

Approximation of Functions of Few Variables in High Dimensions

Ronald DeVore, Guergana Petrova, Przemyslaw Wojtaszczyk *

February 7, 2010

Abstract

Let f be a continuous function defined on $\Omega := [0, 1]^N$ which depends on only ℓ coordinate variables, $f(x_1, \dots, x_N) = g(x_{i_1}, \dots, x_{i_\ell})$. We assume that we are given m and are allowed to ask for the values of f at m points in Ω . If g is in $\text{Lip}1$ and the coordinates i_1, \dots, i_ℓ are known to us, then by asking for the values of f at $m = L^\ell$ uniformly spaced points, we could recover f to the accuracy $|g|_{\text{Lip}1} L^{-1}$ in the norm of $C(\Omega)$. This paper studies whether we can obtain similar results when the coordinates i_1, \dots, i_ℓ are not known to us. A prototypical result of this paper is that by asking for $C(\ell)L^\ell(\log_2 N)$ adaptively chosen point values of f , we can recover f in the uniform norm to accuracy $|g|_{\text{Lip}1} L^{-1}$ when $g \in \text{Lip}1$. Similar results are proven for more general smoothness conditions on g . Results are also proven under the assumption that f can be approximated to some tolerance ϵ (which is not known) by functions of ℓ variables.

AMS subject classification: 41A, 68W25

Key words: approximation of functions, high dimension, significant variables, sensitivity analysis.

1 Introduction

The numerical solution of many scientific problems can be reformulated as the approximation of a function f , defined on a domain in \mathbb{R}^N , with N large. If one only assumes classical smoothness (such as Lipschitz or Besov regularity) of the underlying function f , then numerical recovery and approximation rates deteriorate severely with the growth of N . This is the so-called *curse of dimensionality*. On the other hand, the functions f that arise as solutions to real world problems are thought to be much better behaved

*This research was supported by the Office of Naval Research Contracts ONR-N00014-08-1-1113, ONR N00014-09-1-0107; the AFOSR Contract FA95500910500; the ARO/DoD Contracts W911NF-05-1-0227 and W911NF-07-1-0185; the NSF Grants DMS-0810869 and DMS 0915231; and the Polish MNiSW grant N201 269335. This publication is based on work supported by Award No. KUS-C1-016-04, made by King Abdullah University of Science and Technology (KAUST).

than a general N -variate function in the sense that they depend on only a few parameters or variables or they can be well approximated by such functions. This has led to a concerted effort to develop a theory and algorithms which approximate such functions well without suffering the effect of the curse of dimensionality. There are many impressive approaches (see [1, 4, 12, 8, 11, 13] as representative) which are being developed in a variety of settings. There is also the active literature in compressed sensing which is based on the model that real world functions are sparsely represented in a suitable basis (see e.g. [2, 6, 3] and the references in these papers).

The present paper will study one particular version of variable reduction in high dimension. Our first results assume that f is a continuous function defined on $\Omega := [0, 1]^N$ but it depends on just ℓ of the coordinate variables: $f(x_1, \dots, x_N) = g(x_{i_1}, \dots, x_{i_\ell})$, where i_1, \dots, i_ℓ are not known to us. We are given a budget m of questions we can ask about f . Each question takes the form: What is the value of f at a point of our choosing? We want then to approximate f from these point values. We are interested in what are the best questions to ask and to what error can we capture f as we allow the number m of questions to increase. We shall measure the error of approximation in the norm

$$\|\cdot\| := \|\cdot\|_{C(\Omega)},$$

where $C(\Omega)$ is the space of continuous functions on Ω and the norm is the supremum norm on Ω . Measuring the error in other norms is also of interest but is not discussed here. The quantitative results we obtain will be made under some smoothness assumption on g in the form of Lipschitz or Besov smoothness.

We consider both the nonadaptive setting, where the m points are set in advance, as well as the adaptive setting, where the questions are allowed to depend on the answers to the prior queries. A second problem we discuss is when f is not a function of ℓ variables but it can be approximated to some tolerance ϵ by such a function. We seek again sets of points where the knowledge of the values of f at these points will allow us to approximate f well.

Let us mention one special case of our theory which is particularly easy to understand. Suppose that g is in $\text{Lip}\alpha$ and the coordinates i_1, \dots, i_ℓ are known to us. Then, by asking for the values of f at $m = L^\ell$ appropriately spaced points, we could recover f to the accuracy $|g|_{\text{Lip}\alpha} L^{-\alpha}$ in the norm of $C(\Omega)$. We show that we can obtain similar estimates even when the coordinates i_1, \dots, i_ℓ are not known to us. However, to achieve this performance we have to ask slightly more questions. For example, we give an adaptive algorithm in §4, see Theorem 4.2, which asks for the values of f at $m = C(\ell)L^\ell(\log_2 N)$ points and from these values gives an approximation with accuracy $|g|_{\text{Lip}\alpha} L^{-\alpha}$. The additional factor $\log_2 N$ is the price our algorithm pays for not knowing the coordinates i_1, \dots, i_ℓ .

The paper is organized as follows. In the next section, we gather a few well-known results about multivariate functions and their approximation which we need later. In §3, we discuss the case when g is a univariate function. Our reason for discussing this case separately from the general case of g being ℓ -variate is that the arguments are particularly simple and transparent. This will give the reader a good anchor for understanding the general case which is discussed in §4.

