Detection And Approximation Of Function Of Two Variables In High Dimensions

Minzhe Pan
University of Central Florida

DETECTION AND APPROXIMATION OF FUNCTION OF TWO VARIABLES IN HIGH DIMENSIONS

by

MINZHE PAN
B.S. National University of Defense Technology, China, 2002

A thesis submitted in partial fulfillment of the requirements
for the degree of Master of Science
in the Department of Mathematics
in the College of Science
at the University of Central Florida
Orlando, Florida

Fall Term
2010

Major Professor: XIN LI

ABSTRACT

This thesis originates from the deterministic algorithm of DeVore, Petrova, and Wojtaszcsyk for the detection and approximation of functions of one variable in high dimensions. We propose a deterministic algorithm for the detection and approximation of function of two variables in high dimensions.

I dedicate my dissertation work to my family and many friends. A special feeling of gratitude to my wife, Qiling Shi, for being incredibly understanding and supportive.

ACKNOWLEDGMENTS

I wish to thank all my committee members Dr. Deguang Han and Dr. Qiyu Sun who were more than generous with their expertise and precious time. A special thanks to Dr. Xin Li, my committee chairman and advisor for his patience throughout the entire process. I would like to thank all the people who helped me throughout my life.

TABLE OF CONTENTS

CHAPTER ONE: MOTIVATION

1.1 Background

Many solutions of scientific problems can be reformulated as the approximation of a function f, defined on a domain in $\mathbb{R}^N$, with N large. In many cases, considering all the parameters is either not necessary or not feasible. In the real world applications, we always just consider a few most important parameters and develop the function f depending on only a few parameters. For example, in the insurance industry, the insurance companies collect a lot of data from many different dimensions. Then analyzing this data set will find that the function f depends only on one or two correlations or patterns among many fields. Then we may use this data with very few variables to build a model to calculate the premium for the subscription.

DeVore, Petrova, and Wojtaszczyk recently ([1]) pointed out that many existing classical numerical methods cannot be used for the approximation of function of N variables when N is very large, due to the phenomenon called the curse of dimensionality. They observed that in many cases the function f can be approximated well by a function of very few variables. Based on this observation, they formulated a version of variable reduction in high dimension as follows (and for simplicity of reference, we will refer to this problem as DeVore's reduction problem in this thesis):

DeVore's Variable Reduction Problem: *Assume that f is a continuous function defined on $[0,1]^N$ but it depends on just l of the N coordinates, i.e., $f(x_1, x_2, \ldots, x_N) = g(x_{i_1}, x_{i_2}, \ldots, x_{i_l})$, where $i_1, i_2, \ldots, i_l$ are not known. The question DeVore, Petrova, and Wojtasszczyk ([1])*

proposed is: What is the smallest number of evaluations of f so that we can detect the coordinates $i_1, i_2, \dots, i_l$ ***and*** *recover f based on these evaluations?*

Note that we are free at selecting the points of the evaluations. Indeed, we should be very careful in selecting these points for the function evaluations. In view of sampling theory, we can restate the problem as: Select sampling points to detect the independent variables and recover the function with the minimum number of samples of the function. Assuming that we can do exact evaluation of functions, DeVore, Petrova, and Wojtaszcsyk in [1] gave several algorithms and proved that (c.f., Theorem 4.2 and discussion in the last part of the paper) in an order of $L^l(log_2 N)$ number of sampling, we can detect the variables and recover the function in uniform norm to the accuracy L^{-1} when g is in Lip 1. Their algorithm for $l = 1$ is a constructive and deterministic one while their solution to $l > 1$ is more of a probabilistic nature (i.e., with very high probability, the algorithm will work). See also [8] and [9] for the latest development.

Motivated by the results of DeVore, Petrova, and Wojtaszcsyk, we want to study the case when $l = 2$ and we will propose a deterministic algorithm that can detect and approximate the function within the same order and accuracy as given by the random algorithm (c.f. Theorem 4.2 in [1]) of DeVore, Petrova, and Wojtaszcsyk.

Before we embark our construction of the algorithm, we want to show how the problem can be solved in the case when the function f is a linear function.

Consider a linear function of N variables

$$f(x_1, x_2, \dots, x_N) = a_1 x_1 + a_2 x_2 + \cdots + a_N x_N$$

If we know that f really depends only on l variables $x_{i_1}, x_{i_2}, \dots, x_{i_l}$, then we must have

$$a_{i_1} \neq 0, a_{i_2} \neq 0, \dots, a_{i_l} \neq 0 \quad \text{and}$$

$$a_i = 0 \ \ for\ i \neq i_1, i_2, \dots, i_l$$

Now, DeVore's variable reduction problem becomes: How to sample f so that we can solve for the nonzero coefficients. Suppose that we select the following m sample points:

$$x_{11}, x_{12}, \dots, x_{1N},$$

$$x_{21}, x_{22}, \dots, x_{2N},$$

$$\dots$$

$$x_{m1}, x_{m2}, \dots, x_{mN},$$

and we evaluate f at these sampling points so that we have the values

$$y_1 = f(x_{11}, x_{12}, \dots, x_{1N}) = a_1 x_{11} + a_2 x_{12} + \cdots + a_N x_{1N}$$

$$y_2 = f(x_{21}, x_{22}, \dots, x_{2N}) = a_1 x_{21} + a_2 x_{22} + \cdots + a_N x_{2N}$$

$$\dots$$

$$y_m = f(x_{m1}, x_{m2}, \dots, x_{mN}) = a_1 x_{m1} + a_2 x_{m2} + \cdots + a_N x_{mN}$$

Then, in matrix notation, we have

$$y = Ax$$

where $y = (y_1, y_2, \dots, y_m)^T, A = \left(x_{ij}\right)_{i=1:m; j=1:N}$, and $x = (a_1, a_2, \dots, a_N)^T$. Thus, DeVore's variable reduction problem for linear functions can be re-stated as: If x has only very few nonzero components, say l, then what is the smallest m and what kind of matrix A to choose so that we can solve for x given y? This is indeed the Compressive Sensing or Compressive

Sampling problem currently under intensive investigation by many researchers (see, for example, [2], [3], [5]). Here we will mention one of the most significant results obtained by Candes, Romberg, and Tao in [2].

When we do compressive sampling, we can view the media files (such as image, video, sound) as the N-dimensional vectors $\begin{pmatrix} \mathrm{x}_1 \\ \mathrm{x}_2 \\ \vdots \\ \mathrm{x}_\mathrm{N} \end{pmatrix}$, where N is a very large number, e.g., $N \sim 10^6$. When most of the x_i's are extremely small, we can ignore them. In this way we will get a sparse vector and this vector depends only on l $(l \ll N)$ coordinate variables $f(\mathrm{x}_1, \mathrm{x}_2, \dots, \mathrm{x}_\mathrm{N}) = g(x_{i_1}, x_{i_2}, \dots, x_{i_l})$. If we know m but do not know the coordinates $i_1, i_2, \dots, i_l$, we can use Candes-Romberg-Tao Theorem to recover them.

1.2 Candes-Romberg-Tao Theorem

We know review one of the fundamental results in compressive sensing and show how it can help us to solve DeVore's variable reduction problem in the case when f is a linear function. In doing so, we will also show that the order estimate in the main result of DeVore et al ([1]) is sharp in terms of its dependence on N.

Candes-Romberg-Tao Theorem: *Assume that* $m \ll N$ *and* A *is an* $m \times N$ *matrix. Let*

$$y = Ax_0 + \varrho$$

be given with $\|\varrho\|_2 \leq \varepsilon$ *and unknown* x_0 *whose support* $T_0 = \{i : x_0(i) \neq 0\}$ *has cardinality no more than* $S \leq m$. *If*

$$\delta_{3S} + 3\delta_{4S} < 2$$

and if

$$x^* = arg\,min_{x \in \mathbb{R}^N} \|x\|_1 \textit{ subject to } \|y - Ax_0\|_2 \leq \varepsilon$$

then

$$\|x^* - x_0\|_2 \leq C_s \varepsilon$$

The statement above used the quantity δ_k referred to as the constant of restrictive isometry defined by Candes, Romberg, and Tao as follows: it is the smallest positive constant δ such that

$$(1 - \delta)\|x\|_2 \leq \|Ax\|_2 \leq (1 + \delta)\|x\|_2$$

for all $x \in R^N$ with support no more than k.

