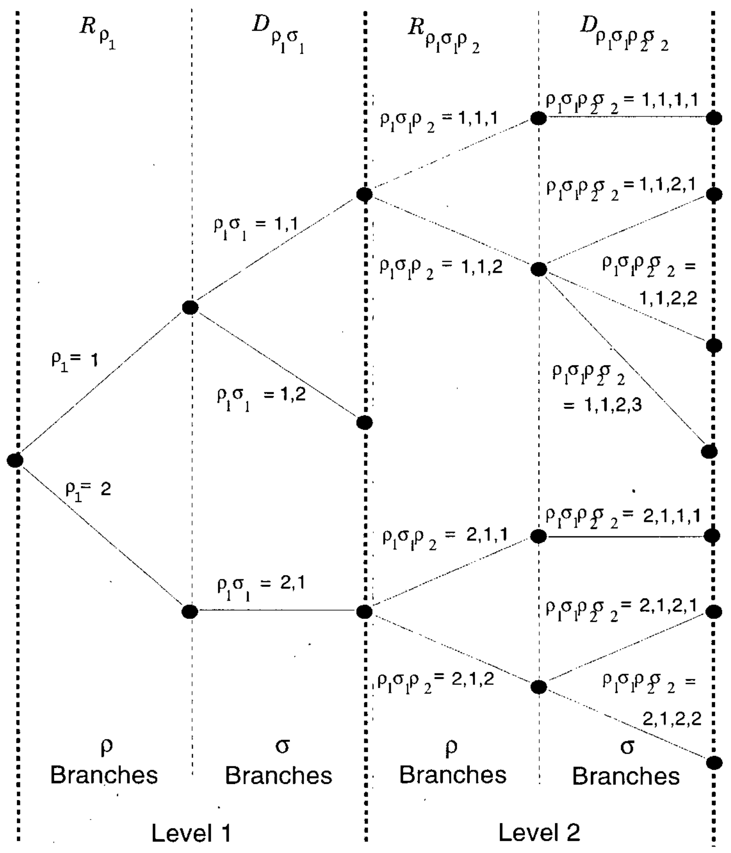
Tree Notation: Each level of the proposed model has 2 types of branches: ρ branches and σ branches. These two types of branches, as is described later, are differentiated from one another by the type of *discriminant functions* (which are functions that dictate which path through the tree is taken) associated with them. Note that the length of a branch path and the number of branches emanating from a particular tree node is not fixed a priori, and is determined based on the mapping of the function being approximated, as the model is being built.

The branches of the tree are labeled as follows: let $i_1, i_2, \ldots, i_{2l}$ be positive integers, then level 1 branches are identified by the labeling $\rho_1 = i_1$ for ρ branches and $\rho_1\sigma_1 = i_1, i_2$ for σ branches; level 2 branches are identified by the labeling $\rho_1\sigma_1\rho_2 = i_1, i_2, i_3$ for ρ branches and $\rho_1\sigma_1\rho_2\sigma_2 = i_1, i_2, i_3, i_4$ for σ branches; similarly, level l branches are identified by the labeling $\rho_1\sigma_1 \ldots \rho_l = i_1, i_2, \ldots, i_{2l-1}$ for ρ branches and $\rho_1\sigma_1 \ldots \rho_l\sigma_l = i_1, i_2, \ldots, i_{2l-1}, i_{2l}$ for σ branches.



The above is an example of how a 4 dimensional function $f(x_0, x_1, x_2, x_3)$ is approximated in the proposed structure. The approximation is symbolized above by $\hat{f}$. The above example shows the calculation of the approximation $\hat{f}$ along one path of the approximation tree. This path passes through the branches $\rho_1 = 1$, $\rho_1\sigma_1 = 1, 1$, $\rho_1\sigma_1\rho_2 = 1, 1, 1$, and $\rho_1\sigma_1\rho_2\sigma_2 = 1, 1, 1, 1$ (see Figure 2), and continues on through to branch $\rho_1\sigma_1 \ldots \rho_L\sigma_L = 1, 1, \ldots, 1, 1$, where the subscript L defines the number of levels of the tree required to get a sufficiently good approximation along this path of the tree. As explained in the body, the proposed approximation approximates functions 2 dimensions at a time. These 2 dimensional functions are symbolized as $g_{\rho_1\sigma_1 \ldots \rho_l\sigma_l}$ and $h_{\rho_1\sigma_1 \ldots \rho_l\sigma_l}$ (the integer l defines the level of the tree). At tree branch $\rho_1\sigma_1 = 1, 1$ of level $l = 1$, the functions are symbolized by $g_{1,1}$ and $h_{1,1}$; at tree branch $\rho_1\sigma_1\rho_2\sigma_2 = 1, 1, 1, 1$ of level $l = 2$, the functions are symbolized by $g_{1,1,1,1}$ and $h_{1,1,1,1}$; and so on through to branch $\rho_1\sigma_1 \ldots \rho_L\sigma_L = 1, 1, \ldots, 1, 1$ at level $l = L$, where the functions are symbolized by $g_{1,1,\ldots,1,1}$ and $h_{1,1,\ldots,1,1}$.

Figure 1 4–D Example of the Proposed Approximation


R_{ρ_1}
$D_{\rho_1\sigma_1}$
$R_{\rho_1\sigma_1\rho_2}$
$D_{\rho_1\sigma_1\rho_2\sigma_2}$
$\rho_1 = 1$
$\rho_1 = 2$
$\rho_1\sigma_1 = 1,1$
$\rho_1\sigma_1 = 1,2$
$\rho_1\sigma_1 = 2,1$
$\rho_1\sigma_1\rho_2 = 1,1,1$
$\rho_1\sigma_1\rho_2 = 1,1,2$
$\rho_1\sigma_1\rho_2 = 2,1,1$
$\rho_1\sigma_1\rho_2 = 2,1,2$
$\rho_1\sigma_1\rho_2\sigma_2 = 1,1,1,1$
$\rho_1\sigma_1\rho_2\sigma_2 = 1,1,2,1$
$\rho_1\sigma_1\rho_2\sigma_2 = 1,1,2,2$
$\rho_1\sigma_1\rho_2\sigma_2 = 1,1,2,3$
$\rho_1\sigma_1\rho_2\sigma_2 = 2,1,1,1$
$\rho_1\sigma_1\rho_2\sigma_2 = 2,1,2,1$
$\rho_1\sigma_1\rho_2\sigma_2 = 2,1,2,2$
ρ Branches
σ Branches
ρ Branches
σ Branches
Level 1
Level 2

The above is an example of the tree structure of the proposed model which partitions the input space of the function being approximated into subdomains that allow the function to be approximated using a superposition of a cascade of 2 dimensional functions. The example tree given shows only 2 levels. The notation used in the diagram is explained in detail within the text of the paper.

Figure 2 The Space Partitioning Characteristic of the Proposed Model

Each branch of the approximation tree represents a specific subdomain of the function being approximated. Thus, one can define the notion of

$$(x_0, \ldots, x_{N-1}) \in \text{tree branch } \rho_1\sigma_1 \ldots \rho_l\sigma_l \tag{25A}$$

to mean that the point $(x_0, \ldots, x_{N-1})$ (which is within the domain of the N dimensional function being approximated) is contained within the subdomain represented by branch $\rho_1\sigma_1 \ldots \rho_l\sigma_l$. For $l = 2$, this subdomain is defined as follows:

$$(x_0, \ldots, x_{N-1}) \in \text{tree branch } \rho_1\sigma_1\rho_2\sigma_2 \Leftrightarrow$$
$$((x_3, x_4, \ldots, x_{N-1}) \in R_{\rho_1} \wedge (x_0, x_1) \in D_{\rho_1\sigma_1})\wedge \tag{26A}$$
$$((x_0, x_1, x_4, x_5, \ldots, x_{N-1}) \in R_{\rho_1\sigma_1\rho_2} \wedge (h_{\rho_1\sigma_1}(x_0, x_1), x_2) \in D_{\rho_1\sigma_1\rho_2\sigma_2}),$$

where the domains $R_{\rho_1}, D_{\rho_1\sigma_1}, R_{\rho_1\sigma_1\rho_2}, D_{\rho_1\sigma_1\rho_2\sigma_2}$ and the functions $h_{\rho_1\sigma_1}$ are defined in THEOREM II. For $l > 2$, the domain represented by branch $\rho_1\sigma_1 \ldots \rho_l\sigma_l$ is defined as follows:

$$(x_0, \ldots, x_{N-1}) \in \text{tree branch } \rho_1\sigma_1 \ldots \rho_l\sigma_l \Leftrightarrow$$
$$((x_0, \ldots, x_{N-1}) \in \text{tree branch } \rho_1\sigma_1 \ldots \rho_{l-1}\sigma_{l-1})\wedge \tag{27A}$$
$$((x_{i_1}, \ldots, x_{i_{N-2}}) \in R_{\rho_1\sigma_1 \ldots \rho_l} \wedge (h_{\rho_1\sigma_1 \ldots \rho_{l-1}\sigma_{l-1}}, x_{j_1}) \in D_{\rho_1\sigma_1 \ldots \rho_l\sigma_l}),$$

where the domains $R_{\rho_1\sigma_1 \ldots \rho_l}, D_{\rho_1\sigma_1 \ldots \rho_l\sigma_l}$, the functions $h_{\rho_1\sigma_1 \ldots \rho_{l-1}\sigma_{l-1}}$, and the subscripts $i_1, \ldots, i_{N-2}, j_1$ are defined in THEOREM II. Note that the integer subscripts $i_1, \ldots, i_{N-2}, j_1$ are used to keep track of which variables of the function being approximated are used at which node of the approximation tree. Also, note that the modulo function used in equation (23A) and in the statement of THEOREM II, is used to cycle through all possible variables as often as is necessary to produce an approximation which is within some maximum allowable error.

THEOREM II: *Let $f(x_0, x_1, \ldots, x_{N-1})$ be a bounded continuous N dimensional function ($N > 3$) defined on the dense domain $\mathcal{D}$, where $\mathcal{D} \subset \Re^N$ and $\mathcal{D}$ has finite non-zero size. For* *all $l \in \mathbb{I}$ such that $l > 1$, let $i_1, i_2, \ldots, i_{N-2} \in \{0, 1, \ldots, N-1\}$ and $j_1, j_2 \in \{0, 1, \ldots, N-1\}$ such that,*

a. $j_1 = (l \text{ } mod \text{ } N)$, *and* $j_2 = ((l+1) \text{ } mod \text{ } N)$;

b. $\forall m, n \in \{1, 2, \ldots, N-2\}\big[(m \neq n \Rightarrow i_m \neq i_n) \wedge (m > n \Rightarrow i_m > i_n)\big]$; *and,*

c. $\forall m \in \{1, 2, \ldots, N-2\} \wedge \forall n \in \{1, 2\}[i_m \neq j_n]$.

Then there exists,

1. $\forall \rho_1 \in \{1, 2, 3, \ldots\}$, *a set of domains* $R_{\rho_1} \subset \Re^{N-3}$ *where*

   a. $\forall i, j \in \{1, 2, 3, \ldots\}[i \neq j \Rightarrow R_i \cap R_j = \emptyset]$, *and*

   b. $\exists i \in \{1, 2, 3, \ldots\}[(\rho_1 > i) \Rightarrow (R_{\rho_1} = \emptyset)]$;

2. $\forall \rho_1, \sigma_1 \in \{1, 2, 3, \ldots\}$, *a set of bounded continuous 2 dimensional functions* $h_{\rho_1 \sigma_1}(x_0, x_1)$ *defined on* $(x_0, x_1) \in D_{\rho_1 \sigma_1}$ *where*

   a. $D_{\rho_1 \sigma_1} \subset \Re^2$,

   b. $\forall \rho_1, i, j \in \{1, 2, \ldots\}[i \neq j \Rightarrow D_{\rho_1 i} \cap D_{\rho_1 j} = \emptyset]$, *and,*

   c. $\exists i \in \{1, 2, 3, \ldots\}[(\sigma_1 > i) \Rightarrow (D_{\rho_1 \sigma_1} = \emptyset)]$;

3. $\forall \rho_1, \sigma_1 \in \{1, 2, , 3, \ldots\}$, *a set of bounded continuous 2 dimensional functions* $g_{\rho_1 \sigma_1}(h_{\rho_1 \sigma_1}, x_2)$ *defined on* $(h_{\rho_1 \sigma_1}, x_3) \in Q_{\rho_1 \sigma_1}$, *where*

   a. $Q_{\rho_1 \sigma_1} \subset \Re^2$, *and*

   b. $\exists i \in \{1, 2, 3, \ldots\}[(\sigma_1 > i) \Rightarrow (Q_{\rho_1 \sigma_1} = \emptyset)]$;