In §4, we consider two settings. Our first algorithms apply when f is known to depend only on ℓ variables. The construction of the favorable set of points where we ask for the values of f in these algorithms is based on having a family \mathcal{A} of partitions of $\{1, \dots, N\}$ into ℓ disjoint sets. The requirements on \mathcal{A} are given in what we call the **Partition Assumption**, formulated in §4. The second algorithms we consider apply to any continuous f which is defined on Ω and the performance of the algorithms depends on how well f can be approximated by functions which depend only on ℓ variables. The second algorithms require a second set \mathcal{B} of partitions which satisfy a property we call **Partition Assumption II**.

The smaller the collection \mathcal{A} , the smaller the number of point evaluations of f that is needed in our algorithms. Therefore, it is very interesting to give constructions of partitions which satisfy the **Partition Assumption** with $\#(\mathcal{A})$ small. The **Partition Assumption** is called perfect hashing in theoretical computer science (see [7, 9]). We recall what is known about constructions of such sets \mathcal{A} in the last section of this paper. There, for the convenience of the reader, we include a standard probabilistic argument to show the existence of sets \mathcal{A} with small cardinality which satisfy the **Partition Assumption** for general ℓ . We also discuss how to obtain collections \mathcal{B} which satisfy the **Partition Assumption II**.

Let us mention that hashing and perfect hashing techniques are frequently used in theoretical computer science to reduce computational cost. The closest work in that field to what we are doing in this paper is what are called the JUNTA theorems (see e.g. [11]). The JUNTA theorems use hashing to determine change variables in a discrete setting. While their techniques have similarity to our approach, they do not consider the approximation setting of our paper.

The problems we consider in this paper are in the language of information-based complexity described as the L_∞ approximation problem with standard information (i.e. function values). The distinction between our results and the literature in information-based complexity is that the latter typically has other model classes for sparsity (see [12] and their upcoming second volume).

The viewpoint of this paper is of a theoretical nature. We concentrate on the number of queries needed for approximate recovery of our model classes of functions. We do not discuss computational aspects of our algorithms.

2 Simple facts about approximation

Let $\Omega_0 := [0, 1]^\ell$. Given any $h = 1/L$ with L an integer, we denote by

$$\mathcal{L} := h\mathcal{L}_\ell := \{h(i_1, \dots, i_\ell) : 0 \leq i_1, \dots, i_\ell \leq L\},$$

the lattice of equally spaced points (spacing h) on Ω_0 . Given the values of a function $g \in C(\Omega_0)$ at the set of points \mathcal{L} , there are many ways to create a favorable approximation to g from these values. Rather than spell out exactly the approximation scheme and the corresponding smoothness classes which guarantee an approximation rate, we instead postulate the properties we need of such a system. We give some examples of such approximation schemes at the end of this section.

We now describe the formal setting in which we shall work. For each $h = 1/L$, we assume that A_h is a linear mapping from $C(\Omega_0)$ into itself which satisfies the following:

Approximation Properties:

- (i) The value of $A_h(g)$ depends only on $g|_{h\mathcal{L}_\ell}$. Thus, if $g = \tilde{g}$ on $h\mathcal{L}_\ell$, then $A_h(g) = A_h(\tilde{g})$.
- (ii) There is an absolute constant C_0 such that $\|A_h(g)\|_{C(\Omega_0)} \leq C_0 \max_{x \in h\mathcal{L}_\ell} |g(x)|$, for all $h = 1/L$, $L = 1, 2, \dots$
- (iii) $A_h(g) \equiv g$ if g is constant.
- (iv) If $g|_{h\mathcal{L}_\ell}$ depends only on the variables x_{j_1}, \dots, x_{j_k} , $k \leq \ell$ then $A_h(g)$ also depends only on these variables.
- (v) If π is any permutation of the variables $x = (x_1, \dots, x_\ell)$ (which can respectively be thought of as a permutation of the indices $\{1, \dots, \ell\}$), then $A_h(g(\pi(\cdot)))(\pi^{-1}(x)) = A_h(g)(x)$, $x \in \Omega_0$.

We define the following approximation class:

$$\mathcal{A}^s := \mathcal{A}^s((A_h)) = \{g \in C(\Omega_0) : \|g - A_h(g)\|_{C(\Omega_0)} \leq Ch^s, h = 1/L, L = 1, 2, \dots\}, \quad (2.1)$$

with semi-norm

$$|g|_{\mathcal{A}^s} := \sup_h \{h^{-s} \|g - A_h(g)\|_{C(\Omega_0)}\}. \quad (2.2)$$

We obtain the norm on \mathcal{A}^s by adding $\|\cdot\|_{C(\Omega_0)}$ to the semi-norm. As will be discussed below, there is typically a range $0 < s \leq S$, where the approximation classes can be characterized as smoothness spaces.

We need the following simple fact about \mathcal{A}^s functions.

Lemma 2.1 *Suppose $g \in \mathcal{A}^s$ and $|g(x)| \leq \epsilon$, $x \in h\mathcal{L}_\ell$. Then,*

$$\|g\|_{C(\Omega_0)} \leq C_0\epsilon + |g|_{\mathcal{A}^s}h^s,$$

where C_0 is the constant in **Approximation Property** (ii).