Using the Candes-Romberg-Tao Theorem, we see that we need to select m large enough so that the condition in the theorem ($\delta_{3S} + 3\delta_{4S} < 2$) is satisfied. Then we can recover the nonzero coefficients $a_{i_1}, a_{i_2}, \dots, a_{i_m}$.

Using the probabilistic argument, it is shown (cf. [5]) that the condition in the theorem can be satisfied for randomly selected matrix A (of Gauss distribution) when $m \sim l(log\ N)$. So, in the case when the function is linear, we can recover the independent variables (and recover the function at the same time) by using as little as $l(\log_2 N)$ samples of the function at randomly choosing (Gaussian) points. This order is consistent with the one given for the general case as mentioned above in the paper of DeVore, Petrova, and Wojtaszcsyk [1]. So, we have just verified the sharpness of the order of [1] (at least for its dependence on N).

Note that, again, even in the case of linear functions, the algorithm depends on randomly generated sampling points. Deterministic constructions in this linear case are still under intensive study. For example, Howard, Applebaum, Searle, and Calderbank ([4]) have proposed the use of deterministic compressive sensing measurement matrices comprised of certain families of frequency modulated sinusoids or second-order Reed-Muller sequences. These matrices come with a very fast reconstruction algorithm whose complexity depends only on the number of measurements m and not on the signal length N. See also [6] and [7] for other recent work on the deterministic construction of the measurement matrices. Our main result of this thesis is the construction of a deterministic algorithm for detecting two variables in the general (nonlinear) case.

CHAPTER TWO: THE CASE OF ONE INDEPANDENT VARIBALE

2.1 Sampling Points of DeVore, Petrova, and Wojtaszczyk

We first consider the simplest case in which we just need to detect one variable at each time. Thus, $f(x_1, x_2, \dots, x_N) = g(x_{i_1})$ for some unknown index i_1.

The set of sampling points where we will ask for the values of f is the union of two sets of points. The first set of points consists of the *base points*: $\mathcal{P} := \{P_i := L^{-1}(i, i, i, \dots, i),\ i = 0,1, \dots, L\}$. The second set of points consists of the *padding points* (to be introduced later). The base points are selected so that its projection onto any single variable will give us a set of uniformly distributed points in that variable. The padding points are used to detect the variable that the function f depends on. Padding points are associated to a pair of points: Let $P, P^{'} \in \mathcal{P}$, $n := \lceil \log_2 N \rceil - 1$. Define the point $[P, P^{'}]_k \quad (k \in \{0,1, \dots, n\})$ whose j-th coordinate is $P(j)$ if $b_k(j) = 0$ or it is $P^{'}(j)$ if $b_k(j) = 1$, where $b_k(j)$ is the k-th bit of the binary representation of $j - 1$:

$$j = 1 + \sum_{k=0}^{n} b_k(j) 2^k.$$

We now prove two lemmas on the properties of the padding points.

Lemma 1. *Assume that f depends on only one variable and assume that $f(P) \neq f(P')$. For any $k = 1,2, \dots, n$, $f([P, P']_k)$ is either $f(P)$ or $f(P')$.*

Proof. By the definition of the padding points, we know that $[P, P']_k(j)$ is either $P(j)$ or $P'(j)$ for $j = 1,2, \dots, N$. Assume that f depends on variable x_{i_1}. Then,

$$f([P,P']_k) = f\big([P,P']_k(1),[P,P']_k(2),\ldots,[P,P']_k(N)\big) = g([P,P']_k(i_1))$$

which is either $g(P(i_1))$ or $g(P'(i_1))$, which, in turn, can be written as either $f(P)$ or $f(P')$. To summarize, we have proved that $f([P,P']_k)$ is either $f(P)$ or$f(P')$. QED

Lemma 2. *Assume that f depends on only variable x_{i_1} and assume that $f(P) \neq f(P')$. For each $k = 0,1,\ldots,n$, $f([P,P']_k)$ is $f(P)$ if and only if* $b_k(i_1) = 0$ *and* $f([P,P']_k)$ *is* $f(P')$ *if and only if* $b_k(i_1) = 1$.

Proof. From the proof of Lemma 1, we see that $f([P,P']_k)$ is $f(P)$ if and only if $[P,P']_k(i_1) = P(i_1)$. But $[P,P']_k(i_1) = P(i_1)$ if and only if $b_k(i_1) = 0$. This proves the first half of the statement in this Lemma. The proof of the second half of the Lemma is similar. QED

We will see how Lemma 2 is used to detect i_1 from the values (indeed, the differences in values) of the function f in next Section.

2.2 Algorithm 1.1 of DeVore, Petriova, and Wojtaszczyk

We consider an algorithm given by DeVore, Petrova, and Wojtaszcsyk in [1] for the case when the function f depends on only one coordinate variable. Recall that we can rewrite function f as $f(\mathrm{x}_1,\mathrm{x}_2,\ldots,\mathrm{x}_\mathrm{N}) = g(\mathrm{x}_{\mathrm{i}_1})$ where i_1 is unknown to us. We first evaluate our function f at all the sampling/query points in advance:

The base points set: $\mathcal{P} := \{P_i := \left(\frac{i}{L},\frac{i}{L},\frac{i}{L},\ldots,\frac{i}{L},\frac{i}{L}\right) \mid i = 0,1,2,\ldots,N\}$

The padding points set: $\mathcal{Q} := \{[p_0, p_1]_k, [p_1, p_2]_k, [p_2, p_3]_k, \dots, [p_{L-1}, p_L]_k\}_{k=0,\dots,n}$.

Recall that

$$[p_i, p_{i+1}]_k(j) = \begin{cases} p_i(j), & if\ b_k(j) = 0 \\ p_{i+1}(j), & if\ b_k(j) = 1 \end{cases}$$

where

$$j = 1 + \sum_{k=0}^{n} b_k(j) 2^k$$

and

$$n = \lceil log_2 N \rceil - 1$$

It is clear that the set of base points contains $L + 1$ points. The important property of this set is when we project it into any coordinate axis, we will get a set of $L + 1$ equally spaced points. So, the set of base points is good for the construction of an approximation of f when we know which variable it depends on.

If f is constant on $\mathcal{P}$, then we just use a constant function to approximate f and we do not need to know the independent coordinate i_1 since the constant can be used to constant an approximate to f with error $= O(L^{-1})$ when g is assumed to be of Lip 1. If f is not constant on $\mathcal{P}$, then we will use the padding points to find the coordinate i_1 as follows: Let i be the smallest index such that $f(P_i) \neq f(P_{i+1})$. Now, we examine the values of $f([P_i, P_{i+1}]_k)$: By lemmas 1 and 2, for each k, $f([P_i, P_{i+1}]_k)$ is equal to exactly one of $f(P_i)$ or $f(P_{i+1})$ and, furthermore, it equals $f(P_i)$ is and only if $b_k(i_1) = 0$; it equals $f(P_{i+1})$ is and only if $b_k(i_1) = 1$. Thus, we will have the values of the follow finite sequence (by checking if $f([P_i, P_{i+1}]_k) = f(P_i)$ or not):

$$b_0(i_1), b_1(i_1), \dots\ , b_n(i_1)$$

Note that

$$i_1 = 1 + \sum_{k=0}^{n} b_k(i_1)2^k \ .$$

So, we can recover the bits of i_1 in its binary representation. Hence we have found the index of the independent variable the function f depends on.

Next, we use a numerical example to demonstrate the definitions and algorithm described above.

Example 1. (Padding Points: N=4)

For N=4 and take L=3.