4. $\forall l \in \{2,3,4,\ldots\}$ *and* $\forall \rho_1, \sigma_1, \ldots, \rho_l \in \{1,2,\ldots\}$, *a set of domains* $R_{\rho_1\sigma_1\ldots\rho_l} \subset \Re^{N-2}$ *where*

   a. $\forall \rho_1, \sigma_1, \ldots, \rho_{l-1}, \sigma_{l-1}, i, j \in \{1,2,\ldots\}$
   $\left[(i \neq j) \Rightarrow \left(R_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}i} \cap R_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}j} = \emptyset\right)\right]$, *and,*

   b. $\exists i \in \{1,2,3,\ldots\}[(\rho_l > i) \Rightarrow (R_{\rho_1\sigma_1\ldots\rho_l} = \emptyset)]$;

5. $\forall l \in \{2,3,4,\ldots\}$ *and* $\forall \rho_1, \sigma_1, \rho_2, \sigma_2, \ldots, \rho_l, \sigma_l \in \{1,2,3,\ldots\}$, *a set of bounded continuous 2 dimensional functions* $h_{\rho_1\sigma_1\ldots\rho_l\sigma_l}(h_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}}, x_{j_1})$ *defined on* $\left(h_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}}, x_{j_1}\right) \in D\rho_1\sigma_1\ldots\rho_l\sigma_l$ *where*

   a. $D_{\rho_1\sigma_1\ldots\rho_l\sigma_l} \subset \Re^2$,

   b. $\forall \rho_1, \sigma_1, \ldots, \rho_l, i, j \in \{1,2,\ldots\}[(i \neq j) \Rightarrow (D_{\rho_1\sigma_1\ldots\rho_l i} \cap D_{\rho_1\sigma_1\ldots\rho_l j} = \emptyset)]$, *and*

   c. $\exists i \in \{1,2,3,\ldots\}[(\sigma_l > i) \Rightarrow (D_{\rho_1\sigma_1\ldots\rho_l\sigma_l} = \emptyset)]$;

   and,

6. $\forall l \in \{2,3,4,\ldots\}$ *and* $\forall \rho_1, \sigma_1, \rho_2, \sigma_2, \ldots, \rho_l, \sigma_l \in \{1,2,3,\ldots\}$, *a set of bounded continuous 2 dimensional functions* $g_{\rho_1\sigma_1\ldots\rho_l\sigma_l}(h_{\rho_1\sigma_1\ldots\rho_l\sigma_l}, x_{j_2})$ *defined on* $(h_{\rho_1\sigma_1\ldots\rho_l\sigma_l}, x_{j_2}) \in Q_{\rho_1\sigma_1\ldots\rho_l\sigma_l}$, *where*

   a. $Q_{\rho_1\sigma_1\ldots\rho_l\sigma_l} \subset \Re^2$, *and*

   b. $\exists i \in \{1,2,3,\ldots\}[(\sigma_l > i) \Rightarrow (Q_{\rho_1\sigma_1\ldots\rho_l\sigma_l} = \emptyset)]$;

such that (given all of the above definitions), for any arbitrarily small $\epsilon_{max} \in \Re$ *where* $\epsilon_{max} > 0$, *the following is true:*

$$\begin{gathered}\forall(x_0, x_1, \ldots, x_{N-1}) \in \mathcal{D},\ \exists l_m \in \{1,2,3,\ldots\} \\ (\forall l \in \{1,2,3,\ldots,l_m\}(\exists(\rho_l \in \{1,2,3,\ldots\} \wedge \sigma_l \in \{1,2,3,\ldots\}))) \\ \left[\left| f(x_0, x_1, \ldots, x_{N-1}) - \left(\sum_{l=1}^{l_m} \Omega_l\right)\right| < \epsilon_{max}\right],\end{gathered} \tag{28A}$$

where Ω_l is defined as follows:

i. *for $l = 1$, $((x_0, x_1) \in D_{\rho_1\sigma_1} \wedge (x_3, \ldots, x_{N-1}) \in R_{\rho_1}) \Rightarrow \Omega_1 = g_{\rho_1\sigma_1}(h_{\rho_1\sigma_1}(x_0, x_1), x_2)$,*

ii. $\forall l \in \{2,3,4,\ldots,l_m\}$, *if* $(x_0, \ldots, x_{N-1}) \in$ tree branch $\rho_1\sigma_1 \ldots \rho_l\sigma_l$ *(note that the concept of "$(x_o, \ldots, x_{N-1})$ an element of tree branch $\rho_1\sigma_1 \ldots \rho_l\sigma_l$" is defined on page 119), then*

$$\Omega_l = g_{\rho_1\sigma_1\ldots\rho_l\sigma_l}\big(h_{\rho_1\sigma_1\ldots\rho_l\sigma_l}\big(h_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}}, x_{j_1}\big), x_{j_2}\big). \tag{29A}$$

Proof: The proof of THEOREM II is based on constructions of the functions $h_{\rho_1\sigma_1\ldots\rho_l\sigma_l}$ and $g_{\rho_1\sigma_1\ldots\rho_l\sigma_l}$, and the domains $R_{\rho_1\sigma_1\ldots\rho_l}$ and $D_{\rho_1\sigma_1\ldots\rho_l\sigma_l}$ (for all $l \in \{1,2,3,\ldots\}$ and $\forall \rho_1, \sigma_1, \ldots, \rho_l, \sigma_l \in \{1,2,3,\ldots\}$). These constructions are given in CONSTRUCTION V of Appendix D.

LEMMA I: *Let $\left(x_3^1, x_4^1, \ldots, x_{N-1}^1\right) \in \Re^{N-3}$ be a single point which satisfies equation (83A). Then there exists a domain $R_{\rho_1} \subset \Re^{N-3}$ such that $\left(x_3^1, x_4^1, \ldots, x_{N-1}^1\right) \in R_{\rho_1}$ and the conditions of* **STEP 3.ii** *of CONSTRUCTION V are satisfied.*

Proof of LEMMA I: Let $\gamma \in \Re$ such that $\gamma \geq 0$. Then, define the domain R_{ρ_1} as follows:

$$\begin{aligned} R_{\rho_1} = \Big\{(x_3,\ldots,x_{N-1}) | \Big((x_3,\ldots,x_{N-1}) \in \Re^{N-3}\Big) \wedge \\ (\exists (x_0,x_1,x_2)[(x_0,\ldots,x_{N-1}) \in \mathcal{D}]) \wedge \\ \left(\left\|(x_3,\ldots,x_{N-1}) - (x_3^1,\ldots,x_{N-1}^1)\right\| \leq \gamma\right) \wedge \\ (\forall i \in \{0,\ldots,\rho_1 - 1\}[(x_3,\ldots,x_{N-1}) \notin R_i])\Big\}. \end{aligned} \tag{30A}$$

Note that one can always choose a domain R_{ρ_1} which satisfies the above condition because the domain $\mathcal{D}$ of the function being approximated, is dense, and because equation (83A) is true. Note that the above R_{ρ_1}, by definition, satisfies the condition of **STEP 3.ii.a** and the condition of **STEP 3.ii.b** of CONSTRUCTION V.

Given the above definition of R_{ρ_1}, for $\gamma = 0$ equation (85A) becomes:

$$\mathcal{P}_{\rho_1}(x_0,x_1,x_2) = f\big(x_0,x_1,x_2,x_3^1,x_4^1\ldots,x_{N-1}^1\big). \tag{31A}$$

Because $f(x_0,x_1,\ldots,x_{N-1})$ is continuous within the dense domain $\mathcal{D}$, it follows then that as γ approaches zero,

$$\max_{T_{\rho_1}} |\mathcal{P}_{\rho_1}(x_0,x_1,x_2) - f(x_0,x_1\ldots,x_{N-1})| \to 0. \tag{32A}$$

By the definition of continuous functions, this implies that $\mathcal{P}_{\rho_1}(x_0,x_1,x_2)$ is also continuous. Thus, because the domain $\mathcal{D}$ is dense, there must exist a $\gamma > 0$ such that equation (88A) is satisfied. Therefore, there must exist a domain R_{ρ_1} which satisfies the condition of **STEP 3.ii.d** of CONSTRUCTION V.

Finally, if one chooses any value of $\gamma > 0$ which satisfies equation (88A), then the condition of **STEP 3.ii.c** of CONSTRUCTION V is also satisfied because R_{ρ_1} is dense and because the domain $\mathcal{D}$ of the function being approximated, $f(x_0, x_1, \ldots, x_{N-1})$, is also dense and finite in size.

This completes the proof of LEMMA I.

As shown in the proof of LEMMA I, the function $\mathcal{P}_{\rho_1}(x_0, x_1, x_2)$ is continuous and therefore, by THEOREM I, the construction in **STEP 3.iii** of CONSTRUCTION V is valid. Also, by THEOREM I, the functions $h_{\rho_1 b}(x_0, x_1)$ and $g_{\rho_1 b}(h_{\rho_1 b}, x_2)$ are continuous and therefore the function $f_{\rho_1 \sigma_1}(x_0, \ldots, x_{N-1})$, constructed in **STEP 3.iv** of CONSTRUCTION V as

$$f_{\rho_1 \sigma_1}(x_0, \ldots, x_{N-1}) = f(x_0, \ldots, x_{N-1}) - g_{\rho_1 \sigma_1}(h_{\rho_1 \sigma_1}(x_0, x_1), x_2), \tag{33A}$$

is also continuous within the domain $\mathrm{Dom}_{\rho_1 \sigma_1}$. Note that, because the domains $D_{\rho_1 \sigma_1}, R_{\rho_1}, U$ and $\mathcal{D}$ are all dense and of finite size, then by **STEP 3.iv** of CONSTRUCTION V, the domain $\mathrm{Dom}_{\rho_1 \sigma_1}$ is also dense and of finite size. Also, by THEOREM I and **STEP 3.iv** of CONSTRUCTION V, the union of the domains $\mathrm{Dom}_{\rho_1 \sigma_1}$ span the domain $\mathcal{D}$; i.e.

$$\mathcal{D} = \bigcup_{\substack{i \in \{1, 2, \ldots\} \\ j \in \{1, 2, \ldots\}}} \mathrm{Dom}_{ij}. \tag{34A}$$

Next, by THEOREM I and **STEP 3.iii** of CONSTRUCTION V, the following is true:

$$\max_{\mathrm{Dom}_{\rho_1 \sigma_1}} |\mathcal{P}_{\rho_1}(x_0, x_1, x_2) - g_{\rho_1 \sigma_1}(h_{\rho_1 \sigma_1}(x_0, x_1), x_2)| < \epsilon_\beta. \tag{35A}$$

Thus, given that equation (88A) of CONSTRUCTION V is satisfied, one can rewrite equation (88A) as follows:

$$\max_{\mathrm{Dom}_{\rho_1 \sigma_1}} |f(x_0, \ldots, x_{N-1}) - \mathcal{P}_{\rho_1}(x_0, x_1, x_2)| +$$
$$\max_{\mathrm{Dom}_{\rho_1 \sigma_1}} |\mathcal{P}_{\rho_1}(x_0, x_1, x_2) - g_{\rho_1 \sigma_1}(h_{\rho_1 \sigma_1}(x_0, x_1), x_2)| \leq \beta \cdot \left(\max_{\mathrm{Dom}_{\rho_1 \sigma_1}} |f(x_0, \ldots, x_{N-1})| \right). \tag{36A}$$

Using the Schwartz inequality, the above equation becomes:

$$\max_{\text{Dom}_{\rho_1\sigma_1}} \Big| f(x_0,\ldots,x_{N-1}) - \mathcal{P}_{\rho_1}(x_0,x_1,x_2) + \mathcal{P}_{\rho_1}(x_0,x_1,x_2) - g_{\rho_1\sigma_1}(h_{\rho_1\sigma_1}(x_0,x_1),x_2) \Big| \leq \beta \cdot \left(\max_{\text{Dom}_{\rho_1\sigma_1}} |f(x_0,\ldots,x_{N-1})| \right), \tag{37A}$$

or, simplifying the equation, one gets

$$\max_{\text{Dom}_{\rho_1\sigma_1}} \Big| f(x_0,\ldots,x_{N-1}) - g_{\rho_1\sigma_1}(h_{\rho_1\sigma_1}(x_0,x_1),x_2) \Big| \leq \beta \cdot \left(\max_{\text{Dom}_{\rho_1\sigma_1}} |f(x_0,\ldots,x_{N-1})| \right), \tag{38A}$$

which can be expressed as

$$\max_{\text{Dom}_{\rho_1\sigma_1}} \Big| f_{\rho_1\sigma_1}(x_0,\ldots,x_{N-1}) \Big| \leq \beta \cdot \left(\max_{\text{Dom}_{\rho_1\sigma_1}} |f(x_0,\ldots,x_{N-1})| \right). \tag{39A}$$

Now consider the statement and proof of LEMMA II.