Proof: This follows directly from the triangle inequality

$$\|g\|_{C(\Omega_0)} \leq \|g - A_h(g)\|_{C(\Omega_0)} + \|A_h(g)\|_{C(\Omega_0)} \leq |g|_{\mathcal{A}^s}h^s + C_0\epsilon. \quad \square$$

There are several ways to construct operators A_h of the above form. However, spelling out their details can be cumbersome and this is why we have chosen the above approach of merely postulating their existence. The most prominent operators would be tensor product quasi-interpolant constructions such as those used in spline theory, shift invariant space theory, and wavelet theory (see e.g. [5, 10] as representative). Quasi-interpolants begin with a function ϕ , its dilate ϕ_h and the shift invariant space $\mathcal{S}_h(\phi)$ spanned by the translates of ϕ_h with respect to the lattice $h\mathcal{L}_\ell$. A common setting (for example in

multivariate box spline theory) is that the shifts of ϕ are locally linearly independent. It follows that the point evaluation functionals at the lattice points $h\mathcal{L}_\ell$ are linearly independent and span the space of all functionals defined on $\mathcal{S}_h(\phi)$. This enables one to construct projectors A_h mapping $C(\mathbb{R}^\ell)$ into $\mathcal{S}_h(\phi)$ of the form

$$A_h(f) = \sum_{p \in h\mathcal{L}_\ell} c_p(f) \phi_h(x - p),$$

where each linear functional c_p is a finite linear combination of point evaluation functionals at points from the lattice $h\mathcal{L}_\ell$ near p . These functionals can even be chosen so that the evaluations are done only at points inside Ω_0 whenever the support of $\phi_h(\cdot - p)$ intersects Ω_0 nontrivially. In quasi-interpolant constructions, the spaces \mathcal{A}^s correspond to the generalized Lipschitz spaces of order s (equivalently the Besov space $B_\infty^s(C(\Omega_0))$).

3 The case of one change variable

It will be instructive to consider first the case when f depends only on one coordinate variable where the arguments and proofs are most transparent. That is, we first suppose $f(x_1, \dots, x_N) = g(x_j)$ with j unknown to us. Later we treat the case where f is only approximated by such a function g .

3.1 Non-adaptive algorithms

We first consider non-adaptive algorithms in which we spell out all the query points in advance. Later, we consider adaptive algorithms which allow the query points to depend on the answers to the previous questions.

The set of points where we will ask for the values of f is the union of two point sets. The first of these sets consists of what we call *base points*. In the case of one variable the base points are all points in the set $\mathcal{P} := \{P_i := L^{-1}(i, i, \dots, i)\}_{i=0}^L$. Clearly there are $L + 1$ such base points. The important property of this set is that when its points are projected onto any of the coordinate axes, we get a set of $L + 1$ equally spaced points.

The second set of points we need are *padding points*. These points are used to find the coordinate j . Padding points are associated to a pair of points $P, P' \in \mathcal{P}$ and are constructed as follows. Every integer $j \in \{1, \dots, N\}$ has a unique binary representation $j = 1 + \sum_{k=0}^n b_k(j)2^k$, where $n := \lceil \log_2 N \rceil - 1$ and each bit $b_k(j) \in \{0, 1\}$. Given a pair of points $P, P' \in \mathcal{P}$ and a value of $k \in \{0, 1, \dots, n\}$, we define the point $[P, P']_k$ whose j -th coordinate is $P(j)$ (i.e. the same as the j -th coordinate of P) if $b_k(j) = 0$ and otherwise this coordinate of $[P, P']_k$ is defined to be the same as the j -th coordinate of P' .

3.1.1 Algorithm 1

For this first algorithm, we ask for the values of f at the base points in \mathcal{P} given above. We also ask for the values of f at the following set \mathcal{Q} of padding points. To each pair P_{i-1}, P_i , $i = 1, \dots, L$, of consecutive points, we associate the padding points $[P_{i-1}, P_i]_k$, $k = 0, \dots, n$. Thus the collection of padding points is $\mathcal{Q} := \{[P_{i-1}, P_i]_k, i = 1, \dots, L, k = 0, \dots, n\}$. Clearly there are $L(n + 1)$ points in \mathcal{Q} .

Notice that $f(P_i) = g(i/L)$, $i = 0, \dots, L$. Therefore, after receiving the values of f at the base points \mathcal{P} , we know $g(i/L)$, $i = 0, \dots, L$. We use the approximation operator A_h of the previous section to construct the function $\hat{g} = A_h(g)$. Note that the construction of \hat{g} only uses the points from \mathcal{P} . The function $\hat{g}(x_j)$ would provide a good approximation to f if we knew j . Also, observe that if f is constant on \mathcal{P} then we do not need to know j . If f is not constant on \mathcal{P} then we use the padding points to find j . There is an i such that $f(P_{i-1}) \neq f(P_i)$ with $1 \leq i \leq L$. For each $k = 0, \dots, n$, the value $f([P_{i-1}, P_i]_k)$ is either $f(P_{i-1})$ or $f(P_i)$ because of the way the padding points are constructed and the fact that f depends on only one variable. If it is $f(P_{i-1})$, then we know that $b_k(j) = 0$; if it is $f(P_i)$ then we know $b_k(j) = 1$. Thus from the answer to these questions, we know all the bits of j and hence we know j . We define our approximation to f to be $\hat{f}(x_1, \dots, x_N) := \hat{g}(x_j)$.

Theorem 3.1 *If $f(x_1, \dots, x_N) = g(x_j)$ with $g \in \mathcal{A}^s$, then the function \hat{f} defined above satisfies*

$$\|f - \hat{f}\|_{C(\Omega)} \leq |g|_{\mathcal{A}^s} h^s. \quad (3.1)$$

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

Proof: We have $f(x_1, \dots, x_N) - \hat{f}(x_1, \dots, x_N) = g(x_j) - A_h(g)(x_j)$. Therefore, (3.1) follows from (2.1) and (2.2) since $g \in \mathcal{A}^s$. \square

Note that the logarithm appearing in the number of evaluations needed of f is the price we pay for not knowing the change coordinate j .