Consider two base points: $p = \left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)$ and $p' = (\frac{2}{3}, \frac{2}{3}, \frac{2}{3}, \frac{2}{3})$

We can write down all binary representations of 1,2,3, and 4 (=N):

		Binary Representation	
$j = 1$:	$j - 1 = 0$	0	0
$j = 2$:	$j - 1 = 1$	0	1
$j = 3$:	$j - 1 = 2$	1	0
$j = 4$:	$j - 1 = 3$	1	1

For k=1, we have the corresponding column consisting of the digits sequence 0011

For k=0, we have the corresponding column consisting of the digits sequence 0101

So, the padding points are:

$$[p, p']_1 = \left(\frac{1}{3}, \frac{1}{3}, \frac{2}{3}, \frac{2}{3}\right)$$

$$[p, p']_0 = \left(\frac{1}{3}, \frac{2}{3}, \frac{1}{3}, \frac{2}{3}\right)$$

Example 2. (Padding Points, N=8)

Consider $N = 8$ and $L = 8$.

Consider two base points : $p = (\frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8})$

$$p' = (\frac{2}{8}, \frac{2}{8}, \frac{2}{8}, \frac{2}{8}, \frac{2}{8}, \frac{2}{8}, \frac{2}{8}, \frac{2}{8})$$

Write down all the binary representations for $j = 1,2, \dots ,8$:

$$j = 1: \quad j - 1 = 0 = \quad 000$$

$$j = 2: \quad j - 1 = 1 = \quad 001$$

$$j = 3: \quad j - 1 = 2 = \quad 010$$

$$j = 4: \quad j - 1 = 3 = \quad 011$$

$$j = 5: \quad j - 1 = 4 = \quad 100$$

$$j = 6: \quad j - 1 = 5 = \quad 101$$

$$j = 7: \quad j - 1 = 6 = \quad 110$$

$$j = 8: \quad j - 1 = 7 = \quad 111$$

So, we obtain the columns:

$$k = 2 \quad 00001111$$

$$k = 1 \quad 00110011$$

$$k = 0 \quad 01010101$$

The padding points for $k = 0,1,2$ are:

$$[p, p']_0 = \left(\frac{1}{8}, \frac{2}{8}, \frac{1}{8}, \frac{2}{8}, \frac{1}{8}, \frac{2}{8}, \frac{1}{8}, \frac{2}{8}\right)$$

$$[p, p']_1 = \left(\frac{1}{8}, \frac{1}{8}, \frac{2}{8}, \frac{2}{8}, \frac{1}{8}, \frac{1}{8}, \frac{2}{8}, \frac{2}{8}\right)$$

$$[p, p']_2 = \left(\frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{2}{8}, \frac{2}{8}, \frac{2}{8}, \frac{2}{8}\right)$$

Example 3. (**Numerical Experiments of the Algorithm**)

Assume $f(x_1, x_2, \dots, x_N) = g(x_3) = \sin(x_3 * 16\pi)$. We will take N=9 in our experiments. So,

$n = \lceil log_2 N \rceil - 1 = 3$. The base points set is: $\mathcal{P} \coloneqq \{P_i = (\frac{i}{L}, \frac{i}{L}, \frac{i}{L}, \dots, \frac{i}{L}, \frac{i}{L}) |\; i = 0,1,2,\dots, L\}$

and there are 4 padding points for each pair $[P_i, P_{i+1}]_k$: Let $a_i = \frac{i}{L},\ a_{i+1} = \frac{i+1}{L}$, then

$[P_i, P_{i+1}]_3 = \{a_i\, a_i\, a_i\, a_i\, a_i\, a_i\, a_i\, a_i\, a_{i+1}\}$

$[P_i, P_{i+1}]_2 = \{a_i\, a_i\, a_i\, a_i\, a_{i+1}\, a_{i+1}\, a_{i+1}\, a_{i+1}\, a_i\}$

$[P_i, P_{i+1}]_1 = \{a_i\, a_i\, a_{i+1}\, a_{i+1}\, a_i\, a_i\, a_{i+1}\, a_{i+1}\, a_i\}$

$[P_i, P_{i+1}]_0 = \{a_i\, a_{i+1}\, a_i\, a_{i+1}\, a_i\, a_{i+1}\, a_i\, a_{i+1}\, a_i\}$

Numerically, we have

a) L=4

The function values at the base points:

L=4	i=0	i=1	i=2	i=3	i=4
$\frac{i}{L}$	0	0.2500	0.5000	0.7500	1.0000
$f(P_i) =$ 1.0e-014 *	0	-0.0490	-0.0980	-0.1470	-0.1959

The function values at the padding points: $f\ = 1.0e - 014\ *$

$f([P_i, P_{i+1}]_k)$	k=3	k=2	k=1	k=0	Independent variable coordinate
i=0	0	0	-0.0490	0	3
i=1	-0.0490	-0.0490	-0.0980	-0.0490	3
i=2	-0.0980	-0.0980	-0.1470	-0.0980	3
i=3	-0.1470	-0.1470	-0.1959	-0.1470	3

The algorithm needs to take 21 function evaluations.

b) L=8

The function values at the base points:

L=4	i=0	i=1	i=2	i=3	i=4	i=5	i=6	i=7	i=8
$\frac{i}{L}$	0	0.1250	0.2500	0.3750	0.5000	0.6250	0.7500	0.8750	1.00
$f(P_i) =$ 1.0e-014 *	0	-0.0245	-0.0490	-0.0735	-0.098	-0.1225	-0.147	-0.1715	-0.1959

The function values at the padding points:

$$f = 1.0e - 014 *$$

$f([P_i, P_{i+1}]_k)$	k=3	k=2	k=1	k=0	Independent variable coordinate
i=0	0	0	-0.0490	0	3
i=1	-0.0245	-0.0245	-0.0490	-0.0245	3
i=2	-0.0490	-0.0490	-0.0735	-0.0490	3
i=3	-0.0735	-0.0735	-0.0980	-0.0735	3
i=4	-0.0980	-0.0980	-0.1225	-0.0980	3
i=5	-0.1225	-0.1225	-0.1470	-0.1225	3
i=6	-0.1470	-0.1470	-0.1715	-0.1470	3
i=7	-0.1715	-0.1715	-0.1959	-0.1715	3

It needs to ask for f's values for 41 times.

c) L=12

The function values at the base points:

	i=0	i=1	i=2	i=3	i=4	i=5	i=6	i=7	i=8	i=9	i=10	i=11	i=12
$\frac{i}{L}$	0	0.0833	0.1667	0.2500	0.3333	0.4167	0.5000	0.5833	0.6667	0.7500	0.8333	0.9167	1.0000
f	0	-0.8660	0.8660	-0.0000	-0.8660	0.8660	-0.0000	-0.8660	0.8660	-0.0000	-0.8660	0.8660	-0.0000

The function values at the padding points:

$f([P_i, P_{i+1}]_k)$	k=3	k=2	k=1	k=0	Independent variable coordinate
i=0	0	0	-0.8660	0	3
i=1	-0.8660	-0.8660	0.8660	-0.8660	3
i=2	0.8660	0.8660	-0.0000	0.8660	3
i=3	-0.0000	-0.0000	-0.8660	-0.0000	3
i=4	-0.8660	-0.8660	0.8660	-0.8660	3
i=5	0.8660	0.8660	-0.0000	0.8660	3
i=6	-0.0000	-0.0000	-0.8660	-0.0000	3
i=7	-0.8660	-0.8660	0.8660	-0.8660	3
i=8	0.8660	0.8660	-0.0000	0.8660	3
i=9	-0.0000	-0.0000	-0.8660	-0.0000	3
i=10	-0.8660	-0.8660	0.8660	-0.8660	3
i=11	0.8660	0.8660	-0.0000	0.8660	3

It needs to ask for f's values for 61 times.

d) L=5。 The function value at the base points:

$$f = 1.0e - 014 \ *$$

L=4	i=0	i=1	i=2	i=3	i=4	i=5
$\frac{i}{L}$	0	0.2000	0.4000	0.6000	0.8000	1.0000
$f(P_i)$	0	-0.5878	0.9511	-0.9511	0.5878	-0.0000

The padding points value:

$$f = 1.0e - 014 \ *$$

$f([P_i, P_{i+1}]_k)$	k=3	k=2	k=1	k=0	Independent variable coordinate
i=1	0	0	-0.5878	0	3
i=2	-0.5878	-0.5878	0.9511	-0.5878	3
i=3	0.9511	0.9511	-0.9511	0.9511	3
i=4	-0.9511	-0.9511	0.5878	-0.9511	3
i=5	0.5878	0.5878	-0.0000	0.5878	3

It needs to ask for the values of f for 26 times.