LEMMA II: *Let* $\left(x^1_{i_1},\ldots,x^1_{i_{N-2}}\right) \in \Re^{N-2}$ *be a single point which satisfies equation (94A). Then there exists a domain* $R_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}\rho_l} \subset \Re^{N-2}$ *such that* $\left(x^1_{i_1},\ldots,x^1_{i_{N-2}}\right) \in R_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}\rho_l}$ *and conditions of* **STEP 7.iii.2** *of CONSTRUCTION V are satisfied.*

Proof of LEMMA II: The lemma is first proven for the case $l = 2$. Let $\gamma \in \Re$ such that $\gamma \geq 0$. Then, define the domain $R_{\rho_1\sigma_1\rho_2}$ as follows:

$$\begin{aligned} R_{\rho_1\sigma_1\rho_2} = \Big\{ (x_{i_1},\ldots,x_{i_{N-2}}) | & \Big((x_{i_1},\ldots,x_{i_{N-2}}) \in \Re^{N-2} \Big) \wedge \\ & (\exists (x_{j_1},x_{j_2})[(x_0,\ldots,x_{N-1}) \in D]) \wedge \\ & ((x_0,\ldots,x_{N-1}) \in \text{tree branch } \rho_1\sigma_1) \wedge \\ & (\|(x_{i_1},\ldots,x_{i_{N-2}}) - (x^1_{i_1},\ldots,x^1_{i_{N-2}})\| \leq \gamma) \wedge \\ & (\forall i \in \{0,\ldots,\rho_2 - 1\}[(x_{i_1},\ldots,x_{i_{N-2}}) \notin R_{\rho_1\sigma_1 i}]) \Big\}. \end{aligned} \tag{40A}$$

Note that one can always choose a domain $R_{\rho_1\sigma_1\rho_2}$ which satisfies the above condition because, as shown above, the domain, $\text{Dom}_{\rho_1\sigma_1}$, of the function being approximated, $f_{\rho_1\sigma_1}(x_0,\ldots,x_{N-1})$, is dense and because equation (94A) is true. Note that the above $R_{\rho_1\sigma_1\rho_2}$, by definition, satisfies the condition of **STEP 7.iii.2.a** and the condition of **STEP 7.iii.2.b** of CONSTRUCTION V.

Given the above definition of $R_{\rho_1\sigma_1\rho_2}$, for $\gamma = 0$ equation (97A) becomes:

$$\mathcal{P}_{\rho_1\sigma_1\rho_2}(h_{\rho_1\sigma_1}, x_2, x_3) = f_{\rho_1\sigma_1}(x_0^1, x_1^1, x_2, x_3, x_4^1, \ldots, x_{N-1}^1). \tag{41A}$$

Because the function being approximated, $f_{\rho_1\sigma_1}(x_0,\ldots,x_{N-1})$, is continuous within the dense domain $\text{Dom}_{\rho_1\sigma_1}$, and because the function $h_{\rho_1\sigma_1}$ is continuous (as proven in THEOREM I), it follows then that as γ approaches zero,

$$\max_{T_{\rho_1\sigma_1\rho_2}} |\mathcal{P}_{\rho_1\sigma_1\rho_2}(h_{\rho_1\sigma_1}, x_2, x_3) - f_{\rho_1\sigma_1}(x_0, x_1, \ldots, x_{N-1})| \to 0. \tag{42A}$$

By the definition of continuous functions, this implies that $\mathcal{P}_{\rho_1\sigma_1\rho_2}(h_{\rho_1\sigma_1}, x_2, x_3)$ is also continuous. Thus, because the domain $\text{Dom}_{\rho_1\sigma_1}$ is dense, there must exist a $\gamma > 0$ such that equation (100A) is satisfied. Therefore, there must exist a domain $R_{\rho_1\sigma_1\rho_2}$ which satisfies the condition of **STEP 7.iii.2.d** of CONSTRUCTION V.

Finally, if one chooses any value of $\gamma > 0$ which satisfies equation (100A), then the condition of **STEP 7.iii.2.c** of CONSTRUCTION V is also satisfied because $R_{\rho_1\sigma_1\rho_2}$ is dense and the domain $\text{Dom}_{\rho_1\sigma_1}$ of the function being approximated, $f_{\rho_1\sigma_1}(x_0,\ldots,x_{N-1})$, is also dense and finite in size.

This completes the proof of LEMMA II for the case $l = 2$. The proof of LEMMA II for the cases $l \in \{3, 4, \ldots\}$ is given next.

As shown above, the function $\mathcal{P}_{\rho_1\sigma_1\rho_2}(h_{\rho_1\sigma_1}, x_2, x_3)$ is continuous and therefore, by THEOREM I, the construction in **STEP 7.iii.3** of CONSTRUCTION V is valid. Also, by THEOREM I, the functions $h_{\rho_1\sigma_1\rho_2\sigma_2}$ and $g_{\rho_1\sigma_1\rho_2\sigma_2}$ are continuous and therefore the function $f_{\rho_1\sigma_1\rho_2\sigma_2}(x_0, \ldots, x_{N-1})$, constructed in **STEP 7.iii.4** of CONSTRUCTION V as

$$f_{\rho_1\sigma_1\rho_2\sigma_2}(x_0, \ldots, x_{N-1}) = f_{\rho_1\sigma_1}(x_0, \ldots, x_{N-1}) - g_{\rho_1\sigma_1\rho_2\sigma_2}(h_{\rho_1\sigma_1\rho_2\sigma_2}, x_3), \tag{43A}$$

is also continuous within the domain $\mathrm{Dom}_{\rho_1\sigma_1\rho_2\sigma_2}$. Note that, because the domains $D_{\rho_1\sigma_1\rho_2\sigma_2}, R_{\rho_1\sigma_1\rho_2}, U$ and $\mathcal{D}$ are all dense and of finite size, then by **STEP 7.iii.4** of CONSTRUCTION V, the domain $\mathrm{Dom}_{\rho_1\sigma_1\rho_2\sigma_2}$ is also dense and of finite size. Also, by THEOREM I and **STEP 7.iii.4** of CONSTRUCTION V, the union of the domains $\mathrm{Dom}_{\rho_1\sigma_1\rho_2\sigma_2}$ span the domain $\mathrm{Dom}_{\rho_1\sigma_1}$; i.e.

$$\mathrm{Dom}_{\rho_1\sigma_1} = \bigcup_{\substack{i \in \{1,2,\ldots\} \\ j \in \{1,2,\ldots\}}} \mathrm{Dom}_{\rho_1\sigma_1 ij}. \tag{44A}$$

Therefore, given that $f_{\rho_1\sigma_1\rho_2\sigma_2}(x_0, \ldots, x_{N-1})$ is continuous within the dense finite domain $\mathrm{Dom}_{\rho_1\sigma_1\rho_2\sigma_2}$, the proof of LEMMA II for the cases $l \in \{3, 4, \ldots\}$ follows by induction the proof for the case $l = 2$. Note that for $l \in \{3, 4, \ldots\}$ the domains $R_{\rho_1\sigma_1\ldots\rho_l}$ are constructed as follows:

$$\begin{aligned} R_{\rho_1\sigma_1\ldots\rho_l} = \Big\{ & (x_{i_1}, \ldots, x_{i_{N-2}}) | (x_{i_1}, \ldots, x_{i_{N-2}}) \in \Re^{N-2} \wedge \\ & (\exists (x_{j_1}, x_{j_2})[(x_0, \ldots, x_{N-1}) \in D]) \\ & ((x_0, \ldots, x_{N-1}) \in \text{tree branch } \rho_1\sigma_1 \ldots \rho_{l-1}\sigma_{l-1}) \wedge \\ & \left\| (x_{i_1}, \ldots, x_{i_{N-2}}) - (x^1_{i_1}, \ldots, x^1_{i_{N-2}}) \right\| \leq \gamma \wedge \\ & \forall i \in \{0, \ldots, \rho_l - 1\} \left[(x_{i_1}, \ldots, x_{i_{N-2}}) \notin R_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}i} \right] \Big\}. \end{aligned} \tag{45A}$$

Also, note that, by induction, and by THEOREM I and **STEP 7.iii.4** of CONSTRUCTION V, the union of the domains $\mathrm{Dom}_{\rho_1\sigma_1\ldots\rho_l\sigma_l}$ span the domain $\mathrm{Dom}_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}}$; i.e.

$$\mathrm{Dom}_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}} = \bigcup_{\substack{i\in\{1,2,\ldots\}\\ j\in\{1,2,\ldots\}}} \mathrm{Dom}_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}ij}. \tag{46A}$$

This completes the proof of LEMMA II.

Next, by THEOREM I, and **STEP 7.iii.3** of CONSTRUCTION V, the following is true for the case $l = 2$:

$$\max_{\mathrm{Dom}_{\rho_1\sigma_1\rho_2\sigma_2}} |\mathcal{P}_{\rho_1\sigma_1\rho_2}(h_{\rho_1\sigma_1}, x_2, x_3) - g_{\rho_1\sigma_1\rho_2\sigma_2}(h_{\rho_1\sigma_1\rho_2\sigma_2}, x_3)| < \epsilon_\beta. \tag{47A}$$

Thus, given that equation (100A) of CONSTRUCTION V is satisfied, one can rewrite equation (100A) as follows:

$$\begin{aligned}&\max_{\mathrm{Dom}_{\rho_1\sigma_1\rho_2\sigma_2}} |f_{\rho_1\sigma_1}(x_0,\ldots,x_{N-1}) - \mathcal{P}_{\rho_1\sigma_1\rho_2}(h_{\rho_1\sigma_1}, x_2, x_3)| + \\ &\max_{\mathrm{Dom}_{\rho_1\sigma_1\rho_2\sigma_2}} |\mathcal{P}_{\rho_1\sigma_1\rho_2}(x_0, x_1, x_2) - g_{\rho_1\sigma_1\rho_2\sigma_2}(h_{\rho_1\sigma_1\rho_2\sigma_2}(h_{\rho_1\sigma_1}, x_2), x_3)| \leq \\ &\beta\cdot\left(\max_{\mathrm{Dom}_{\rho_1\sigma_1\rho_2\sigma_2}} |f_{\rho_1\sigma_1}(x_0,\ldots,x_{N-1})|\right).\end{aligned} \tag{48A}$$

Using the Schwartz inequality, the above equation becomes:

$$\begin{aligned}&\max_{\mathrm{Dom}_{\rho_1\sigma_1\rho_2\sigma_2}} \Big| f_{\rho_1\sigma_1}(x_0,\ldots,x_{N-1}) - \mathcal{P}_{\rho_1\sigma_1\rho_2}(h_{\rho_1\sigma_1}, x_2, x_3) + \\ &\mathcal{P}_{\rho_1\sigma_1\rho_2}(x_0, x_1, x_2) - g_{\rho_1\sigma_1\rho_2\sigma_2}(h_{\rho_1\sigma_1\rho_2\sigma_2}(h_{\rho_1\sigma_1}, x_2), x_3) \Big| \leq \\ &\beta\cdot\left(\max_{\mathrm{Dom}_{\rho_1\sigma_1\rho_2\sigma_2}} |f_{\rho_1\sigma_1}(x_0,\ldots,x_{N-1})|\right),\end{aligned} \tag{49A}$$

or, simplifying the equation, one gets

$$\begin{aligned}&\max_{\mathrm{Dom}_{\rho_1\sigma_1\rho_2\sigma_2}} \Big| f_{\rho_1\sigma_1}(x_0,\ldots,x_{N-1}) - g_{\rho_1\sigma_1\rho_2\sigma_2}(h_{\rho_1\sigma_1\rho_2\sigma_2}(h_{\rho_1\sigma_1}, x_2), x_3) \Big| \leq \\ &\beta\cdot\left(\max_{\mathrm{Dom}_{\rho_1\sigma_1\rho_2\sigma_2}} |f_{\rho_1\sigma_1}(x_0,\ldots,x_{N-1})|\right).\end{aligned} \tag{50A}$$

Therefore, for the cases $l \in \{3, 4, \ldots\}$, it follows by induction that

$$\begin{aligned}\max_{\mathrm{Dom}_{\rho_1\sigma_1\cdots\rho_l\sigma_l}} \Big| & f_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}}(x_0, \ldots, x_{N-1}) - \\ & g_{\rho_1\sigma_1\ldots\rho_l\sigma_l}\big(h_{\rho_1\sigma_1\ldots\rho_l\sigma_l}\big(h_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}}, x_{j_1}\big), x_{j_2}\big) \Big| \leq \\ & \beta \cdot \left(\max_{\mathrm{Dom}_{\rho_1\sigma_1\cdots\rho_l\sigma_l}} \left| f_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}}(x_0, \ldots, x_{N-1}) \right| \right),\end{aligned} \tag{51A}$$

or, simply,

$$\begin{aligned}\max_{\mathrm{Dom}_{\rho_1\sigma_1\cdots\rho_l\sigma_l}} \Big| f_{\rho_1\sigma_1\ldots\rho_l\sigma_l}(x_0, \ldots, x_{N-1}) \Big| \leq \\ \beta \cdot \left(\max_{\mathrm{Dom}_{\rho_1\sigma_1\cdots\rho_l\sigma_l}} \left| f_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}}(x_0, \ldots, x_{N-1}) \right| \right).\end{aligned} \tag{52A}$$

Thus, given equation (39A), it follows that

$$\max_{\mathrm{Dom}_{\rho_1\sigma_1\cdots\rho_l\sigma_l}} \Big| f_{\rho_1\sigma_1\ldots\rho_l\sigma_l}(x_0, \ldots, x_{N-1}) \Big| \leq \beta^l \cdot \left(\max_{\mathrm{Dom}_{\rho_1\sigma_1\cdots\rho_l\sigma_l}} \left| f(x_0, \ldots, x_{N-1}) \right| \right). \tag{53A}$$

Therefore, because $0 < \beta < 1$ and because the function being approximated, $f(x_0, \ldots, x_{N-1})$ is bounded, it follows that there exists an $l \in \{1, 2, \ldots\}$ such that, for any arbitrarily small $\epsilon_{max} > 0$,

$$\beta^l \cdot \left(\max_{\mathrm{Dom}_{\rho_1\sigma_1\cdots\rho_l\sigma_l}} \left| f(x_0, \ldots, x_{N-1}) \right| \right) < \epsilon_{max}. \tag{54A}$$

Thus, because equation (34A) and equation (46A) are true, it must be the case that equation (28A) of THEOREM II is satisfied. This completes the proof of THEOREM II □.