3.1.2 Algorithm 2

The purpose of our second algorithm is to handle the case where f is not necessarily a function of just one variable but it can be approximated well by such a function. To describe this we introduce the following notation. Given a univariate function g , we define the multivariate functions

$$I_\nu(g)(x_1, \dots, x_N) := g(x_\nu), \quad \nu = 1, \dots, N.$$

Our starting assumption about f is that for some $g \in \mathcal{A}^s$, some $\nu \in \{1, \dots, N\}$, and some $\epsilon > 0$, we have

$$\|f - I_\nu(g)\|_{C(\Omega)} \leq \epsilon.$$

The algorithm given below does not need to know g, ν or ϵ .

In this second algorithm we take $\mathcal{Q} := \{[P_i, P_{i'}]_k, 0 \leq i < i' \leq L, k = 0, \dots, n\}$. Clearly there are now $(n+1)(L+1)L/2$ points in \mathcal{Q} . As in the first algorithm, we use the values of f at the points $P_i \in \mathcal{P}$, $i = 0, \dots, L$ and apply the operator A_h , $h = 1/L$ to receive a function \hat{g} .

Now to find a change coordinate j from this information, we choose a pair $P_i, P_{i'}$, $i < i'$, for which $|f(P_i) - f(P_{i'})|$ is the largest among all such pairs. To identify the change coordinate, we proceed as follows. We consider the value $f(Q_k)$ at each of the points $Q_k := [P_i, P_{i'}]_k$, $k = 0, \dots, n$. If this value is closest to $f(P_i)$, we assign the bit $b_k = 0$. If this value is closest to $f(P_{i'})$ or if there is a tie, we assign the bit $b_k = 1$. These

bits determine an integer $j := 1 + \sum_{k=0}^n b_k 2^k$. If $j \leq N$, we define $\hat{f}(x_1, \dots, x_N) = \hat{g}(x_j)$. Otherwise, we define $\hat{f}(x_1, \dots, x_N) := \hat{g}(x_1)$. As we shall see in the proof of the following theorem, the above algorithm does not necessarily find the change coordinate of f but, when it does not, then f does not deviate much and we can still approximate it well.

Theorem 3.2 *Suppose that f is a function of N variables for which there is a function $g \in \mathcal{A}^s$ and a $\nu \in \{1, \dots, N\}$, such that*

$$\|f - I_\nu(g)\|_{C(\Omega)} \leq \epsilon.$$

Then the function \hat{f} defined above satisfies

$$\|f - \hat{f}\|_{C(\Omega)} \leq (6C_0 + 1)\epsilon + 3|g|_{\mathcal{A}^s} h^s, \quad (3.2)$$

where C_0 is the constant of Lemma 2.1. The definition of \hat{f} uses at most $L + 1 + \lceil \log_2 N \rceil L(L + 1)/2$ point evaluations of f .

Proof: We consider two cases.

Case 1: We assume in this case that the maximum deviation in the values $f(P_i)$, $i = 0, \dots, L$, is greater than 4ϵ . We choose i, i' such that $|f(P_i) - f(P_{i'})|$ is largest. At each of the padding points $Q_k := [P_i, P_{i'}]_k$, $k = 0, \dots, n$, we have $|f(Q_k) - I_\nu(g)(Q_k)| \leq \epsilon$. Now if $b_k(\nu) = 0$ then $I_\nu(g)(Q_k) = I_\nu(g)(P_i)$ (since $I_\nu(g)$ is a function only of the ν -th variable) and, therefore, $f(Q_k)$ is within 2ϵ of $f(P_i)$ but further than 2ϵ from $f(P_{i'})$. This means that the bit b_k assigned by the algorithm is zero and, therefore, $b_k = b_k(\nu)$. The same conclusion holds if $b_k(\nu) = 1$. Hence the value j determined by the algorithm is equal to ν . We therefore obtain from the definition of \mathcal{A}^s that

$$\|f - \hat{f}\|_{C(\Omega)} \leq \|f - I_\nu(g)\|_{C(\Omega)} + \|g - \hat{g}\|_{C([0,1])} \leq \epsilon + |g|_{\mathcal{A}^s} h^s, \quad (3.3)$$

which is the desired inequality.

Case 2: In this case, the maximum deviation of f over the points P_i , $i = 0, \dots, L$, is at most 4ϵ . Hence the maximum deviation of g over the points $h\mathcal{L}_1 = \{0, 1/L, \dots, 1\}$ is at most 6ϵ . We consider the function $\tilde{g} = g - c$, where c is the average of the minimum and maximum values of g on $h\mathcal{L}_1$. Then $|\tilde{g}| \leq 3\epsilon$ on $h\mathcal{L}_1$ and $|\tilde{g}|_{\mathcal{A}^s} = |g|_{\mathcal{A}^s}$. From Lemma 2.1, we see that $\|\tilde{g}\|_{C([0,1])} \leq 3C_0\epsilon + |g|_{\mathcal{A}^s} h^s$. It follows that

$$\|I_\nu(g) - I_j(g)\|_{C(\Omega)} \leq \|I_\nu(g) - c\|_{C(\Omega)} + \|c - I_j(g)\|_{C(\Omega)} = 2\|\tilde{g}\|_{C([0,1])} \leq 6C_0\epsilon + 2|g|_{\mathcal{A}^s} h^s.$$

Hence,

$$\|f - I_j(g)\|_{C(\Omega)} \leq \|f - I_\nu(g)\|_{C(\Omega)} + \|I_\nu(g) - I_j(g)\|_{C(\Omega)} \leq (6C_0 + 1)\epsilon + 2|g|_{\mathcal{A}^s} h^s. \quad (3.4)$$

Finally, using that $\|f - I_j(g)\|_{C(\Omega)} \leq |g|_{\mathcal{A}^s} h^s$ with (3.4), we arrive at (3.2). Notice that in this case we may have selected a wrong change coordinate j . However, since the maximum deviation of f over P_i , $i = 1, \dots, L$, is small, estimate (3.2) still holds. \square .