Finally, let us try to run the program on a function of two variables (and we know it will fail).

e) L=5 and assume $f(x_1, x_2, \dots, x_9) = g(x_3, x_5) = \sin(x_3 * 16\pi) + 3 * \sin(x_5 * 8\pi)$

The function values at the base points:

L=4	i=0	i=1	i=2	i=3	i=4	i=5
$\frac{i}{L}$	0	0.2000	0.4000	0.6000	0.8000	1.0000
$f(P_i)$	0	-3.4410	-0.8123	0.8123	3.4410	-0.0000

The padding points value:

$f([P_i, P_{i+1}]_k)$	k=3	k=2	k=1	k=0	Independent variable coordinate
i=1	0	-2.8532	-0.5878	0	7
i=2	-3.4410	-2.3511	-1.9021	-3.4410	16
i=3	-0.8123	2.7144	-2.7144	-0.8123	16
i=4	0.8123	1.9021	2.3511	0.8123	16
i=5	3.4410	0.5878	2.8532	3.4410	16

After evaluating the function 26 times, we get the independent variables' coordinate. But for different base point pairs, the detected position is changed. This signals the possibility that the function is not of one independent variable.

2.3 Approximation Theorem

Once the variable of the function is detected, we can use any standard approximation method to construct an approximation $\hat{f}$ of the function f from its values at the base points. There are many ways to construct good approximations to a given function. We will assume that there will be enough samples of our functions available to us at the uniformly distributed points and we will also assume that the functions we want to approximate are of Lip 1 class (or whose derivatives of order s are in Lip 1, denoted by $Д^s$). Then as in [1], DeVore, Petrova, and Wojtaszcsyk quoted the following result from approximation theory. Let $h = 1/L$ and let X be the discrete points obtained by uniformly partition the interval [0,1]. Let $A_h(g)$ be the approximation to g from the space $Д^s$ based on the values of g at the uniformly distributed points such that (i) $A_h(g) = g$ if g is constant; (ii) $\|A_h(g)\| \leq C_0 \max_{x\in X} |g(x)|$ for all h. When f depends only on one variable $f(x_1, x_{2,} \dots, x_N) = g(x_j)$, we define $\hat{f} = A_h(g)$.

Theorem (Theorem 2.1 in [1]). *If $f(x_1, x_{2,} \dots, x_N) = g(x_j)$ $with$ $g \in Д^s$, then the function $\hat{f}$ defined above satisfies*

$$\left\|f - \hat{f}\right\|_{c(\Omega)} \leq |g|_{Д^s} h^s$$

This algorithm uses at most $L + 1 + L\lceil log_2 N\rceil$ *evaluations of f.*

Proof. We have

$$f(x_1, x_2, \dots, x_N) - \hat{f}(x_1, x_2, \dots, x_N) = g(x_j) - A_h(g)(x_j)$$

and, for $g \in Д^s$,

$$\|g - A_h(g)\|_{c(\Omega)} \leq Ch^s.$$

Recall that

$$|g|_{Д^s} := sup_h\{h^{-s}\|g - A_h(g)\|_{c(\Omega)}\}$$

So,

$$\left\| f(x_1, x_2, \dots, x_N) - \hat{f}(x_1, x_2, \dots, x_N) \right\| = \left\| g(x_j) - A_h(g)(x_j) \right\| \leq |g|_{Д^s} h^s.$$

QED

We have a similar result for two variables and we will just assume that once we can detect the variables, we use the above procedure to approximate our function (of lower dimension) at the uniformly distributed points with step length of $h = 1/L$ in each dimension.

CHAPTER THREE: THE CASE OF TWO INDEPENDNET VARIABLE

3.1 Analysis on one variable case

Now we consider the case that function f depends on two coordinate variables:

$$f(\mathrm{x}_1, \mathrm{x}_2, \ldots, \mathrm{x}_\mathrm{N}) = g(x_\mathrm{i}, x_\mathrm{j})$$

where i, j are unknown to us.

To motivate the algorithm of detecting the two unknown variables, we now reformulate the algorithm of DeVore, Petrova, and Wojtaszcsyk in a different but equivalent way.

First we write down the unique binary representations for all j's.

a) $N = 4$

$$\begin{array}{cc} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{array}$$

b) $N = 7$

$$\begin{array}{ccc} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{array}$$

c) $N = 15$

0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0

Then we spell out the query points. We will examine the query points in each case. We can see that padding points are constructed by changing about half of the coordinates, in a uniform way. Please refer to the colored blocks to see the pattern. The 0’s and 1’s tell us when to use the first or the second point in forming the coordinates of the padding points from the two given points: 0 means the use of the first point’s coordinate and 1 means the use of the second point’s coordinate.

a) $N = 4, L = 3$

Assume that $f(p) \neq f(p')$ with the base points:

$$p = \left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)$$

$$p' = (\frac{2}{3}, \frac{2}{3}, \frac{2}{3}, \frac{2}{3})$$

The related padding points for us to use in order to detect the independent variable are:

$$[p, p']_1 = \left(\frac{1}{3}, \frac{1}{3}, \frac{2}{3}, \frac{2}{3}\right)$$

$$[p,p^{'}]_0 = \left(\frac{1}{3},\frac{2}{3},\frac{1}{3},\frac{2}{3}\right)$$

The value of f at the first padding point will tell us whether the independent variable is in the first half of the coordinates or the second half; the value of f at the second padding point will inform us exactly which location the independent variable is: 1 or 2 in the case it lies in the first half or 3 or 4 if it lies in the second half.

b) $N = 7, L = 8$

Assume that $f(p) \neq f(p')$ with the base points:

$$p = \left(\frac{1}{8},\frac{1}{8},\frac{1}{8},\frac{1}{8},\frac{1}{8},\frac{1}{8},\frac{1}{8}\right)$$

$$p^{'} = (\frac{2}{8},\frac{2}{8},\frac{2}{8},\frac{2}{8},\frac{2}{8},\frac{2}{8},\frac{2}{8})$$

The related padding points for us to use in order to detect the independent variable are:

$$[p,p^{'}]_2 = \left(\frac{1}{8},\frac{1}{8},\frac{1}{8},\frac{1}{8},\frac{2}{8},\frac{2}{8},\frac{2}{8}\right)$$

$$[p,p^{'}]_1 = \left(\frac{1}{8},\frac{1}{8},\frac{2}{8},\frac{2}{8},\frac{1}{8},\frac{1}{8},\frac{2}{8}\right)$$

$$[p,p^{'}]_0 = \left(\frac{1}{8},\frac{2}{8},\frac{1}{8},\frac{2}{8},\frac{1}{8},\frac{2}{8},\frac{1}{8}\right)$$

The value of f at the first point will tell us whether the independent variable is in the first half of the coordinates or the second half; the value of f at the second padding point will inform us which quarter the independent variable lies; and the value of f at the third padding point tells us exactly which location the independent variable is: 1 or 2 in the case it lies in the first quarter, 3 or 4 if it lies in the second quarter, 5 or 6 if it lies in the third quarter, and, finally, 7 or 8 if it lies in the last quarter. Note that once we know which quarter the independent variable lies, we will not be concerned about the coordinates in all other quarters. In particular, we can allow the padding point be changed in its three quarters locations without interfering our detection of the independent variable.

c) $N = 15, L = 16$

Assume that $f(p) \neq f(p')$ with the base points:

$$p = \left(\frac{1}{16}, \frac{1}{16}, \frac{1}{16}, \frac{1}{16}, \frac{1}{16}, \frac{1}{16}, \frac{1}{16}, \frac{1}{16}, \frac{1}{16}, \frac{1}{16}, \frac{1}{16}, \frac{1}{16}, \frac{1}{16}, \frac{1}{16}, \frac{1}{16}\right)$$

$$p^{'} = (\frac{2}{16}, \frac{2}{16}, \frac{2}{16}, \frac{2}{16}, \frac{2}{16}, \frac{2}{16}, \frac{2}{16}, \frac{2}{16}, \frac{2}{16}, \frac{2}{16}, \frac{2}{16}, \frac{2}{16}, \frac{2}{16}, \frac{2}{16}, \frac{2}{16})$$