Appendix C. Rate of Convergence

The approximations developed in the previous two appendices are here be applied to nonparametric regression.

Let (X, Y) be a pair of random variables such that X is of dimension N, where $N > 3$, and Y is of dimension one. Let $\theta(X) = E(Y|X)$ be the regression function of Y on the measurement variable X. Let $\widehat{\theta}_n$, $n \geq 1$, denote an estimation of θ given the n random samples $(X_1, Y_1), \ldots, (X_n, Y_n)$ of the distribution (X, Y).

Let the estimators $\widehat{\theta}_n$ be based on the nonparametric model given in equation (23A) of Appendix B. Let the two dimensional functions $g_{\rho_1\sigma_1\ldots\rho_l\sigma_l}$ and $h_{\rho_1\sigma_1\ldots\rho_l\sigma_l}$ of $\widehat{\theta}_n$ be constructed as defined in CONSTRUCTION V of Appendix D with $\theta(\mathrm{x}) = f(\mathrm{x})$, $\mathrm{x} = (x_0, x_1, \ldots, x_{N-1}) \in \mathcal{D}$. Assume that the function $\theta = f$, and the domain $\mathcal{D}$, are as defined in THEOREM II, with the additional constraint that θ is p–times differentiable ($p \geq 1$) on $\mathcal{D}$.

As defined in [Stone, 1982], assume that the distribution of Y depends on the value of the measurement variable x and takes the form $h(y|\mathrm{x}, t)\phi(dy)$, where ϕ is a measure on $\Re$ and $t = \theta(\mathrm{x})$ is the mean of the distribution as defined by:

$$t = \theta(\mathrm{x}) = \int yh(y|\mathrm{x}, t)\phi(dy), \qquad \mathrm{x} \in \mathcal{D}. \tag{55A}$$

THEOREM III*: Assume that conditions 1–3 defined in [Stone, 1982] are true. Assume that θ and $\widehat{\theta}_n$ are as defined above. Then, there exist a finite set of dense domains $\mathcal{S}_k \subseteq \mathcal{D}$, where $k \in \{1, \ldots, K\}$ (K finite), and $\bigcup_{k=1}^{K} \mathcal{S}_k \equiv \mathcal{D}$, such that, given n random samples of θ from any* *domain $\mathcal{S}_k$, the rate of convergence of $\widehat{\theta}_n$ to θ on $\mathcal{S}_k$ is* $\left\|\theta - \widehat{\theta}_n\right\|_\infty \leq O\Big((n)^{-1}\log(n)\Big)^r$ *where* $r = p/(2p+3)$.

Proof: Because the dense domains $\mathrm{Dom}_{\rho_1\sigma_1\ldots\rho_l\sigma_l}$ are finite in number and span $\mathcal{D}$ (see equation (46A)), in order to prove this theorem, it is sufficient to establish the rate of convergence of the error functions $f^n_{\rho_1\sigma_1\ldots\rho_l\sigma_l}(x_0,\ldots,x_{N-1})$, where n is the number of random samples from the corresponding domain $\mathcal{S}_k \equiv \mathrm{Dom}_{\rho_1\sigma_1\ldots\rho_l\sigma_l}$. As defined in equations (91A) and (103A) of CONSTRUCTION V in Appendix D, the error function after the construction of l levels is given by:

$$\begin{aligned} f^n_{\rho_1\sigma_1\ldots\rho_l\sigma_l}(x_0,\ldots,x_{N-1}) =&\theta(x_0,\ldots,x_{N-1}) - \Big[g_{\rho_1\sigma_1}(h_{\rho_1\sigma_1}(x_0,x_1),x_2)+ \\ &g_{\rho_1\sigma_1\rho_2\sigma_2}(h_{\rho_1\sigma_1\rho_2\sigma_2}(h_{\rho_1\sigma_1},x_2),x_3)+ \\ &\sum_{i=3}^{l} g_{\rho_1\sigma_1\ldots\sigma_i}\Big(h_{\rho_1\sigma_1\ldots\sigma_i}\Big(h_{\rho_1\sigma_1\ldots\sigma_{(i-1)}},x_{(i \bmod N)}\Big),x_{((i+1) \bmod N)}\Big)\Big], \end{aligned} \tag{56A}$$

We prove THEOREM III by showing that $\left\|f^n_{\rho_1\sigma_1\ldots\rho_l\sigma_l}(x_0,\ldots,x_{N-1})\right\|_\infty \leq O\Big((n)^{-1}\log(n)\Big)^r$ given n random samples from the corresponding domain $\mathcal{S}_k \equiv \mathrm{Dom}_{\rho_1\sigma_1\ldots\rho_l\sigma_l}$.

As defined in CONSTRUCTION V of Appendix D, the functions $h_{\rho_1\sigma_1\ldots\rho_l\sigma_l}$ and $g_{\rho_1\sigma_1\ldots\rho_l\sigma_l}$ in (56A) are constructed from the 3 dimensional functions $\mathcal{P}_{\rho_1}$ and $\mathcal{P}_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}\rho_l}$ (see equations (85A) and (97A)) which are in turn constructed from the function $\theta(\mathrm{x}) = f(\mathrm{x})$ being approximated. However, $\theta(\mathrm{x})$ is unknown, and therefore, $\mathcal{P}_{\rho_1}$ and $\mathcal{P}_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}\rho_l}$ must be approximated from n random samples from the domain $T_{\rho_1} \supseteq \mathrm{Dom}_{\rho_1\sigma_1}$ and $T_{\rho_1\sigma_1\ldots\rho_l} \supseteq \mathrm{Dom}_{\rho_1\sigma_1\ldots\rho_l\sigma_l}$ respectively (see equations (87A) and (98A)). We denote this by defining $\mathcal{P}^n_{\rho_1}$ and $\mathcal{P}^n_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}\rho_l}$ to be approximations of $\mathcal{P}_{\rho_1}$ and $\mathcal{P}_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}\rho_l}$, respectively, based on n from there respective domains. The following lemma establishes the optimal rate of convergence for the estimation of $\left\|\mathcal{P}_{\rho_1} - \mathcal{P}^n_{\rho_1}\right\|_\infty$ and $\left\|\mathcal{P}_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}\rho_l} - \mathcal{P}^n_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}\rho_l}\right\|_\infty$, as a function of n random samples.

LEMMA III: *Let the functions $\mathcal{P}_{\rho_1}$ and $\mathcal{P}_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}\rho_l}$, for all $l \in \{2,\ldots,l_m\}$, be estimated using the local average estimator of [Stone, 1982]. Then the optimal L^∞ rate of convergence of each estimator $\mathcal{P}^n_{\rho_1}$ and $\mathcal{P}^n_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}\rho_l}$ is $O\left(n^{-1}\log n\right)^r$, where $r = p/(2p+3)$, and n is the number of points used in constructing the estimator.*

Proof of LEMMA III: This lemma is proven by showing that, given the assumption in Theorem III, all functions $\mathcal{P}_{\rho_1}$ and $\mathcal{P}_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}\rho_l}$ meet the conditions defined in [Stone, 1982]. First consider the estimation of $\mathcal{P}_{\rho_1}$. By equation (85A) and equation (55A), the expected value $\mathcal{P}_{\rho_1}$ takes the following form:

$$\begin{aligned}\mathcal{P}_{\rho_1}(x_0,x_1,x_2) &= \frac{\int\limits_{R_{\rho_1}} \theta(x_0,x_1,\ldots,x_{N-1})dR_{\rho_1}}{\int\limits_{R_{\rho_1}} dR_{\rho_1}} \\ &= \frac{\int\limits_{R_{\rho_1}} \int yh(y|\mathbf{x},t)\phi(dy)dR_{\rho_1}}{\int\limits_{R_{\rho_1}} dR_{\rho_1}} \\ &= \int y \left[\frac{\int\limits_{R_{\rho_1}} h(y|\mathbf{x},t)dR_{\rho_1}}{\int\limits_{R_{\rho_1}} dR_{\rho_1}}\right]\phi(dy).\end{aligned} \tag{57A}$$

Therefore, the distribution of $\mathcal{P}_{\rho_1}$ is of the form

$$\left[\frac{\int\limits_{R_{\rho_1}} h(y|\mathbf{x},t)dR_{\rho_1}}{\int\limits_{R_{\rho_1}} dR_{\rho_1}}\right]\phi(dy). \tag{58A}$$

Because (58A) is an appropriately scaled version of $h(y|\mathrm{x},t)\phi(dy)$, it follows that given that $h(y|\mathrm{x},t)\phi(dy)$ meets conditions 1–3 defined in [Stone, 1982], then the distribution in (58A) must also meet these conditions. Also, note that because θ is p–times differentiable, then given the above calculation, it must be the case that $\mathcal{P}_{\rho_1}$ is also p–times differentiable. Therefore, given that $\mathcal{P}_{\rho_1}$ is a p–times differentiable 3 dimensional regression function, then by Theorem 1 of [Stone, 1982], the optimal L^{∞} rate of convergence is $\left\|\mathcal{P}_{\rho_1} - \mathcal{P}^n_{\rho_1}\right\|_{\infty} \leq O(n^{-1}\log n)^r$, where $r = p/(2p+3)$.

Next consider the construction of the 2 dimensional functions $h_{\rho_1\sigma_1}$ and $g_{\rho_1\sigma_1}$ in **STEP 3.iii** of CONSTRUCTION V, and functions $h_{\rho_1\sigma_1\ldots\rho_l\sigma_l}$ and $g_{\rho_1\sigma_1\ldots\rho_l\sigma_l}$, for all $l \in \{2,\ldots,l_m\}$, in **STEP 7.iii.3** of CONSTRUCTION V. As defined in CONSTRUCTION III and CONSTRUCTION IV, these functions are constructed using linear interpolation based on $\mathcal{P}_{\rho_1}$. Because of equations (91A), (97A), and (103A), in order to complete the proof of LEMMA III, we must ensure that the 2 dimensional functions $h_{\rho_1\sigma_1}$, $g_{\rho_1\sigma_1}$, $h_{\rho_1\sigma_1\ldots\rho_l\sigma_l}$ and $g_{\rho_1\sigma_1\ldots\rho_l\sigma_l}$ are at least p–times differentiable (that being the number of times that the regression function θ is differentiable). Without loss of generality, this condition can be satisfied by representing $h_{\rho_1\sigma_1}$, $g_{\rho_1\sigma_1}$, $h_{\rho_1\sigma_1\ldots\rho_l\sigma_l}$ and $g_{\rho_1\sigma_1\ldots\rho_l\sigma_l}$, for all $l \in \{2,\ldots,l_m\}$, using, for example, 2 dimensional spline approximations, of differentiability p or higher (i.e. instead of the linear interpolation scheme defined in CONSTRUCTION III and CONSTRUCTION IV).

Given that the functions $f^n_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}}$ of equation (97A) are at least p–times differentiable, the proof of the lemma for the functions $\mathcal{P}_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}\rho_l}$, for all $l \in \{2,\ldots,l_m\}$, follows the same steps outlined above for $\mathcal{P}_{\rho_1}$, and is therefore not be presented here. This completes the proof of LEMMA III.