3.2 Adaptive algorithms

The algorithms we have just described and analyzed are non-adaptive. We can save some on the number of point values we need for f if we work adaptively. We begin by asking for the values of f on the point set \mathcal{P} of base points exactly as before. However, now, for the adaptive version of Algorithm 1, we identify i such that $f(P_{i-1}) \neq f(P_i)$; if there is no such pair we do not have to identify a change coordinate since $A_h(g)$ is constant. To identify the change coordinate, we only ask for the values of f at the padding points $[P_{i-1}, P_i]_k$ associated with this pair. This means the total number of values we need for f is $L + 1 + \lceil \log_2 N \rceil$. Similarly, in Algorithm 2, we identify the pair $P_i, P_{i'}$ corresponding to maximum deviation of f on \mathcal{P} and ask only for the values of f at the padding points $[P_i, P_{i'}]_k$ associated with this pair. This again gives $L + 1 + \lceil \log_2 N \rceil$ point values.

In this adaptive version of Algorithm 1, one can prove that the number of points at which we require the value of f is (up to a fixed constant multiplicative factor) optimal, provided we use standard constructions of the operators A_h (such as univariate spline interpolants). This is proved by using widths and Kolmogorov entropy. However, this will be reported on in another work.

4 The general case of ℓ variables

In this section, we consider the general case where the number of variables in g is ℓ . We assume that we know ℓ (the algorithms work equally well if we only know a bound for ℓ). We present two algorithms which generalize Algorithms 1 and 2 from above. They have a similar flavor to the one variable case but have some important differences.

Our starting point is to assume that we have a set \mathcal{P} of base points in \mathbb{R}^N with certain properties. Let \mathcal{A} be a collection of partitions \mathbf{A} of $\{1, 2, \dots, N\}$. Each \mathbf{A} consists of ℓ disjoint sets A_1, \dots, A_ℓ . We require:

Partition Assumption: *The collection \mathcal{A} is rich enough so that given any ℓ distinct integers $i_1, \dots, i_\ell \in \{1, \dots, N\}$, there is a partition \mathbf{A} in \mathcal{A} such that each set in \mathbf{A} contains precisely one of the integers i_1, \dots, i_ℓ .*

This condition is known as perfect hashing in theoretical computer science and is heavily used in finding change coordinates in a different setting than ours (such as finding JUNTAs). It is an interesting question how to create such a class of partitions and how many partitions are needed to guarantee the **Partition Assumption**. We discuss this issue in the following section, where we show that $C(\ell) \log_2 N$ partitions suffice. But for the remainder of this section, we merely assume that we have such a collection \mathcal{A} in hand.

Corresponding to any $\mathbf{A} \in \mathcal{A}$, we construct the set of *base points*

$$P = h \sum_{i=1}^{\ell} \alpha_i \chi_{A_i}, \quad \alpha_i \in \{0, 1, \dots, L\}, \quad h = 1/L. \quad (4.1)$$

In other words, P has coordinate value $h\alpha_i$ at each of the coordinate indices in A_i . We denote by \mathcal{P} the set of all such base points. Note that there are $(L + 1)^\ell \#(\mathcal{A})$ such base points. An important property of the base points which we use often is the following:

Projection Property: Given any $\mathbf{j} = (j_1, \dots, j_\ell)$, with distinct $j_q \in \{1, \dots, N\}$, $q = 1, \dots, \ell$, and any integers $0 \leq i_1, \dots, i_\ell \leq L$, there is a point $P \in \mathcal{P}$ such that the coordinate j_ν of P is hi_ν , $\nu = 1, \dots, \ell$.

Indeed, it is enough to take a partition \mathbf{A} from \mathcal{A} such that each j_ν is in a different set A_i of \mathbf{A} . Then the point (4.1) with the appropriate value of $\alpha_i = i_{j_\nu}$ when $j_\nu \in A_i$ has the value hi_{j_ν} at coordinate j_ν .

In analogy with our previous notation, given a sequence $\mathbf{j} = (j_1, \dots, j_\ell)$, of distinct integers from $\{1, \dots, N\}$, and a function g defined on $\Omega_0 := [0, 1]^\ell$, we define

$$I_{\mathbf{j}}(g)(x_1, \dots, x_N) = g(x_{j_1}, \dots, x_{j_\ell}).$$

We also use the following restriction operator. Given any partition $\mathbf{A} = (A_1, \dots, A_\ell)$ of $\{1, 2, \dots, N\}$ we define the mapping $R_{\mathbf{A}}$ from functions on Ω into functions on $h\mathcal{L}_\ell$ by

$$R_{\mathbf{A}}(f)(h(i_1, \dots, i_\ell)) = f\left(h \sum_{j=1}^{\ell} i_j \chi_{A_j}\right), \quad h = 1/L.$$

4.1 General adaptive algorithm 1

In this section, we introduce an algorithm which gives an approximation to any function f which is equal to $I_{\mathbf{j}}(g)$ with both \mathbf{j} and g unknown to us. In contrast with the one variable case, we first describe the adaptive version of this algorithm and then later mention the modifications necessary to have a non-adaptive algorithm. We call this adaptive algorithm **GA Algorithm 1**, indicating that it is general (applies to a general ℓ) and adaptive. The algorithm requires us to know ℓ (essentially just a bound for ℓ) but the assumption of knowing ℓ is ameliorated in the second algorithm which works in an approximation setting.