The related padding points for us to use in order to detect the independent variable are:

$$[p, p^{'}]_3 = \left(\frac{1}{16}, \frac{1}{16}, \frac{1}{16}, \frac{1}{16}, \frac{1}{16}, \frac{1}{16}, \frac{1}{16}, \frac{1}{16}, \frac{2}{16}, \frac{2}{16}, \frac{2}{16}, \frac{2}{16}, \frac{2}{16}, \frac{2}{16}, \frac{2}{16}\right)$$

$$[p, p^{'}]_2 = \left(\frac{1}{16}, \frac{1}{16}, \frac{1}{16}, \frac{1}{16}, \frac{2}{16}, \frac{2}{16}, \frac{2}{16}, \frac{2}{16}, \frac{1}{16}, \frac{1}{16}, \frac{1}{16}, \frac{1}{16}, \frac{2}{16}, \frac{2}{16}, \frac{2}{16}\right)$$

$$[p, p^{'}]_1 = \left(\frac{1}{16}, \frac{1}{16}, \frac{2}{16}, \frac{2}{16}, \frac{1}{16}, \frac{1}{16}, \frac{2}{16}, \frac{2}{16}, \frac{1}{16}, \frac{1}{16}, \frac{2}{16}, \frac{2}{16}, \frac{1}{16}, \frac{1}{16}, \frac{2}{16}\right)$$

$$[p, p^{'}]_0 = \left(\frac{1}{16}, \frac{2}{16}, \frac{1}{16}, \frac{2}{16}, \frac{1}{16}, \frac{2}{16}, \frac{1}{16}, \frac{2}{16}, \frac{1}{16}, \frac{2}{16}, \frac{1}{16}, \frac{2}{16}, \frac{1}{16}, \frac{2}{16}, \frac{1}{16}\right)$$

Just like in the first two cases discussed above, the value of f at the first padding point will tell us whether the independent variable is in the first half of the coordinates or the second half; the value of f at the second padding point will inform us which quarter the independent variable lies; the value of f at the third padding point tells which one-eighth the independent variable lies, and the value of f at the forth padding point gives us the us exact location. Note that in the last two examples (the cases in b) and c) above), the dimension is not a power of 2 and can be handled in the algorithm by checking if the index is beyond the dimension.

We can summarize the essentiality of finding one change variable algorithm as follows.

1) Choose an integer L and build the base points set $\mathcal{P}$. $\mathcal{P} \coloneqq \{P_i \coloneqq \frac{1}{L}(i, i, i, \dots, i)\}_{i=0}^{L}$
2) Ask for the values at the base points.
3) Build the padding points set recursively:
   a) For each pair of points from the base points set. $P, P' \in \mathcal{P}$
   b) Let $n = \lceil log_2 N \rceil - 1$
   c) Let the 1st to 2^nth components of the padding point $[P, P']_n$ to be the same as the first base point P, and the next $(2^n + 1)$th to 2×2^nth components of the second point P'. Now recursively define the next 2^n components to be the same as the first base point P and defined the next 2^n components to be the same as the coordinate of the second base point P',…,

      until the coordinate exceeds N.
   d) $n = n - 1$ Continue the step c) to construct next padding point.
   e) Stop when $n < 0$.
4) If $f(P) \neq f(P')$, ask for the values for the padding points. For $= \lceil \log_2 N \rceil - 1{:}-1{:}\,0$, check to see if $f([P, P']_k) = f(P)$, then we know which half, which quarter, …, etc., the independent variable lies (i.e., we know the kth bit of j: $b_k(j) = 0$ if $f([P, P']_k) = f(P)$, and we know $b_k(j) = 1$ if the value $f([P, P']_k) = f(P')$).
5) We know all the bits of j and hence we know j.

This algorithm needs $L + 1 + L\lceil log_2 N \rceil$ evaluations of f.

Once the independent variable is detected, the $L + 1$ values of f at the base points can all be used to construct a good approximation. Hence, the total evaluations needed for the detection and approximation is $L + 1 + L\lceil log_2 N \rceil$.

3.2 Two independent variables case

Now we consider the two independent variables case.

$$f(x_1, x_2, \dots, x_\mathrm{N}) = g(x_\mathrm{i}, x_\mathrm{j})$$

We propose the follow algorithm to find out the two independent variables' position.

1) Choose an integer L and build the base points set. $\mathcal{P}$. $\mathcal{P} := \{P_i := \frac{1}{L}(i, i, i, \dots, i)\}_{i=0}^{L}$
2) Evaluate the function at the base points. Choose a pair of base points P, P':

   $P = \left(\frac{i}{L}, \frac{i}{L}, \dots, \frac{i}{L}, \frac{i}{L}\right) \quad P' = \left(\frac{j}{L}, \frac{j}{L}, \dots, \frac{j}{L}, \frac{j}{L}\right)$

   such that if we denote the values of the function f at these base points P, P' by q_1 and q_2 then

   $$f(P) = q_1 \neq q_2 = f(P').$$
3) Build the basic base points set recursively:
   a) For each pair of points from the base points set. $P, P' \in \mathcal{P}$
   b) Let $n = \lceil log_2 L \rceil - 1$

c) Let the 1st to 2^nth components of the padding point $[P, P']_n$ be the same as those of the first base point P, and the next $(2^n + 1)$th to 2×2^nth components be the same as those of the second base point P'. Now alternatively define the next 2^n components to be the same as the first base point P and defined the next 2^n components to be the same as the coordinate of the second base point P',…,

until the coordinate exceeds N.

d) $n = n - 1$ and continue the step c) to construct next basic padding point.

e) Stop when $n < 0$.

4) Initialize the ***considered range*** bounds as $b_{up} = 1$ and $b_{down} = N$.

5) Recall that P and P' are chosen as in step 2). So, we know that $g(P'(1), P'(1)) \neq g(P(1), P(1))$. Assume that

$$g\big(P'(1), P'(1)\big) \neq g\big(P(1), P'(1)\big) \quad g\big(P(1), P(1)\big) \neq g\big(P(1), P'(1)\big).$$

Evaluate the function f at all the padding point $[P, P^{'}]_k = PP_k$

The function's value at the padding point PP_k is denoted by $w_1 = f(PP_k)$. Then there are three cases.

Case 1: $w_1 = q_1,$ Case 2: $w_1 = q_2,$ and Case 3: $w_1 \neq q_1 \; and \; w_1 \neq q_2$.

We need to consider the three cases separately.

For $k = \lceil log_2 N \rceil : -1 : 1$

Case 1: $w_1 = q_1$

In this case we know two independent variables are in the first half of the coordinates in the range $[b_{up}, b_{down}]$.

I.e., we know the kth bits of both indices $b_k(v_1) = 0$ and $b_k(v_2) = 0$.

Then we narrow the considered range. Reset the boundary by letting

$$b_{down} = b_{up} + 2^{k-1} - 1.$$

Reset the padding points to *sub-padding* points: Let the components of padding point whose coordinates are not in the considered range be the same as the components of the first base point P .

Case 2: $w_1 = q_2$

In this case, the two independent variables are in the second half of considered range, i.e., we know the kth bits $b_k(v_1) = 1$ and $b_k(v_2) = 1$.

Then we narrow the considered range. Reset the boundary by letting

$$b_{up} = b_{up} + 2^k - 1$$

Reset the basic padding points to sub-padding points as follows: Let the components of each padding point whose coordinates are not in the considered range be the same the components of the first point P .

Case 3: $w_1 \neq q_1 \; and \; w_1 \neq q_2$

In this case the two independent variables are separate into two halves of the considered range. In this case, in each half, we use one variable algorithm to find the single variable in that half.