Given LEMMA III, we can establish the rate of convergence of the error functions $f^n_{\rho_1\sigma_1\ldots\rho_l\sigma_l}(x_0,\ldots,x_{N-1})$. Consider the rate of convergence of the first function $f^n_{\rho_1\sigma_1}$. Using an analysis similar to that found in [Barron, 1994] and referring to CONSTRUCTION V of Appendix D, the L_∞ norm of $f^n_{\rho_1\sigma_1}$ is given by (unless otherwise stated, the L_∞ norm is over the domain of the respective function $f^n_{\rho_1\sigma_1\ldots\rho_l\sigma_l}(x_0,\ldots,x_{N-1})$, for all $l = 1, 2, \ldots$):

$$\left\|f^n_{\rho_1\sigma_1}\right\|_\infty \le \|f - \mathcal{P}_{\rho_1}\|_\infty + \left\|\mathcal{P}_{\rho_1} - \mathcal{P}^n_{\rho_1}\right\|_\infty \tag{59A}$$

Letting $\epsilon_p = \left\|\mathcal{P}_{\rho_1} - \mathcal{P}^n_{\rho_1}\right\|_\infty \le O\left(n^{-1}\log n\right)^r$ and using equations (88A) and (39A), equation (59A) can be written as:

$$\left\|f^n_{\rho_1\sigma_1}\right\|_\infty \le \beta\|\theta\|_\infty + \epsilon_p \tag{60A}$$

In order to guarantee convergence we choose an $\alpha \in \Re$ such that $0 < \beta < \alpha < 1$ and equation (60A) satisfies:

$$\left\|f^n_{\rho_1\sigma_1}\right\|_\infty \le \beta\|\theta\|_\infty + \epsilon_p \le \alpha\|\theta\|_\infty \tag{61A}$$

Similarly, the L_∞ norm of $f^n_{\rho_1\sigma_1\rho_2\sigma_2}$ is given by:

$$\left\|f^n_{\rho_1\sigma_1\rho_2\sigma_2}\right\|_\infty \le \left\|f^n_{\rho_1\sigma_1} - \mathcal{P}_{\rho_1\sigma_1\rho_2}\right\|_\infty + \left\|\mathcal{P}_{\rho_1\sigma_1\rho_2} - \mathcal{P}^n_{\rho_1\sigma_1\rho_2}\right\|_\infty \tag{62A}$$

Once again, letting $\epsilon_p = \left\|\mathcal{P}_{\rho_1\sigma_1\rho_2} - \mathcal{P}^n_{\rho_1\sigma_1\rho_2}\right\|_\infty \le O\left(n^{-1}\log n\right)^r$ and using equations (100A) and (50A), equation (62A) can be written as:

$$\left\|f^n_{\rho_1\sigma_1\rho_2\sigma_2}\right\|_\infty \le \beta\left\|f^n_{\rho_1\sigma_1}\right\|_\infty + \epsilon_p \le \alpha\left\|f^n_{\rho_1\sigma_1}\right\|_\infty \le \alpha^2\|\theta\|_\infty \tag{63A}$$

Thus, by induction, the L_∞ norm of $f^n_{\rho_1\sigma_1\ldots\rho_l\sigma_l}$ is given by:

$$\begin{aligned}\left\|f^n_{\rho_1\sigma_1\ldots\rho_l\sigma_l}\right\|_\infty &\le \left\|f^n_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}} - \mathcal{P}_{\rho_1\sigma_1\ldots\rho_l}\right\|_\infty + \left\|\mathcal{P}_{\rho_1\sigma_1\ldots\rho_l} - \mathcal{P}^n_{\rho_1\sigma_1\ldots\rho_l}\right\|_\infty \\ &\le \beta\left\|f^n_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}}\right\|_\infty + \epsilon_p \qquad (64A) \\ &\le \alpha\left\|f^n_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}}\right\|_\infty \le \alpha^l\|\theta\|_\infty\end{aligned}$$

Therefore, construction stops at level l when, at level $l+1$, equation (64A) is no longer true; this is defined by the following condition:

$$\left\|f^n_{\rho_1\sigma_1\ldots\rho_l\sigma_l}\right\|_\infty \le \beta\left\|f^n_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}}\right\|_\infty + \epsilon_p \le \alpha^l\|\theta\|_\infty$$

AND (65A)

$$\left\|f^n_{\rho_1\sigma_1\ldots\rho_{l+1}\sigma_{l+1}}\right\|_\infty \not\le \alpha^{l+1}\|\theta\|_\infty$$

The stopping condition given in equation (65A) is met *if and only if* the number of points used in constructing $\mathcal{P}_{\rho_1\sigma_1\ldots\rho_l\sigma_l\rho_{l+1}}$ is insufficient for error reduction; in other words, the following is true:

$$\left\|\mathcal{P}_{\rho_1\sigma_1\ldots\rho_{l+1}} - \mathcal{P}^n_{\rho_1\sigma_1\ldots\rho_{l+1}}\right\|_\infty \le O(n^{-1}\log n)^r \not\le \alpha\left\|f^n_{\rho_1\sigma_1\ldots\rho_l\sigma_l}\right\|_\infty - \beta\left\|f^n_{\rho_1\sigma_1\ldots\rho_l\sigma_l}\right\|_\infty \qquad (66A)$$

Using the inequality $\left\|f^n_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}}\right\|_\infty \le \alpha^{l-1}\|\theta\|_\infty$, then for all $f^n_{\rho_1\sigma_1\ldots\rho_l\sigma_l}$ satisfying equation (64A), the following is true:

$$\beta\alpha^{l-1}\|\theta\|_\infty + \epsilon_p \le \alpha^l\|\theta\|_\infty \qquad (67A)$$

Solving equation (67A) for $\alpha^{l-1}\|\theta\|_\infty$, we get:

$$\alpha^{l-1}\|\theta\|_\infty \ge \frac{\epsilon_p}{\alpha-\beta} \qquad (68A)$$

Using the inequality $\left\|f^n_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}}\right\|_\infty \le \alpha^{l-1}\|\theta\|_\infty$, and inserting equation (68A) into equation (64A), we get the final result:

$$\begin{aligned}\left\|f^n_{\rho_1\sigma_1\ldots\rho_l\sigma_l}\right\|_\infty &\le \beta\frac{\epsilon_p}{\alpha-\beta}+\epsilon_p \\ &\le \left(\frac{\alpha}{\alpha-\beta}\right)\epsilon_p \\ &\le O\big(n^{-1}\log n\big)^r\end{aligned} \tag{69A}$$

This completes the proof of THEOREM III □.

Appendix D. The Main Theoretical Constructions

CONSTRUCTION I: The Construction of $\mathtt{V}_b(x,y)$, $\mathtt{W}_b(\mathtt{V}_b,z)$, and $\mathbb{S}_b$

STEP 1. Choose $\delta > 0$ ($\delta \in \Re$) such that $\forall (x_1,y_1,z_1),(x_2,y_2,z_2) \in U$, if $\|(x_1,y_1,z_1)-(x_2,y_2,z_2)\| < \delta \cdot \sqrt{2}$, then

$$|\mathcal{U}(x_1,y_1,z_1) - \mathcal{U}(x_2,y_2,z_2)| < \frac{\epsilon}{5}. \tag{70A}$$

STEP 2. Create a uniformly spaced grid of points in $\Re^3$ at all points $(l\cdot\delta, m\cdot\delta, n\cdot\delta)$, such that $l,m,n \in \mathbb{I}$. Define the set $\mathbb{U}$ to be

$$\mathbb{U} = \{(l,m,n) | (l,m,n \in \mathbb{I}) \wedge ((l\cdot\delta, m\cdot\delta, n\cdot\delta) \in U)\}. \tag{71A}$$

[Thus the set $\mathbb{U}$ contains all points which are within the set U and which lie on the uniformly spaced grid of points in $\Re^3$ defined above. (Note that the italic text within the square brackets indicates comments.)]

STEP 3. Define the set $\mathbb{T}_1$ to be

$$\mathbb{T}_1 = \{(l_1,m_1) | (l_1,m_1 \in \mathbb{I}) \wedge (\exists n \in \mathbb{I}[(l_1,m_1,n) \in \mathbb{U}])\}. \tag{72A}$$

[The set $\mathbb{T}_1$ represents the set of all one dimensional functions $\mathcal{U}(l_1\cdot\delta, m_1\cdot\delta, n\cdot\delta)$, where $(l_1,m_1) \in \mathbb{T}_1$.]

STEP 4. Let $b = 1$.

[The index b indicates that discrete functions $\mathtt{V}_b(x,y)$ and $\mathtt{W}_b(\mathtt{V}_b,z)$, along with the discrete set $\mathbb{S}_b$ are constructed.]

STEP 5. Choose an arbitrary point $(l_0, m_0) \in \mathbb{T}_b$. Construct the set $\mathbb{S}_b$ such that:

a. $\mathbb{S}_b \subseteq \mathbb{T}_b$.

b. $(l_0, m_0) \in \mathbb{S}_b$;

c. $\forall (l_1, m_1) \in \mathbb{S}_b$, the 1 dimensional functions $\mathcal{U}(l_1 \cdot \delta, m_1 \cdot \delta, n \cdot \delta)$ form an S-ordering as defined above;

d. $\forall (l_2, m_2) \in \mathbb{T}_b[(l_2, m_2) \notin \mathbb{S}_b]$, the 1 dimensional functions $\mathcal{U}(l_2 \cdot \delta, m_2 \cdot \delta, n \cdot \delta)$ cannot be inserted within the S-ordering of the 1 dimensional functions $\mathcal{U}(l_1 \cdot \delta, m_1 \cdot \delta, n \cdot \delta)$, where $(l_1, m_1) \in \mathbb{S}_b$.

[Thus the set $\mathbb{S}_b$ represents an S-ordering of the functions $\mathcal{U}(l_1 \cdot \delta, m_1 \cdot \delta, n \cdot \delta)$ ($\forall (l_1, m_1) \in \mathbb{S}_b$).]

STEP 6. Let $\mathbb{V}_b^1 \subseteq \mathbb{S}_b$ be the set of all points $(l^1, m^1) \in \mathbb{S}_b$ which are first in the S-ordering of the 1 dimensional functions $\mathcal{U}(l \cdot \delta, m \cdot \delta, n \cdot \delta)$ $(\forall (l, m) \in \mathbb{S}_b)$. Similarly, $\forall p \in \{2, 3, \ldots, p_m\}$, let $\mathbb{V}_b^p \subset \mathbb{S}_b$ be the set of all points $(l^p, m^p) \in \mathbb{S}_b$ which are in position p of the S-ordering of the 1 dimensional functions $\mathcal{U}(l \cdot \delta, m \cdot \delta, n \cdot \delta)$ $(\forall (l, m) \in \mathbb{S}_b)$. EXECUTE the following steps:

i. $\forall (l^1, m^1) \in \mathbb{V}_b^1$ set the value of the 2 dimensional function $\mathtt{V}_b$ to be $\mathtt{V}_b(l^1 \cdot \delta, m^1 \cdot \delta) = 0$.

ii. Let $p = 2$.

iii. WHILE $p \leq p_m$ EXECUTE the following steps:

1. Let $\left(l^{p-1}, m^{p-1}\right) \in \mathsf{V}_b^{p-1}$. $\forall(l^p, m^p) \in \mathsf{V}_b^p$ set the value of the 2 dimensional function $\mathtt{V}_b$ to be:

$$\mathtt{V}_b(l^p \cdot \delta, m^p \cdot \delta) = \mathtt{V}_b\left(l^{p-1} \cdot \delta, m^{p-1} \cdot \delta\right) + \sigma, \tag{73A}$$

where

$$\sigma = \left\|\mathcal{U}(l^p \cdot \delta, m^p \cdot \delta, n \cdot \delta) - \mathcal{U}\left(l^{p-1} \cdot \delta, m^{p-1} \cdot \delta, n \cdot \delta\right)\right\|_{\max}. \tag{74A}$$

2. Let $p = p + 1$.

iv. $\forall p \in \{1, 2, \ldots, p_m\}$, $\forall(l^p, m^p) \in \mathsf{V}_b^p$ and $\forall n$ such that $(l^p, m^p, n) \in \mathbb{U}$, set the value of the 2 dimensional function $\mathtt{W}_b$ to be:

$$\mathtt{W}_b(\mathtt{V}_b(l^p \cdot \delta, m^p \cdot \delta), n \cdot \delta) = \mathcal{U}(l^p \cdot \delta, m^p \cdot \delta, n \cdot \delta). \tag{75A}$$

[Therefore, at all points $(l \cdot \delta, m \cdot \delta, n \cdot \delta)$, where $(l, m) \in \mathbb{S}_b$ and $n \in \mathbb{I}\,[(l, m, n) \in \mathbb{U}]$, the function $\mathtt{W}_b(\mathtt{V}_b(l \cdot \delta, m \cdot \delta), n \cdot \delta)$is equal to the function $\mathcal{U}(l \cdot \delta, m \cdot \delta, n \cdot \delta)$.]

v. Define the set $\mathbb{Q}_b$ as:

$$\begin{aligned} \mathbb{Q}_b = \Big\{ (\alpha, \beta) | \forall p \in \{1, 2, \ldots, p_m\} \Big[\forall (l^p, m^p) \in \mathsf{V}_b^p \\ \big[\forall n \text{ such that } (l^p, m^p, n) \in \mathbb{U} \\ [(\alpha = \mathtt{V}_b(l^p \cdot \delta, m^p \cdot \delta)) \wedge (\beta = n \cdot \delta)]\big]\Big]\Big\}. \end{aligned} \tag{76A}$$

[Therefore, $\mathbb{Q}_b$ contains the set of all possible independent variable values of the function $\mathtt{W}_b(\mathtt{V}_b, n \cdot \delta)$.]