We start **GA Algorithm 1** by asking for the values of f at all of the base points \mathcal{P} described in (4.1). We examine the values of f at these points and from these values we choose one of the partitions in \mathcal{A} , call it \mathbf{A}^* , as follows. Given any $\mathbf{A} \in \mathcal{A}$, $\mathbf{A} = (A_1, \dots, A_\ell)$, we examine the base points P subordinate to this \mathbf{A} . We say the set A_i is a change set if there are P and P' , both subordinate to \mathbf{A} , for which P and P' only differ on A_i and $f(P) \neq f(P')$. We define $n(\mathbf{A})$ as the number of sets A_i , $i = 1, \dots, \ell$, in \mathbf{A} , which are change sets. Now we define

$$\mathbf{A}^* := \operatorname{argmax}_{\mathbf{A} \in \mathcal{A}} n(\mathbf{A}).$$

We call any such partition a *maximal change partition* for f . We note that \mathbf{A}^* is not necessarily unique and so the statements below referring to \mathbf{A}^* refer to any of the \mathbf{A}^* 's.

We say that the change coordinate j_ν is *visible at scale* $h = 1/L$ if there exist two points $h(i_1, \dots, i_N)$ and $h(i'_1, \dots, i'_N)$, $0 \leq i_1, i'_1, \dots, i_N, i'_N \leq L$, which are identical in all coordinates except for the j_ν -th coordinate and $f(h(i_1, \dots, i_N)) \neq f(h(i'_1, \dots, i'_N))$. We can always take $i'_{j_\nu} = i_{j_\nu} + 1$. Then the following lemma holds.

Lemma 4.1 *If \mathbf{A}^* is any maximal change partition, then every change set A_i^* from \mathbf{A}^* contains exactly one coordinate visible at scale h . The sets A_i^* from \mathbf{A}^* which are not change sets do not contain any coordinates visible at scale h .*

Proof: Let j'_1, \dots, j'_k , $k \leq \ell$ be the change coordinates which are visible at scale h . From the **Partition Assumption** there is a partition \mathbf{A}^{**} such that each j'_ν lies in a different set A_i^{**} of \mathbf{A}^{**} . Let us now check that each set A_i^{**} which contains a visible change coordinate (call it j'_ν) is a change set. Indeed, there exists two points $h(i_1, \dots, i_N)$ and $h(i'_1, \dots, i'_N)$, $0 \leq i_1, i'_1, \dots, i_N, i'_N \leq L$, which are identical in all coordinates except for the j'_ν -th coordinate and $f(h(i_1, \dots, i_N)) \neq f(h(i'_1, \dots, i'_N))$. We define P as the point subordinate to \mathbf{A}^{**} such that for any A_μ^{**} which contains a visible change coordinate j , the coordinate values of P on A_μ^{**} is the same as i_j . For sets that do not contain a visible change coordinate, we can define the coordinate value in an arbitrary way. It follows that $f(P) = f(h(i_1, \dots, i_N))$. Indeed, we can change P to $h(i_1, \dots, i_N)$ by just altering coordinates which are not visible change coordinates of f at scale h . We define P' by using the same construction. Then we see that $f(P) = f(h(i_1, \dots, i_N)) \neq f(h(i'_1, \dots, i'_N)) = f(P')$ and so A_i^{**} is a change set.

We have just shown that there is a partition \mathbf{A}^{**} which contains k change sets. Obviously, no partition \mathbf{A} can contain more than k change sets since each change set must contain at least one visible change coordinate. Finally, for any partition \mathbf{A}^* such that $n(\mathbf{A}^*) = k$, we must have that each visible change coordinate lies in a different set A_i^* . \square

We can now easily identify each of the change coordinates that is visible at scale h . For the partition \mathbf{A}^* , we mark the change sets A_i^* (each one of them, by Lemma 4.1 contains a visible change coordinate). We take a pair of base points P, P' subordinate to \mathbf{A}^* which differ only on the coordinates in A_i^* and satisfy $f(P) \neq f(P')$. For P, P' we create the *padding points* $Q_k := [P, P']_k$, $k = 0, \dots, n$, corresponding to binary partitions as follows: Q_k and P differ only on the coordinates in A_i^* which have k -th binary bit equal to one, and on these coordinates Q_k has the same value as P' . We ask for the value of $f(Q_k)$ and check whether it is $f(P)$ (in which case we assign $b_k = 0$) or $f(P')$ (in which case $b_k = 1$). The bits b_k , $k = 0, \dots, n$, uniquely determine the change coordinate in A_i^* .

The change coordinates of f that are visible at scale h have been identified. There may be $k \leq \ell$ of these coordinates, so we add arbitrarily $\ell - k$ coordinates to obtain $\mathbf{j}' = (j'_1, j'_2, \dots, j'_\ell)$, with $1 \leq j'_1 < j'_2 < \dots < j'_\ell$.

We fix a partition \mathbf{A}' that separates all the coordinates in \mathbf{j}' and we consider the base points P defined in (4.1), subordinate to this partition. We can assume without loss of generality that $j'_1 \in A'_1, \dots, j'_\ell \in A'_\ell$. We define $\hat{g} = A_h(R_{\mathbf{A}'}(f))$ and $\hat{f}(x_1, \dots, x_N) := I_{\mathbf{j}'}(\hat{g}) = \hat{g}(x_{j'_1}, \dots, x_{j'_\ell})$.