Step 1: Detect the variable position in the first half of consider range.

We know the kth bit $b_k(v_1) = 0$.

Then we narrow the considered range: Reset the boundary by letting

$$bb_{down} = b_{up} + 2^k - 1, \ \ bb_{up} = b_{up}$$

Copy the remaining padding points set to set A. Reset the padding points in A to sub-padding points as follows: Let the components of each padding point in A whose coordinates are not in the considered range $[bb_{up}, bb_{down}]$ be the same as those of the first point P.

Use the one independent variable algorithm and padding points set A, we can find the independent variable in the range $[bb_{up}, bb_{down}]$.

Step 2: Detect the variable position in the second half of consider range.

We know the kth bit $b_k(v_2) = 1$.

Then we narrow the considered range: Reset the boundary by letting

$$bb_{up} = b_{up} + 2^k - 1, \ \ bb_{down} = b_{down}$$

Copy the padding points set to padding points set B. Reset the padding points in set B to sub-padding point as follows: Let the value of padding point, which coordinate is not in the considered range $[bb_{up}, bb_{down}]$ be the same component as the first point P.

Use the one independent variable algorithm and padding points set B, we can find the independent variable in the range $[bb_{up}, bb_{down}]$.

6) We know all the bits of $b_k(v_1)$ and $b_k(v_2)$ and hence we know the coordinates v_1 and v_2 of two independent variables.

This algorithm uses at most $L + 1 + 2L\lceil log_2 N\rceil$

Once the independent variable is detected, the $L + 1$ values of f at the base points can all be used to construct a good approximation. Hence, the total evaluations needed for the detection and approximation is $L + 1 + 2L\lceil log_2 N\rceil$.

Example 1.

Take N=11.

Choose a pair of base points (a,a,a,a,a,a,a,a,a,a,a) and (b,b,b,b,b,b,b,b,b,b,b) on which the values of f are not the same: f(a,a,a,a,a,a,a,a,a,a,a) $\neq$ f(b,b,b,b,b,b,b,b,b,b,b)

We can write the binary presentation for the number 0 to 10:

$$\begin{array}{cccc}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 \\
0 & 0 & 1 & 1 \\
0 & 1 & 0 & 0 \\
0 & 1 & 0 & 1 \\
0 & 1 & 1 & 0 \\
0 & 1 & 1 & 1 \\
1 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 \\
1 & 0 & 1 & 0
\end{array}$$

Putting them in rows (by taking transpose), we obtain a natural set of partitions of the coordinates (in four different ways):

$$\begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{matrix}$$

Then we get the corresponding four padding points:

$$\begin{matrix} a & a & a & a & a & a & a & a & b & b & b \\ a & a & a & a & b & b & b & b & a & a & a \\ a & a & b & b & a & a & b & b & a & a & b \\ a & b & a & b & a & b & a & b & a & b & a \end{matrix}$$

Assuming independent variables have coordinates at 2 and 3. We want to show how our algorithm can lead us to finding these coordinates.

We consider the first padding point (a,a,a,a,a,a,a,a,b,b,b).

Ask for the first padding point value $f(\mathrm{a}, \mathbf{a}, \mathbf{a}, \mathrm{a}, \mathrm{a}, \mathrm{a}, \mathrm{a}, \mathrm{a}, \mathrm{b}, \mathrm{b}, \mathrm{b})$. We find:

$$f(\mathrm{a}, \mathbf{a}, \mathbf{a}, \mathrm{a}, \mathrm{a}, \mathrm{a}, \mathrm{a}, \mathrm{a}, \mathrm{b}, \mathrm{b}, \mathrm{b}) = f(\mathrm{a}, \mathbf{a}, \mathbf{a}, \mathrm{a}, \mathrm{a}, \mathrm{a}, \mathrm{a}, \mathrm{a}, \mathrm{a}, \mathrm{a}, \mathrm{a})$$

This means the two independent variables are in the same half. We know both variables are in the first half, hence we don't need to consider the second half. We can simply let the second half components to be assigned the same value, a, as the component of first base point. We use 0 to present the first base point component.

We also get $b_3(v_1) = 0, b_3(v_2) = 0$. Now, we reset the padding points to sub-padding points by replacing the coordinates outside the range to take the same value as the first base point and focus on the current range (the first half, colored in red below):

$$\begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \end{matrix}$$

Then we calculate the value of function at the second (sub-)padding point

$$f(\mathrm{a}, \mathbf{a}, \mathbf{a}, \mathrm{a}, \mathrm{b}, \mathrm{b}, \mathrm{b}, \mathrm{b}, \mathrm{a}, \mathrm{a}, \mathrm{a}).$$

We find $f(\mathrm{a}, \mathbf{a}, \mathbf{a}, \mathrm{a}, \mathrm{b}, \mathrm{b}, \mathrm{b}, \mathrm{b}, \mathrm{a}, \mathrm{a}, \mathrm{a}) = f(\mathrm{a}, \mathbf{a}, \mathbf{a}, \mathrm{a}, \mathrm{a}, \mathrm{a}, \mathrm{a}, \mathrm{a}, \mathrm{a}, \mathrm{a}, \mathrm{a})$.

This tells us that the two independent variables are in the first half part of the considered range and we do not need to consider the second half. We will reset the padding points by letting the second half of the considered range components to the same as the first base point component value.

We have $b_2(v_1) = 0,\ b_2(v_2) = 0$. As above, we will focus on the active range (colored red below) and reset the padding points:

$$\begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

We calculate the value of the function at the third (sub-)padding point

$$f(\mathrm{a}, \mathbf{a}, \mathbf{b}, \mathrm{b}, \mathrm{a}, \mathrm{a}, \mathrm{a}, \mathrm{a}, \mathrm{a}, \mathrm{a}, \mathrm{a}).$$

This time, we find

$$f(\mathrm{a}, \mathbf{a}, \mathbf{b}, \mathrm{b}, \mathrm{a}, \mathrm{a}, \mathrm{a}, \mathrm{a}, \mathrm{a}, \mathrm{a}, \mathrm{a}) \neq f(\mathrm{a}, \mathbf{a}, \mathbf{a}, \mathrm{a}, \mathrm{a}, \mathrm{a}, \mathrm{a}, \mathrm{a}, \mathrm{a}, \mathrm{a}, \mathrm{a})$$

$$f(\mathrm{a}, \mathbf{a}, \mathbf{b}, \mathrm{b}, \mathrm{a}, \mathrm{a}, \mathrm{a}, \mathrm{a}, \mathrm{a}, \mathrm{a}, \mathrm{a}) \neq f(\mathrm{b}, \mathbf{b}, \mathbf{b}, \mathrm{b}, \mathrm{b}, \mathrm{b}, \mathrm{b}, \mathrm{b}, \mathrm{b}, \mathrm{b}, \mathrm{b})$$

From this, we know that the two independent variables are separated into two halves of the considered range. One independent variable is in the range [1,2] and another in the other range [3,4]. We have $b_1(v_1) = 0, b_1(v_2) = 1$. Next, we will use the one independent variable algorithm in each half (in two steps) to find their exact position.

Step 1: Finding the independent variable in the first half.

We only consider the first half, so use the first base point component to replace the others part of the remaining components of the padding point(s).

Now the sub-padding points set A corresponds to:

$$\begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

We calculate the value of the function at the (sub-)padding point $f(\mathrm{a}, \mathbf{b}, \mathbf{a}, \mathrm{a}, \mathrm{a}, \mathrm{a}, \mathrm{a}, \mathrm{a}, \mathrm{a}, \mathrm{a}, \mathrm{a})$ and we find

$$f(\mathrm{a}, \mathbf{b}, \mathbf{a}, \mathrm{a}, \mathrm{a}, \mathrm{a}, \mathrm{a}, \mathrm{a}, \mathrm{a}, \mathrm{a}, \mathrm{a}) \neq f(\mathrm{a}, \mathbf{a}, \mathbf{a}, \mathrm{a}, \mathrm{a}, \mathrm{a}, \mathrm{a}, \mathrm{a}, \mathrm{a}, \mathrm{a}, \mathrm{a})$$

$$f(\mathrm{a}, \mathbf{b}, \mathbf{a}, \mathrm{a}, \mathrm{a}, \mathrm{a}, \mathrm{a}, \mathrm{a}, \mathrm{a}, \mathrm{a}, \mathrm{a}) \neq f(\mathrm{b}, \mathbf{b}, \mathbf{b}, \mathrm{b}, \mathrm{b}, \mathrm{b}, \mathrm{b}, \mathrm{b}, \mathrm{b}, \mathrm{b}, \mathrm{b})$$

So, we get $b_0(v_1) = 1$

Hence the first independent variable coordinate is $v_1 = 1 + b_0(v_1) + b_1(v_1)2 + b_2(v_1)2^2 + b_3(v_1)2^3 = 1 + 1 + 0 * 2 + 0 * 2^2 + 0 * 2^3 = 2$.