STEP 7. Let $b_{\max} = b$.

[$b_{\max}$ represents the current maximum number of $\mathtt{V}_b(x,y)$ and $\mathtt{W}_b(\mathtt{V}_b,z)$ functions, and $\mathbb{S}_b$ sets.]

STEP 8. Let $b = b+1$.

[Thus the next set of $\mathtt{V}_b(x,y)$ and $\mathtt{W}_b(\mathtt{V}_b,z)$ functions, and the next set $\mathbb{S}_b$ will be constructed next.]

STEP 9. Let $\mathbb{T}_b = \mathbb{T}_{b-1} - \mathbb{S}_{b-1}$.

[The set $\mathbb{T}_b$ represents the set of all one dimensional functions $\mathcal{U}(l_b \cdot \delta, m_b \cdot \delta, n \cdot \delta)$, where $(l_b, m_b) \in \mathbb{T}_1$, which have not been used in the construction of the functions $\mathtt{V}_b(x,y)$ and $\mathtt{W}_b(\mathtt{V}_b,z)$, and sets $\mathbb{S}_b$. When $\mathbb{T}_b \equiv \emptyset$ the construction is complete.]

STEP 10. IF $\mathbb{T}_b \neq \emptyset$, GOTO **STEP 5**

ELSE END CONSTRUCTION I.

*[**STEP 5** through **STEP 9** are repeated until all one dimensional functions $\mathcal{U}(l_1 \cdot \delta, m_1 \cdot \delta, n \cdot \delta)$, where $(l_1, m_1) \in \mathbb{T}_1$, have been used in the construction of the functions $\mathtt{V}_b(x,y)$ and $\mathtt{W}_b(\mathtt{V}_b,z)$, and sets $\mathbb{S}_b$.]*

CONSTRUCTION II: The Calculation of the Domains S_b

The function of CONSTRUCTION II is, given a point $(x,y) \in \Re^2$ such that $\exists z[(x,y,z) \in U]$, determine which domain S_b, $b \in \{1,2,\ldots,b_{max}\}$, the point (x,y) belongs to.

STEP 1. Let

$$\rho_{\min} = \min_{i \in \{1,\ldots,b_{\max}\}} \left(\min_{(l,m)\in \mathbb{S}_i} \|(x,y) - (l\cdot\delta, m\cdot\delta)\| \right). \tag{77A}$$

[Therefore, $\rho_{\min}$ is the distance from the point $(x,y) \in \Re^2$ to the closest point $(l_b.\delta, m_b \cdot \delta)$ where $(l_b, m_b) \in \mathbb{S}_b$ and $b \in \{1,2,\ldots,b_{max}\}$.]

STEP 2. Let $i_{\min}$ be the minimum value of $i \in \{1,2,\ldots,b_{\max}\}$ such that

$$\begin{aligned}\rho &= \min_{(l,m)\in\mathbb{S}_i} \|(x,y)-(l\cdot\delta, m\cdot\delta)\| \\ &= \rho_{\min}.\end{aligned} \tag{78A}$$

[Thus, $i_{\min}$ is the first i which satisfies the above condition $\rho = \rho_{\min}$. Therefore, for each point (x,y), there is one and only one assigned value for $i_{\min}$.]

STEP 3. Let $b = i_{\min}$. The point (x,y) is therefore in domain S_b..

[Therefore, each point (x,y) belongs to one and only one domain S_b.]

STEP 4. END CONSTRUCTION II.

CONSTRUCTION III: The Calculation of the functions $\mathcal{V}_b(x,y)$

The function of CONSTRUCTION III is, given a point $(x,y) \in S_b$ calculate the function $\mathcal{V}_b(x,y)$.

STEP 1. Given the triangular tessellations grid in Figure 3 find the triangle which contains the point (x,y) (if the point lies at the intersection of 2 or more triangles, arbitrarily choose any one of these triangles). Let $(l_0\cdot\delta, m_0\cdot\delta)$, $(l_1\cdot\delta, m_1\cdot\delta)$, and $(l_2\cdot\delta, m_2\cdot\delta)$ be the vertices, as defined in Figure 4, of the triangle containing the point (x,y). Let $\mathtt{V}_b(l_0\cdot\delta, m_0\cdot\delta)$, $\mathtt{V}_b(l_1\cdot\delta, m_1\cdot\delta)$, and $\mathtt{V}_b(l_2\cdot\delta, m_2\cdot\delta)$ be the values of the function $\mathtt{V}_b$ at these vertices.

STEP 2. Then $\mathcal{V}_b(x,y)$ is calculated as follows:

$$\begin{aligned}\mathcal{V}_b(x,y) =&(\mathtt{V}_b(l_1\cdot\delta, m_1\cdot\delta) - \mathtt{V}_b(l_0\cdot\delta, m_0\cdot\delta))\cdot\frac{s_x(x-l_0\cdot\delta)}{\delta}+\\ &(\mathtt{V}_b(l_2\cdot\delta, m_2\cdot\delta) - \mathtt{V}_b(l_0\cdot\delta, m_0\cdot\delta))\cdot\frac{s_y(y-m_0\cdot\delta)}{\delta}+\mathtt{V}_b(l_0\cdot\delta, m_0\cdot\delta),\end{aligned} \tag{79A}$$

where for an *upper triangle* (as defined in Figure 4) $s_x = s_y = -1$ and for a *lower triangle* $s_x = s_y = +1$. Note that if any of the points (l_0, m_0), (l_1, m_1), or (l_2, m_2) are not elements of $\mathbb{S}_b$, then the value of $\mathtt{V}_b$ at that point must be chosen such that:

a. $\forall i,j \in \{0,1,2\}\left[\|\mathtt{V}_b(l_i\cdot\delta, m_i\cdot\delta) - \mathtt{V}_b(l_j\cdot\delta, m_j\cdot\delta)\| < \delta\right]$, and

b. the resulting surface $\mathcal{V}_b(x,y)$ $((x,y)\in S_b)$ is continuous.

[*Therefore the function $\mathcal{V}_b(x,y)$ is calculated as a linear interpolation between the vertices of the triangle which contains the point (x,y).*]

STEP 3. END CONSTRUCTION III.

CONSTRUCTION IV: The Calculation of the functions $\mathcal{W}_b(\mathcal{V}_b(x,y),z)$

The function of CONSTRUCTION IV is, given $\mathcal{V}_b(x,y)$ and z, where $(x,y,z)\in U$, calculate $\mathcal{W}_b(\mathcal{V}_b(x,y),z)$.

STEP 1. Given the triangular tessellations grid in Figure 5 find the triangle which contains the point $(\mathcal{V}_b(x,y),z)$ (if the point lies at the intersection of 2 or more triangles, arbitrarily choose any one of these triangles). Let $(\mathtt{V}_{b_0}, n_0\cdot\delta)$, $(\mathtt{V}_{b_1}, n_1\cdot\delta)$, and $(\mathtt{V}_{b_2}, n_2\cdot\delta)$ be the vertices, as defined in Figure 6, of the triangle containing the point (x_0, y_0). Let $\mathtt{W}_b(\mathtt{V}_{b_0}, n_0\cdot\delta)$, $\mathtt{W}_b(\mathtt{V}_{b_1}, n_1\cdot\delta)$, and $\mathtt{W}_b(\mathtt{V}_{b_2}, n_2\cdot\delta)$ be the values of the function $\mathtt{W}_b$ at these vertices.

STEP 2. Then $\mathcal{W}_b(\mathcal{V}_b(x,y),z)$ is calculated as follows:

$$\begin{aligned}\mathcal{W}_b(\mathcal{V}_b(x,y),z) =&(\mathtt{W}_b(\mathtt{V}_{b_1},n_1\cdot\delta)-\mathtt{W}_b(\mathtt{V}_{b_0},n_0\cdot\delta))\cdot\frac{s_V(\mathcal{V}_b(x,y)-\mathtt{V}_{b_0})}{\delta_V}+\\ &(\mathtt{W}_b(\mathtt{V}_{b_2},n_2\cdot\delta)-\mathtt{W}_b(\mathtt{V}_{b_0},n_0\cdot\delta))\cdot\frac{s_z(z-n_0\cdot\delta)}{\delta}+\mathtt{W}_b(\mathtt{V}_{b_0},n_0\cdot\delta),\end{aligned} \tag{80A}$$

where for an *upper triangle* (as defined in Figure 6) $s_V = s_z = -1$ and for a *lower triangle* $s_V = s_z = +1$, and

$$\delta_V = \begin{cases} \min\limits_{(\alpha,\beta)\in\mathbb{Q}_b,\alpha<\mathtt{V}_{b_0}} \|\mathtt{V}_{b_0}-\alpha\|, & \text{for } s_V=-1\\ \min\limits_{(\alpha,\beta)\in\mathbb{Q}_b,\alpha>\mathtt{V}_{b_0}} \|\alpha-\mathtt{V}_{b_0}\|, & \text{for } s_V=+1\end{cases} \tag{81A}$$

Note that if any of the points $(\mathtt{V}_{b_0}, n_0\cdot\delta)$, $(\mathtt{V}_{b_1}, n_1\cdot\delta)$, or $(\mathtt{V}_{b_2}, n_2\cdot\delta)$ are not elements of $\mathbb{Q}_b$, then the value of $\mathtt{W}_b$ at that point must be chosen such that:

a. $\forall i,j\in\{0,1,2\}\left[\left\|\mathtt{W}_b(\mathtt{V}_{b_i},n_i\cdot\delta)-\mathtt{W}_b\big(\mathtt{V}_{b_j},n_j\cdot\delta\big)\right\|<\frac{\epsilon}{5}\right]$, and

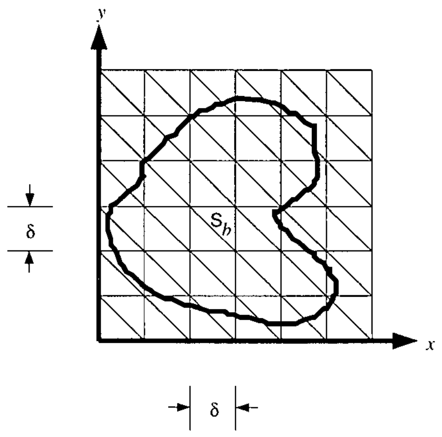
b. the resulting surface $\mathcal{W}_b(\mathcal{V}_b(x,y),z)$, $\forall(x,y,z)[((x,y)\in S_b)\wedge(x,y,z)\in U]$, is continuous.

Also, if the δ_V does not exist as calculated in (81A), then choose δ_V such that:

$$\delta_V = |\mathtt{W}_b(\mathtt{V}_{b_1},n_1\cdot\delta)-\mathtt{W}_b(\mathtt{V}_{b_0},n_0\cdot\delta)|. \tag{82A}$$

[*Therefore the function $\mathcal{W}_b(\mathcal{V}_b(x,y),z)$ is calculated as a linear interpolation between the vertices of the triangle which contains the point $(\mathcal{V}_b(x,y),z)$.*]

STEP 3. END CONSTRUCTION IV.



Triangular tessellations grid used to construct the function $\mathcal{V}_b(x,y)$. Note that there are two types of triangles formed using this tessellations gird: a lower triangle and an upper triangle (see Figure 4).

Figure 3 Triangular Tessellation Grid for $\mathcal{V}_b(x,y)$

CONSTRUCTION V: Approximating Functions of N Dimensions Using Functions of 2 Dimensions ($N > 3$)

STEP 1. Let $R_0 = \emptyset$.

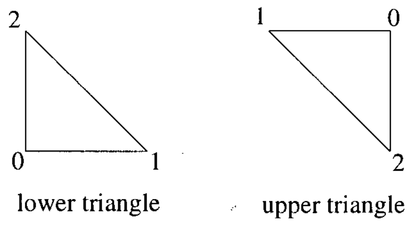
[The first domain R_0 is initialized to be an empty set.]

STEP 2. Let $\rho_1 = 0$.

[The integer ρ_1 is used as a pointer to the domains R_{ρ_1}.]