Theorem 4.2 *If $f = I_{\mathbf{j}}(g)$ with $g \in \mathcal{A}^s$, then the function \hat{f} determined by **GA Algorithm 1** satisfies*

$$\|f - \hat{f}\|_{C(\Omega)} \leq |g|_{\mathcal{A}^s} h^s. \quad (4.2)$$

The number of point values used in the algorithm is at most

$$(L + 1)^\ell \#(\mathcal{A}) + \ell \lceil \log_2 N \rceil, \quad (4.3)$$

*where \mathcal{A} is the collection sets satisfying the **Partition Assumption**.*

Proof: The algorithm requires the values of f at each of the base points which is $\#(\mathcal{P}) = (L + 1)^\ell \#(\mathcal{A})$ and then for each cell of A^* which has a change coordinate it asks for

$n + 1 = \lceil \log_2 N \rceil$ padding points to determine the binary bits of the visible change coordinate in this cell. Since there are at most ℓ visible change coordinates we arrive at (4.3).

To prove the bound on the approximation error, we define $S_0 := A_h(g)$ and write

$$f - \hat{f} = I_{\mathbf{j}}(g) - I_{\mathbf{j}}(S_0) + I_{\mathbf{j}}(S_0) - I_{\mathbf{j}'}(\hat{g}). \quad (4.4)$$

The first term on the right side satisfies

$$\|I_{\mathbf{j}}(g) - I_{\mathbf{j}}(S_0)\|_{C(\Omega)} = \|g - A_h(g)\|_{C(\Omega_0)} \leq |g|_{\mathcal{A}^s} h^s.$$

From **Approximation Properties** (iii), (iv), and (v), and the fact that both \mathbf{j} and \mathbf{j}' contain the coordinate indices of any visible coordinates, we see that $I_{\mathbf{j}}(S_0) = I_{\mathbf{j}'}(\hat{g})$. This means that the second term on the right side of (4.4) is identically zero. We have therefore proved (4.2). \square .

4.2 A non-adaptive version of GA Algorithm 1

If we want to work non-adaptively, i.e. spell out all queries in advance independent of f , then we need to only make the following modification in **GA Algorithm 1**. Since we do not know which partition has maximal change, we need to define the padding points $[P, P']_k$, $k = 0, 1, \dots, n$, for each partition \mathbf{A} , each choice of points P, P' , subordinate to this partition and each choice of a set A_i from \mathbf{A} . The set \mathcal{Q} of all such padding points has cardinality $\lceil \log_2 N \rceil \ell L(L+1)^\ell \#(\mathcal{A})$. We can then proceed as in **GA Algorithm 1** to find a maximal change partition \mathbf{A}^* and then use the padding points to find the visible change coordinates. We see that the total number of queries needed in the non-adaptive algorithm is considerably more than in the adaptive case.

4.3 General adaptive algorithm 2

We now consider the case where f is not a function of ℓ variables but rather that it can be approximated by such a function. We assume that we are given a function f on Ω for which there is a function $g \in \mathcal{A}^s$ and a $\mathbf{j} = (j_1, \dots, j_\ell)$, $1 \leq j_1 < \dots < j_\ell \leq N$, such that

$$\|f - I_{\mathbf{j}}(g)\|_{C(\Omega)} \leq \epsilon.$$

We do not assume that we know ϵ, \mathbf{j} or g . However, we do assume we know ℓ . One could equally well work with just a bound for ℓ .

The algorithm we describe is adaptive. It begins with the same set \mathcal{P} of base points as in **GA Algorithm 1**. However, we modify significantly the padding points which enter into the algorithm.

For each \mathbf{A} and each $i = 1, \dots, \ell$, we choose exactly one pair of base points P, P' subordinate to \mathbf{A} as follows. We require that P and P' agree on all of the sets $A_\mu \neq A_i$. Among all these possible pairs of points we choose one for which the oscillation $\text{osc}(P, P') := |f(P) - f(P')|$ is maximal. There are $\ell \#(\mathcal{A})$ such pairs. We call these pairs *maximal*.

We construct padding points associated to any maximal pair P, P' . However, now the padding points are constructed by using a different set of partitions than the binary ones. We consider partitions \mathbf{B} of $\{1, \dots, N\}$ into two disjoint sets B_0 and B_1 . We introduce the following property of a set \mathcal{B} of such partitions:

Partition Assumption II: *The set \mathcal{B} of partitions is said to have this property if given $\ell + 1$ distinct integers j, j_1, \dots, j_ℓ , there is a partition $\mathbf{B} \in \mathcal{B}$ such that the set in \mathbf{B} that contains j does not contain any of the j_1, \dots, j_ℓ .*

We want such a set \mathcal{B} with cardinality as small as possible. We shall see in the next section that \mathcal{B} is easily constructed from perfect hashing sets. We fix a \mathcal{B} that satisfies **Partition Assumption II**.

For each maximal pair P, P' and each $\mathbf{B} \in \mathcal{B}$, we define two padding points $Q_\nu := [P, P']_{\mathbf{B}, \nu}$, $\nu = 0, 1$, as follows. The j -th coordinate of Q_ν for each $j \in A_\mu$, $\mu \neq i$, is the common j -th coordinate of P and P' . Namely, we do not alter the base points except on A_i . For each $j \in A_i$, the j -th coordinate of Q_ν is the same as that of P' if $j \in A_i \cap B_\nu$. Otherwise it is the same as that of P . In other words, the padding points have the same coordinates as P , except that the coordinates with indices in $B_\nu \cap A_i$ are changed to the coordinates of P' . For each maximal pair there will be at most $2\#(\mathcal{B})$ padding points, corresponding to the $\#(\mathcal{B})$ choices of \mathbf{B} and the two choices for ν .