Step 2: Finding the second independent variable:

We only consider the second part, so use the first base point component to replace the others part of the remaining components of the padding points.

Now the sub-padding points set A is:

$$\begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

We calculate the padding point $f(\mathrm{a}, \mathbf{a}, \mathbf{a}, \mathrm{b}, \mathrm{a}, \mathrm{a}, \mathrm{a}, \mathrm{a}, \mathrm{a}, \mathrm{a}, \mathrm{a})$ and we find

$$f(\mathrm{a}, \mathbf{a}, \mathbf{a}, \mathrm{b}, \mathrm{a}, \mathrm{a}, \mathrm{a}, \mathrm{a}, \mathrm{a}, \mathrm{a}, \mathrm{a}) = f(\mathrm{a}, \mathbf{a}, \mathbf{a}, \mathrm{a}, \mathrm{a}, \mathrm{a}, \mathrm{a}, \mathrm{a}, \mathrm{a}, \mathrm{a}, \mathrm{a})$$

We get $b_0(v_2) = 0$

Hence the second independent variable coordinate is $v_2 = 1 + b_0(v_2) + b_1(v_2)2 + b_2(v_2)2^2 + b_3(v_2)2^3 = 1 + 0 + 1 * 2 + 0 * 2^2 + 0 * 2^3 = 3.$

Example 2

Take N=9 and assume $f(\mathrm{x}_1, \mathrm{x}_2, \dots, \mathrm{x}_\mathrm{N}) = \mathrm{g}(\mathrm{x}_3, \mathrm{x}_5) = \sin(\mathrm{x}_3 * 16\pi) + 3 * \sin(\mathrm{x}_5 * 8\pi)$

a) L=4

The base points value:

1.0e-014 *

L=4	i=0	i=1	i=2	i=3	i=4
$\frac{i}{L}$	0	0.2500	0.5000	0.7500	1.0000
Base points value (1.0e-014 *)	0	-0.1225	-0.2449	-0.3674	-0.4899

The padding points value:

f =1.0e-014 *

	K=3	K=2	K=1	K=0	Independent variable position
i=0	0	-0.0735	-0.0490	0	3
			0	0	5
i=1	-0.1225	-0.1959	-0.1715	-0.1225	3

			-0.1225	-0.1225	5
i=2	-0.2449	-0.3184	-0.2939	-0.2449	3
			-0.2449	-0.2449	5
i=3	-0.3674	-0.4409	-0.4164	-0.3674	3
			-0.3674	-0.3674	5

The algorithm needs to ask value for 37 times.

b)L=8

The base points value:

L=4	i=0	i=1	i=2	i=3	i=4	i=5	i=6	i=7	i=8
$\frac{i}{L}$	0	0.1250	0.2500	0.3750	0.5000	0.6250	0.7500	0.8750	1.00
Base points value 1.0e-014 *	0	0.0122	-0.1225	0.0367	-0.2449	0.0612	-0.3674	0.0857	-0.4899

The padding points value:

f =1.0e-014 *

	K=3	K=2	K=1	K=0	Independent variable position
i=0	0	0.0367	-0.0245	0	3
			0	0	5
i=1	0.0122	-0.0980	-0.0122	0.0122	3
			0.0122	0.0122	5
i=2	-0.1225	0.0612	-0.1470	-0.1225	3
			-0.1225	-0.1225	5
i=3	0.0367	-0.2204	0.0122	0.0367	3
			0.0367	0.0367	5
i=4	-0.2449	0.0857	-0.2694	-0.2449	3
			-0.2449	-0.2449	5
i=5	0.0612	-0.3429	0.0367	0.0612	3
			0.0612	0.0612	5
i=6	-0.3674	0.1102	-0.3919	-0.3674	3
			-0.3674	-0.3674	5
i=7	0.0857	-0.4654	0.0612	0.0857	3
			0.0857	0.0857	5

It needs to ask for *f*'s value for 73 times.

c) L=12

The base points value:

	i=0	i=1	i=2	i=3	i=4	i=5	i=6	i=7	i=8	i=9	i=10	i=11	i=12
$\frac{i}{L}$	0	0.0833	0.1667	0.2500	0.3333	0.4167	0.5000	0.5833	0.6667	0.7500	0.8333	0.9167	1.0000
	0	1.7321	-1.7321	-0.0000	1.7321	-1.7321	-0.0000	1.7321	-1.7321	-0.0000	1.7321	-1.7321	-0.0000

The padding points value:

	K=3	K=2	K=1	K=0	Change variable position
i=0	0	2.5981	-0.8660	0	3
			0	0	5
i=1	1.7321	-3.4641	3.4641	1.7321	3
			1.7321	1.7321	5
i=2	-1.7321	0.8660	-2.5981	-1.7321	3
			-1.7321	-1.7321	5
i=3	-0.0000	2.5981	-0.8660	-0.0000	3
			-0.0000	-0.0000	5
i=4	1.7321	-3.4641	3.4641	1.7321	3
			1.7321	1.7321	5
i=5	-1.7321	0.8660	-2.5981	-1.7321	3
			-1.7321	-1.7321	5
i=6	-0.0000	2.5981	-0.8660	-0.0000	3
			-0.0000	-0.0000	5
i=7	1.7321	-3.4641	3.4641	1.7321	3
			1.7321	1.7321	5
i=8	-1.7321	0.8660	-2.5981	-1.7321	3
			-1.7321	-1.7321	5
i=9	-0.0000	2.5981	-0.8660	-0.0000	3
			-0.0000	-0.0000	5
i=10	1.7321	-3.4641	3.4641	1.7321	3
			1.7321	1.7321	5
i=11	-1.7321	0.8660	-2.5981	-1.7321	3
			-1.7321	-1.7321	5

It need ask for *f*'s value for 109 times.

3.3 Conclusions and Future Research

We have described a deterministic algorithm for the detection of two independent variables of a function given in a high dimensional space based on the algorithm of DeVore, Petrova, and Wojtaszcsyk for the case of the detection of one variable. Our numerical experiments show that our algorithm can detect the two independent variables very fast. We have focused on the detection step and rely on the standard approximation procedure to recover the function once we find the two independent variables. The complexity of our algorithm is in the order of L+1+[log N] evaluations and 2*[log N] comparisons.

We assume that all evaluations are exact and we have used the fact that there is at least one i such that $g(i/L, j/L)$ is different from both $g(i/L, i/L)$ and $g(j/L, j/L)$ for some large value of L. The fact we have used can be verified if we know that the function g is monotonic in both variables locally.

One immediate future research problem is to extend the algorithm to larger numbers of independent variables. Another problem is to address sufficient condition for the local monotonicity in higher dimensional space. Finally, we mention the problem of modifying our algorithm for inexact evaluation and to the sampling and recovery the functions that can be approximated by functions of few variables.