STEP 3. WHILE it is true that

$$\exists(x_3,\ldots,x_{N-1})\left[\left((x_3,\ldots,x_{N-1}) \notin \bigcup_{i=0}^{\rho_1} R_i\right) \wedge (\exists(x_0,x_1,x_2)[(x_0,\ldots,x_{N-1}) \in \mathcal{D}])\right], \tag{83A}$$



The point $(x, y) \in S_b$ can be located in either a lower triangle or an upper triangle of the tessellations grid defined in Figure 3. For both types of triangles, vertex "0" is located at point $(l_0 \cdot \delta, m_0 \cdot \delta) \in \Re^2$ and the value of the function $\mathtt{V}_b$ at this point is $\mathtt{V}_b(l_0 \cdot \delta, m_0 \cdot \delta)$; vertex "1" is located at point $(l_1 \cdot \delta, m_1 \cdot \delta) \in \Re^2$ and the value of the function $\mathtt{V}_b$ at this point is $\mathtt{V}_b(l_1 \cdot \delta, m_1 \cdot \delta)$; and finally, vertex "2" is located at point $(l_2 \cdot \delta, m_2 \cdot \delta) \in \Re^2$ and the value of the function $\mathtt{V}_b$ at this point is $\mathtt{V}_b(l_2 \cdot \delta, m_2 \cdot \delta)$.

Figure 4 Lower And Upper Triangles of S_b

EXECUTE the following steps:

[Therefore, at the completion of **STEP 3**, *the entire domain of definition of the function being approximated will be spanned, along the axis $(x_3, x_4, \ldots, x_{N-1})$, by the union of all the domains R_i, for all $i \in \{0, 1, \ldots, \rho_1\}$.]*

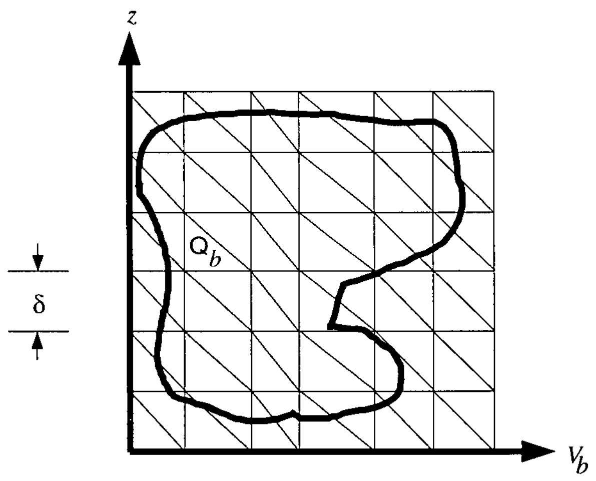
i. Let $\rho_1 = \rho_1 + 1$.

[This initiates the construction of the next domain R_{ρ_1}.]

ii. Choose $R_{\rho_1} \subset \Re^{N-3}$ such that

a. $\forall i \in \{1, \ldots, (\rho_1 - 1)\}[R_i \cap R_{\rho_1} = \emptyset]$,

[Thus the domains R_{ρ_1} are all disjoint, satisfying definition 1.a of THEOREM II.]



Triangular tessellations grid used to construct the function $\mathcal{W}_b(\mathcal{V}_b(x,y),z)$. Note that there are two types of triangles formed using this tessellations gird: a lower triangle and an upper triangle (see Figure 6). Note also that the vertical lines are not regularly spaced: the vertical line spacing is defined by the value of σ in **STEP 6.iii** of CONSTRUCTION I.

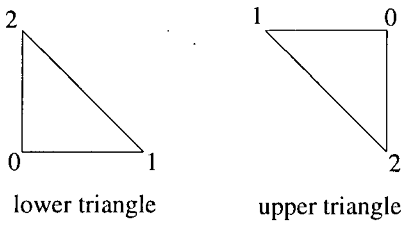
Figure 5 Triangular Tessellation Grid for $\mathcal{W}_b(\mathcal{V}_b(x,y),z)$

b. $\exists(x_0,x_1,x_2)[((x_0,x_1,\ldots,x_{N-1})\in\mathcal{D})\wedge((x_3,x_4,\ldots,x_{N-1})\in R_{\rho_1})]$, and,

[Thus the domains R_{ρ_1} contain points which correspond to the domain of definition of the function being approximated.]

c. The domain R_{ρ_1} is dense.

d. The boundaries of the domains R_{ρ_1} are chosen such that a 3 dimensional approximation, along the axes (x_0,x_1,x_2), of the function $f(x_0,\ldots,x_{N-1})$ within the domain R_{ρ_1}, is at least better than no approximation at all. Or, put formally, this condition is satisfied as follows. Define the 3 dimensional



The point $(\mathcal{V}_b(x,y),z)$ can be located in either a lower triangle or an upper triangle of the tessellations grid defined in Figure 5. For both types of triangles, vertex "0" is located at point $(\mathtt{V}_{b_0}, n_0 \cdot \delta)$ and the value of the function $\mathtt{W}_b$ at this point is $\mathtt{W}_b(\mathtt{V}_{b_0}, n_0 \cdot \delta)$; vertex "1" is located at point $(\mathtt{V}_{b_1}, n_1 \cdot \delta)$ and the value of the function $\mathtt{W}_b$ at this point is $\mathtt{W}_b(\mathtt{V}_{b_1}, n_1 \cdot \delta)$; and finally, vertex "2" is located at point $(\mathtt{V}_{b_2}, n_2 \cdot \delta)$ and the value of the function $\mathtt{W}_b$ at this point is $\mathtt{W}_b(\mathtt{V}_{b_2}, n_2 \cdot \delta)$.

Figure 6 Lower And Upper Triangles of $(\mathcal{V}_b(x,y),z)$

function $\mathcal{P}_{\rho_1}(x_0,x_1,x_2)$, defined on the domain

$$U_{\rho_1} = \{(x_0,x_1,x_2)|((x_0,x_1,\dots,x_{N-1}) \in \mathcal{D}) \wedge ((x_3,\dots,x_{N-1}) \in R_{\rho_1})\}, \tag{84A}$$

to be

$$\mathcal{P}_{\rho_1}(x_0,x_1,x_2) = \frac{\int\limits_{R_{\rho_1}} f(x_0,x_1,\dots,x_{N-1})dR_{\rho_1}}{\int\limits_{R_{\rho_1}} dR_{\rho_1}}. \tag{85A}$$

Let $\beta \in \Re$ such that $0 < \beta < 1$, and let $\epsilon_\beta \in \Re$ such that

$$0 < \epsilon_\beta < \beta \cdot \left(\max_{(x_3,\dots,x_{N-1}) \in T_{\rho_1}} |f(x_0,\dots,x_{N-1})| \right), \tag{86A}$$

where the domain T_{ρ_1} is defined as

$$T_{\rho_1} = \{(x_0,x_1,\dots,x_{N-1})|((x_0,x_1,\dots,x_{N-1}) \in \mathcal{D}) \wedge ((x_3,x_4,\dots,x_{N-1}) \in R_{\rho_1})\}. \tag{87A}$$

Then, the domains R_{ρ_1} must be chosen such that

$$\max_{T_{\rho_1}} |f(x_0,\ldots,x_{N-1}) - \mathcal{P}_{\rho_1}(x_0,x_1,x_2)| + \epsilon_\beta \leq \beta \cdot \left(\max_{T_{\rho_1}} |f(x_0,\ldots,x_{N-1})|\right). \tag{88A}$$

[In LEMMA I it is shown that such a domain R_{ρ_1} must exist. The proof of LEMMA I is based on the construction of a domain R_{ρ_1} which satisfies all of the above conditions.]

iii. Let $\mathcal{U}(x,y,z) \equiv \mathcal{P}_{\rho_1}(x_0,x_1,x_2)$ and

$$\begin{aligned} U = \big\{(x,y,z)|((x_0,\ldots,x_{N-1}) \in \mathcal{D}) \wedge ((x_3,\ldots,x_{N-1}) \in R_{\rho_1})\wedge \\ (x=x_0) \wedge (y=x_1) \wedge (z=x_2)\big\}. \end{aligned} \tag{89A}$$

Use CONSTRUCTION I, CONSTRUCTION II, CONSTRUCTION III and CONSTRUCTION IV to construct the functions $\mathcal{V}_b(x,y)$ and $\mathcal{W}_b(\mathcal{V}_b,z)$, and the domain S_b ($\forall b \in \{1,2,\ldots,b_{\max}\}$). Note that the ϵ used in CONSTRUCTION I must be chosen to satisfy equation (86A) of this CONSTRUCTION; i.e. $\epsilon < \epsilon_\beta$. For all $b \in \{1,2,\ldots,b_{\max}\}$, let $h_{\rho_1 b}(x_0,x_1) \equiv \mathcal{V}_b(x,y)$, $g_{\rho_1 b}(h_{\rho_1 b},x_2) \equiv \mathcal{W}_b(\mathcal{V}_b,z)$, and $D_{\rho_1 b} \equiv S_b$. Also, $\forall b \in \{(b_{\max}+1),(b_{\max}+2),\ldots\}$ let $D_{\rho_1 b} = \emptyset$.

[Thus the domains $D_{\rho_1 b}$ and the functions $h_{\rho_1 b}(x_0,x_1)$ satisfy definition 2 of THEOREM II, and the functions $g_{\rho_1 b}(h_{\rho_1 b},x_2)$ satisfy definition 3 of THEOREM II (for all $b \in \{1,2,\ldots,b_{\max}\}$).]

iv. For all $\sigma_1 \in \{1,2,\ldots,b_{\max}\}$ define the N dimensional function $f_{\rho_1\sigma_1}(x_0,\ldots,\dot{x}_{N-1})$, $\forall(x_0,\ldots,x_{N-1}) \in \mathrm{Dom}_{\rho_1\sigma_1}$ where

$$\begin{aligned} \mathrm{Dom}_{\rho_1\sigma_1} = \{(x_0,\ldots,x_{N-1})|((x_0,\ldots,x_{N-1}) \in \mathcal{D}) \wedge ((x_3,x_4,\ldots,x_{N-1}) \in R_{\rho_1})\wedge \\ ((x_0,x_1) \in D_{\rho_1\sigma_1}) \wedge ((x_0,x_1,x_2) \in U_{\rho_1})\}, \end{aligned} \tag{90A}$$

as

$$f_{\rho_1\sigma_1}(x_0,\ldots,x_{N-1}) = f(x_0,\ldots,x_{N-1}) - g_{\rho_1\sigma_1}(h_{\rho_1\sigma_1}(x_0,x_1),x_2). \tag{91A}$$

[Thus, the function $f_{\rho_1\sigma_1}(x_0,\ldots,x_{N-1})$ represents the approximation error along branch $\rho_1\sigma_1$ for the case $l_m = 1$ in equation (28A) of THEOREM II. Note that if

$$\max_{\forall(x_0,\ldots,x_{N-1})\in \mathrm{Dom}_{\rho_1\sigma_1}} |f_{\rho_1\sigma_1}(x_0,\ldots,x_{N-1})| < \epsilon_{max}, \tag{92A}$$

then the approximation along branch $\rho_1\sigma_1$ is complete and the branch need not be extended.]

STEP 4. $\forall i \in \{(\rho_1+1),(\rho_1+2),\ldots\}$, let $R_i = \emptyset$.

[This satisfies definition 1.b of THEOREM II.]

STEP 5. Let $l = 2$.

[This initiates the construction of level $l = 2$ of the approximation tree.]

STEP 6. Let $i_1, i_2, \ldots, i_{N-2} \in \{0,1,\ldots,N-1\}$ and $j_1, j_2 \in \{0,1,\ldots,N-1\}$ such that,

a. $j_1 = (l \bmod (N-1))$, and $j_2 = ((l+1) \bmod (N-1))$;

b. $\forall m,n \in \{1,2,\ldots,N-2\}\big[(m \neq n \Rightarrow i_m \neq i_n) \wedge (m > n \Rightarrow i_m > i_n)\big]$; and,

c. $\forall m \in \{1,2,\ldots,N-2\} \wedge \forall n \in \{1,2\}[i_m \neq j_n]$.