APPENDIX A: ALGORITHM 2.1

```
function fff=findchange1(N,NN,fj)

% ----------------------------------------

%      N is L

%      NN is the number of variables

%      fj is the coorinate point where the change variable

%      a1,a2 are the base point

%------------------------------------------

iV=NN;

a=N;

fj=fj;

iN=0;

%b is the ceil function of log2(a)

b=ceil(log2(NN));

% the base points set

% f1 is the value when base point value equals to a1.  f(a1,a1,a1,...,a1)

% f2 is the value when base point value equals to a2.  f(a2,a2,a2,...,a2)

basepoints(1)=0;

for i=1:N

    basepoints(i+1)=i/N;

end


for i=1:N+1

   basepointvalue(i)=fvalue(basepoints(i));
   iN=iN+1;
end
for iii=1:N
a1=basepoints(iii);
a2=basepoints(iii+1);
f1=fvalue(a1);
f2=fvalue(a2);
% the follow matrix is used to create the padding points set.
% when c(i,j)==0 then the padding point value is a1, otherwise c(i,j)=1 the
% padding point value is a2.
 for i=1:iV
    d=i-1;
    for j=1:b
       e=mod(d,2);
       c(j,i)=e;
       d=(d-e)/2;
    end
 end
%fj is the position of the change variable.
% we use z to save the change variable's position.

 for j=b:-1:1
   if (c(j,fj)==0)
```

```
    aa=a1;
  else
    aa=a2;
  end
    f(iii,b-j+1)=fvalue(aa);
    iN=iN+1;
  if (f(iii,b-j+1)==f1)
    z(b+1-j)=0;
  else
    z(b+1-j)=1;
  end
 end
 % caculate the output value (the position of change value)
 ccc= 0;
 for j=1:b
    ccc=ccc*2+z(j);
 end
 fff=ccc+1;
 ff(iii)=fff;
end
 basepoints
 basepointvalue
 f
 iN

 ff

 c

end


function yy=fvalue(xx)

  yy=sin (xx*16*pi);

end
```

APPENDIX B: ALGORITHM 3.1

```
function [fff,lll]=findchange2(N,NN,fj11,fj22)

% ----------------------------------------

%          N is L

%      NN is the number of variables

%      fj1,fj2 is the coorinate where the change variable

%      a1,a2 are the base points

%-----------------------------------------

iV=NN;

a=N;

fj1=fj11;

fj2=fj22;

iN=0;

% b is the number

b=ceil(log2(iV));

% the base points set

% f1 is the value when base point value equals to a1.  f(a1,a1,a1,...,a1)

% f2 is the value when base point value equals to a2.  f(a2,a2,a2,...,a2)

basepoints(1)=0;


for i=1:N
```

```
    basepoints(i+1)=i/N;
end
for i=1:N+1
    basepointvalue(i)=fvalue2(basepoints(i),basepoints(i));
    iN=iN+1;
end
for iii=1:N
a1=basepoints(iii);
a2=basepoints(iii+1);
f1=fvalue2(a1,a1);
f2=fvalue2(a2,a2);
  iN=iN+2;
% the follow matrix is used to create the padding points set.
% when c(i,j)==0 then the padding point value is a1, otherwise c(i,j)=1 the
% padding point value is a2.
 for i=1:iV
     d=i-1;
     for j=1:b
        e=mod(d,2);
        c(j,i)=e;
        d=(d-e)/2;
    end
  end
%use bb1 and bb2 as the bound of the change variables.
% we use z,l to save the change variable's position.
  bb1=1;
  bb2=iV;
 for j=b:-1:1
   % If the variable aa1 aa2 are not in (bb1,bb2),we set it to a1
   if (c(j,fj1)==1)&&(bb1<=fj1)&&(fj1<=bb2)
     aa1=a2;
   else
     aa1=a1;
   end
   if (c(j,fj2)==1)&&(bb1<=fj2)&&(fj2<=bb2)
     aa2=a2;
   else
     aa2=a1;
   end
   f=fvalue2(aa1,aa2);
   ff1(iii,b-j+1)=f;
   ff2(iii,b-j+1)=f;
     iN=iN+1;
   if (f==f1)
    % when f=f1 this means the two varible are in the up area.
     if (bb2>bb1+2^(j-1)-1)&&(bb2-bb1>2)
       bb2 = bb1+2^(j-1)-1;
     end
     z(b+1-j)=0;
     l(b+1-j)=0;
   elseif (f==f2)
     % when f=f2 this means the two variable are in the down area.
       bb1 = bb1+2^(j-1);
       z(b+1-j)=1;
       l(b+1-j)=1;
   else
     % In this case, one of the variable is in the up area and the other one is in the down area.
     % we set one varible to a1 and use the find one variable method to
     % find it.
       if (j==1)
         z(b+1-j)=0;
         l(b+1-j)=1;
       else
```

```
bb3=bb1+2^(j-1)-1;

bb4=bb1+2^(j-1);

z(b+1-j)=0;

l(b+1-j)=1;

j1=j-1;

% the up area

for j=j1:-1:1

  if (c(j,fj1)==1)&&(bb1<=fj1)&&(fj1<=bb3)

    aa1=a2;

  else

    aa1=a1;

  end

  if (c(j,fj2)==1)&&(bb1<=fj2)&&(fj2<=bb3)

    aa2=a2;

  else

    aa2=a1;

  end

  f=fvalue2(aa1,aa2);

    ff1(iii,b-j+1)=f;

    iN=iN+1;

  if (f==f1)
```

```
      z(b+1-j)=0;

    else

      z(b+1-j)=1;

    end

  end

% the down area

  for j=j1:-1:1

    if (c(j,fj1)==1)&&(bb4<=fj1)&&(fj1<=bb2)

      aa1=a2;

    else

      aa1=a1;

    end

    if (c(j,fj2)==1)&&(bb4<=fj2)&&(fj2<=bb2)

      aa2=a2;

    else

      aa2=a1;

    end

    f=fvalue2(aa1,aa2);

     ff2(iii,b-j+1)=f;

    iN=iN+1;

    if (f==f1)
```

```
            l(b+1-j)=0;
          else
            l(b+1-j)=1;
          end
        end
        % break the for loop.
        break;
      end   %end  if (j==1)
  end  % end if ((f==f1))
end   % end j=b:-1:1
% caculate the output value (the position of change value)
ccc= 0;
for j=1:b
   ccc=ccc*2+z(j);
end
fff=ccc+1;
fff1(iii)=fff;
ccc= 0;
for j=1:b
   ccc=ccc*2+l(j);
end
```

```
 lll=ccc+1;

 fff2(iii)=lll;

end

 basepoints

 basepointvalue

 ff1

 ff2

 iN

 fff1

 fff2

end


function yy2=fvalue2(x1,x2)

  yy2=sin(x1*16*pi)+3*sin(x2*8*pi);

end
```

LIST OF REFERENCES

[1] Ronald DeVore, Guergana Petrova, Przemyslaw Wojtaszczyk, “Approximation of Functions of Few Variables in High Dimensions”, *Constructive Approximation*, to appear (2010).

[2] E. Candes, J. Romberg and T. Tao, “Robust Uncertainty Principles: Exact Signal Reconstruction from Highly Incomplete Frequency Information”, *IEEE Transactions on Information Theory*, 52(2) pp. 489 - 509, Feb. 2006.

[3] E. Candes and T .Tao, “Near Optimal Signal Recovery From Random Projections: Universal Encoding Strategies?”, *IEEE Trans. on Information Theory*, 52(12) pp. 5406 - 5425, Dec. 2006.

[4] M. Capalbo, O. Reingold, S. Vadhan, A. Wigderson, “Randomness Conductors and Constant-Degree Expansion Beyond the Degree/2 Barrier”, *Proc. of the 34th STOC*, pp. 659-668, 2002.

[5] D. Donoho, “Compressed Sensing”, *IEEE Transactions on Information Theory,* 52(4), pp. 1289 - 1306, April 2006

[6] R.A. DeVore, “Deterministic Constructions of Compressed Sensing Matrices”, Preprint, 2007.

[7] P. Indyk, “Explicit constructions for compressed sensing of sparse signals”, in 19th Symposium on Discrete Algorithms, 2008.

[8] A. Cohen, I. Daubechies, R. DeVore, G. Kerkyacharian, and D. Picard, “Capturing Ridge Functions in High Dimensions from Point Queries”, preprint, 2010.

[9] P. Wojtaszczyk, “Complexity of Approximation of Functions of Few Variables in High Dimensions”, preprint, 2010.