STEP 7. For all $\rho_1,\sigma_1,\ldots,\rho_{l-1},\sigma_{l-1} \in \{1,2,3,\ldots\}$ and (note that for the case $l = 2$, $\rho_1,\sigma_1,\ldots,\rho_{l-1}$ is read as ρ_1, and $\rho_1,\sigma_1,\ldots,\rho_{l-1},\sigma_{l-1}$ is read as ρ_1,σ_1) $\forall(x_0,\ldots,x_{N-1}) \in$ tree branch $\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}$, such that a.

$$\max_{(x_0,\ldots,x_{N-1})\in \mathrm{Dom}_{\rho_1\sigma_1\cdots\rho_{l-1}\sigma_{l-1}}} \left|f_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}}(x_0,\ldots,x_{N-1})\right| \geq \epsilon_{max}, \tag{93A}$$

where the domain $\mathrm{Dom}_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}}$ is defined in **STEP 3.iv** and **STEP 7.iii.4** of this CONSTRUCTION, and,

b. $R_{\rho_1\sigma_1\ldots\rho_{l-1}} \neq \emptyset$ and $D_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}} \neq \emptyset$,

EXECUTE the following steps:

[The goal of the following steps is to continue the construction of the tree until each branch has an approximation error which is less than the maximum allowable error. The steps executed are similar in function to the first 4 steps of this CONSTRUCTION, with the exception that the function being approximated at branch $\rho_1, \sigma_1, \ldots, \rho_{l-1}, \sigma_{l-1}$ is denoted by $f_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}}(x_0,\ldots,x_{N-1})$.]

i. Let $R_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}0} = \emptyset$.

[The first domain $R_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}0} = \emptyset$ is initialized to be an empty set.]

ii. Let $\rho_l = 0$.

[The integer ρ_l is used as a pointer to the domains $R_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}\rho_l}$.]

iii. WHILE it is true that,

$$\exists(x_{i_1},\ldots,x_{i_{N-2}})\Bigg[\left((x_{i_1},\ldots,x_{i_{N-2}}) \notin \bigcup_{i=0}^{\rho_l} R_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}i}\right)\wedge \\ \Big(\exists(x_{j_1},x_{j_2})\big[(x_0,\ldots,x_{N-1})\in\mathcal{D}\wedge \\ (x_0,\ldots,x_{N-1}) \in_{\text{tree branch}} \rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}\big]\Big)\Bigg], \tag{94A}$$

EXECUTE the following steps:

[Therefore, at the completion of **STEP 7.iii**, *the entire domain of definition of the function $f_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}}(x_0,\ldots,x_{N-1})$ will be spanned, along the axis $(x_{i_1},x_{i_2},\ldots,x_{i_{N-2}})$, by the union of all the domains $R_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}i}$, for all $i\in\{0,1,\ldots,\rho_l\}$. Note that the goal of this step is similar to goal of* **STEP 3**.*]*

1. Let $\rho_l = \rho_l + 1$.

   [This initiates the construction of the next domain $R_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}\rho_l}$.]

2. Choose $R_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}\rho_l}\subset\Re^{N-2}$ such that

   a. $\forall i\in\{1,\ldots,(\rho_l-1)\}\left[R_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}i}\cap R_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}\rho_l}=\emptyset\right]$,

   [Thus the domains $R_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}\rho_l}$ are all disjoint, satisfying definition 4.a of THEOREM II.]

   b. $\exists(x_{j_1},x_{j_2})[((x_1,\ldots,x_N)\in\mathcal{D})\wedge$

   $((x_0,\ldots,x_{N-1})\in$ tree branch $\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1})\wedge$

   $((x_{i_1},\ldots,x_{i_{N-2}})\in R_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}\rho_l})]$

   and,

   [Thus the domains $R_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}\rho_l}$ contain points which correspond to the domain of definition of the function being approximated.]

   c. The domain $R_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}\rho_l}$ is dense.

   d. The boundaries of the domains $R_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}\rho_l}$ are chosen such that a 3 dimensional approximation, along the axes $\big(h_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}}, x_{j_1}, x_{j_2}\big)$, of the function $f_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}}(x_0,\ldots,x_{N-1})$ within a domain $R_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}\rho_l}$, is at least better than no approximation at all. Or, put formally, this condition is satisfied as follows. Define the 3 dimensional function

$$\mathcal{P}_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}\rho_l}\big(h_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}}, x_{j_1}, x_{j_2}\big), \tag{95A}$$

defined on the domain

$$\begin{aligned} U_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}\rho_l} = \big\{\big(h_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}}, x_{j_1}, x_{j_2}\big) \big| ((x_0,\ldots,x_{N-1}) \in \mathcal{D}) \wedge \\ ((x_0,\ldots,x_{N-1}) \in \text{tree branch } \rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}) \wedge \\ \big((x_{i_1},\ldots,x_{i_{N-2}}) \in R_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}\rho_l}\big)\big\}, \end{aligned} \tag{96A}$$

to be

$$\mathcal{P}_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}\rho_l}\big(h_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}}, x_{j_1}, x_{j_2}\big) = \frac{\int_{\Pi} f_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}}\, d\Pi}{\int_{\Pi} d\Pi} \tag{97A}$$

where $\Pi = T_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}\rho_l} \cap U_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}\rho_l}$ and the domain $T_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}\rho_l}$ is defined as

$$\begin{aligned} T_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}\rho_l} = \Big\{(x_0,\ldots,x_{N-1}) \;\Big|\; ((x_0,\ldots,x_{N-1}) \in \mathcal{D}) \wedge \\ ((x_0,\ldots,x_{N-1}) \in \text{tree branch } \rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}) \wedge \\ \big((x_{i_1},\ldots,x_{i_{N-2}}) \in R_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}\rho_l}\big)\Big\}. \end{aligned} \tag{98A}$$

Let $\beta \in \Re$ such that $0 < \beta < 1$, and let $\epsilon_\beta \in \Re$ such that

$$0 < \epsilon_\beta < \beta \cdot \left(\max_{(x_{i_1},\ldots,x_{i_N}) \in T_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}\rho_l}} \big| f_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}}(x_0,\ldots,x_{N-1}) \big| \right). \tag{99A}$$

Then, the domains $R_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}\rho_l}$ must be chosen such that

$$\max_{T_{\rho_1\sigma_1\cdots\rho_{l-1}\sigma_{l-1}\rho_l}} \Big| f_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}}(x_0,\ldots,x_{N-1}) - \mathcal{P}_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}\rho_l}\big(h_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}}, x_{j_1}, x_{j_2}\big) \Big| + \epsilon_\beta \leq \beta \cdot \left(\max_{T_{\rho_1\sigma_1\cdots\rho_{l-1}\sigma_{l-1}\rho_l}} \big| f_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}}(x_0,\ldots,x_{N-1}) \big| \right). \tag{100A}$$

[In LEMMA II it is shown that such a domain $R_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}\rho_l}$ must exist. The proof of LEMMA II is based on the construction of a domain $R_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}\rho_l}$ which satisfies the all of the above conditions.]

3. Let $\mathcal{U}(x,y,z) \equiv \mathcal{P}_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}\rho_l}\big(h_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}}, x_{j_1}, x_{j_2}\big)$ and

$$U = \big\{(x,y,z) | ((x_0,\ldots,x_{N-1}) \in \mathcal{D}) \wedge ((x_0,\ldots,x_{N-1}) \in \text{tree branch } \rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}) \wedge ((x_{i_1},\ldots,x_{i_{N-2}}) \in R_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}\rho_l}) \wedge (x = h_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}}) \wedge (y = x_{j_1}) \wedge (z = x_{j_2})\big\}. \tag{101A}$$

Use CONSTRUCTION I, CONSTRUCTION II, CONSTRUCTION III and CONSTRUCTION IV to construct the functions $\mathcal{V}_b(x,y)$ and $\mathcal{W}_b(\mathcal{V}_b,z)$, and the domain S_b ($\forall b \in \{1,2,\ldots,b_{\max}\}$). Note that the ϵ used in CONSTRUCTION I must be chosen to satisfy equation (99A) of this CONSTRUCTION i.e. $\epsilon < \epsilon_\beta$. For all $b \in \{1,2,\ldots,b_{\max}\}$, let $h_{\rho_1\sigma_1\ldots\rho_l b}\big(h_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}}, x_{j_1}\big) \equiv \mathcal{V}_b(x,y)$, $g_{\rho_1\sigma_1\ldots\rho_l b}\big(h_{\rho_1\sigma_1\ldots\rho_l b}, x_{j_2}\big) \equiv \mathcal{W}_b(\mathcal{V}_b,z)$, and $D_{\rho_1\sigma_1\ldots\rho_l b} \equiv S_b$. Also, $\forall b \in \{(b_{\max}+1),(b_{\max}+2),\ldots\}$ let $D_{\rho_1\sigma_1\ldots\rho_l b} = \emptyset$.

[Thus the domains $D_{\rho_1\sigma_1\ldots\rho_l b} = \emptyset$ and the functions $h_{\rho_1\sigma_1\ldots\rho_l b}(h_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}}, x_{j_1})$ satisfy definition 5 of THEOREM II, and the functions $g_{\rho_1\sigma_1\ldots\rho_l b}(h_{\rho_1\sigma_1\ldots\rho_l b}, x_{j_2})$ satisfy definition 6 of THEOREM II (for all $b \in \{1, 2, \ldots, b_{\max}\}$).]

4. For all $\sigma_l \in \{1, 2, \ldots, b_{\max}\}$ define the N dimensional function

   $f_{\rho_1\sigma_1\ldots\rho_l\sigma_l}(x_0, \ldots, x_{N-1})$, $\forall(x_0, \ldots, x_{N-1}) \in \mathrm{Dom}_{\rho_1\sigma_1\ldots\rho_l\sigma_l}$ where

$$
\begin{aligned}
\mathrm{Dom}_{\rho_1\sigma_1\ldots\rho_l\sigma_l} = \Big\{&(x_0, \ldots, x_{N-1}) | ((x_0, \ldots, x_{N-1}) \in \mathcal{D}) \wedge \\
&((x_{i_1}, \ldots, x_{i_{N-2}}) \in R_{\rho_1\sigma_1\ldots\rho_l}) \wedge \\
&((x_1, \ldots, x_N) \in \text{tree branch } \rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}) \wedge \\
&((h_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}}, x_{j_1}, x_{j_2}) \in U_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}\rho_l}) \wedge \\
&((h_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}}, x_{j_1}) \in D_{\rho_1\sigma_1\ldots\rho_l\sigma_l})\Big\},
\end{aligned}
\tag{102A}
$$

   as

$$
\begin{aligned}
f_{\rho_1\sigma_1\ldots\rho_l\sigma_l}(x_0, \ldots, x_{N-1}) =& f_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}}(x_0, \ldots, x_{N-1}) - \\
& g_{\rho_1\sigma_1\ldots\rho_l\sigma_l}(h_{\rho_1\sigma_1\ldots\rho_l\sigma_l}, x_{j_2}).
\end{aligned}
\tag{103A}
$$

   [Thus, the function $f_{\rho_1\sigma_1\ldots\rho_l\sigma_l}(x_0, \ldots, x_{N-1})$ represents the approximation error along branch $\rho_1\sigma_1\ldots\rho_l\sigma_l$ for the case $l_m = l$ in equation (28A) of THEOREM II. Note that if $\max_{\forall(x_0,\ldots,x_{N-1})\in\mathrm{Dom}_{\rho_1\sigma_1\ldots\rho_l\sigma_l}} |f_{\rho_1\sigma_1\ldots\rho_l\sigma_l}(x_0, \ldots, x_{N-1})| < \epsilon_{max}$, then the approximation along branch $\rho_1\sigma_1\ldots\rho_l\sigma_l$ is complete and the branch need not be extended.]

iv. $\forall i \in \{(\rho_k + 1), (\rho_k + 2), \ldots\}$, let $R_{\rho_1\sigma_1\ldots\rho_{l-1}\sigma_{l-1}i} = \emptyset$.

[This satisfies definition 4.b of THEOREM II.]

STEP 8. IF $\exists \rho_1, \sigma_1, \ldots, \rho_l, \sigma_l \in \{1, 2, 3, \ldots\}$ such that, $\forall (x_0, \ldots, x_{N-1}) \in$ tree branch $\rho_1\sigma_1 \ldots \rho_l\sigma_l$,

[The goal of this step is to search out branches $\rho_1, \sigma_1, \ldots, \rho_l, \sigma_l$ which still have an approximation error which is greater than the maximum allowable error.]

a.

$$\max_{(x_0,\ldots,x_{N-1})\in \mathrm{Dom}_{\rho_1\sigma_1\ldots\rho_l\sigma_l}} |f_{\rho_1\sigma_1\ldots\rho_l\sigma_l}(x_0, \ldots, x_{N-1})| \geq \epsilon_{max}, \tag{104A}$$

where the domain $\mathrm{Dom}_{\rho_1\sigma_1\ldots\rho_l\sigma_l}$ is defined in **STEP 3.iv** and **STEP 7.iii.4** of this CONSTRUCTION, and,

b. $R_{\rho_1\sigma_1\ldots\rho_l} \neq \emptyset$ and $D_{\rho_1\sigma_1\ldots\rho_l\sigma_l} \neq \emptyset$,

THEN EXECUTE the following steps:

i. Let $l = l + 1$.

[This initiates the construction of level l of the approximation tree.]

ii. GOTO **STEP 6**

ELSE END CONSTRUCTION V.