We are only interested in certain maximal pairs. We call a pair P, P' *useful* if for each $\mathbf{B} \in \mathcal{B}$, there is exactly one value $\nu(\mathbf{B}) \in \{0, 1\}$ such that

$$|f([P, P']_{\mathbf{B}, \nu(\mathbf{B})}) - f(P')| < \frac{1}{4} \text{osc}(P, P'),$$

and

$$|f([P, P']_{\mathbf{B}, \mu}) - f(P)| < \frac{1}{4} \text{osc}(P, P'),$$

for $\mu \neq \nu(\mathbf{B})$.

For each maximal and useful pair of points P, P' which differ on A_i , we define

$$J_{P, P'} := \bigcap_{\mathbf{B} \in \mathcal{B}} B_{\nu(\mathbf{B})} \cap A_i.$$

We say that a change coordinate j_ν is ϵ -*visible at scale* h , if for some pair of points $P = h(i_1, \dots, i_N)$ and $P' = h(i'_1, \dots, i'_N)$, $0 \leq i_1, i'_1, \dots, i_N, i'_N \leq L$ which are identical in all coordinates except for the j_ν -th coordinate, we have

$$|f(P) - f(P')| \geq 12\epsilon. \tag{4.5}$$

The following lemma will show that the set of indices $J_{P, P'}$ is either empty or contains precisely one integer j .

Lemma 4.3 *The following properties hold:*

- (i) *For each maximal and useful pair P, P' the set $J_{P, P'}$ is either empty or it contains precisely one integer j ,*
- (ii) *For each change coordinate j_r which is ϵ -visible at scale h , there is a maximal, useful pair P, P' for which $J_{P, P'} = \{j_r\}$.*

Proof: (i) Given any two distinct integers $j, j' \in A_i$, we want to show that not both of these integers can be in $J_{P,P'}$. To see this, we take a partition \mathbf{B} such that $j \in B_0$ and $j' \in B_1$. The existence of such a partition follows from the **Partition Assumption II**. From the definition of useful, we cannot have $|f([P, P']_{\mathbf{B}, \nu}) - f(P')| < \text{osc}(P, P')/4$ for both $\nu = 0, 1$. So only one of these integers j, j' can be in $J_{P,P'}$.

(ii) Given a change coordinate j_r which is ϵ -visible at scale h , we know there are points $R := h(i_1, \dots, i_N)$ and $R' := h(i'_1, \dots, i'_N)$ at which $|f(R') - f(R)| \geq 12\epsilon$ and R and R' differ only in the coordinate j_r . By the **Partition Assumption**, we can choose a partition \mathbf{A} such that each set A_μ , $\mu = 1, \dots, \ell$, contains exactly one change coordinate. Let j_r be in the set A_i . Consider the pairs Q, Q' subordinate to \mathbf{A} , for which Q and Q' differ only on A_i . We can take such a pair, call it Q_0, Q'_0 , so that Q_0 is identical to R at each change coordinate and Q'_0 is identical to R' at each change coordinate. The pair Q_0, Q'_0 may not be maximal so we choose a pair P, P' which is maximal from the various Q, Q' . Then clearly $\text{osc}(P, P') \geq \text{osc}(Q_0, Q'_0) \geq 12\epsilon$. By construction, we have that $I_j(g)(Q_0) = I_j(g)(R)$ and $I_j(g)(Q'_0) = I_j(g)(R')$.

We want to show that P, P' is useful and $j_r \in B_{\nu(\mathbf{B})}$ for all $\mathbf{B} \in \mathcal{B}$. First note that $\text{osc}(P, P') \geq \text{osc}(Q_0, Q'_0) \geq |I_j(g)(R') - I_j(g)(R)| - 2\epsilon \geq 8\epsilon$. Now, given any $\mathbf{B} \in \mathcal{B}$, let $j_r \in B_{\nu(\mathbf{B})}$, $\nu(\mathbf{B}) \in \{0, 1\}$. Then $[P, P']_{\mathbf{B}, \nu(\mathbf{B})}$ agrees with P' in all change coordinates. Hence $I_j(g)([P, P']_{\mathbf{B}, \nu(\mathbf{B})}) = I_j(g)(P')$, and therefore we have

$$\begin{aligned} |f([P, P']_{\mathbf{B}, \nu(\mathbf{B})}) - f(P')| &\leq |f([P, P']_{\mathbf{B}, \nu(\mathbf{B})}) - I_j(g)([P, P']_{\mathbf{B}, \nu(\mathbf{B})})| + |I_j(g)(P') - f(P')| \\ &\leq 2\epsilon \leq \frac{1}{4}\text{osc}(P, P'). \end{aligned}$$

Similarly if μ is the complementary index in $\{0, 1\}$ to $\nu(\mathbf{B})$, then $[P, P']_{\mathbf{B}, \mu}$ agrees with P in all change coordinates and hence

$$\begin{aligned} |f([P, P']_{\mathbf{B}, \mu}) - f(P)| &\leq |f([P, P']_{\mathbf{B}, \mu}) - I_j(g)([P, P']_{\mathbf{B}, \mu})| + |I_j(g)(P) - f(P)| \\ &\leq 2\epsilon \leq \frac{1}{4}\text{osc}(P, P'). \end{aligned}$$

Thus, P, P' is useful and $j_r \in J_{P,P'}$, as desired. \square

Let us denote by $\mathcal{J} := \cup\{J_{P,P'} : P, P' \text{ is maximal and useful}\}$. For any $j \in \mathcal{J}$, there may be many useful pairs P, P' which generate j . We define $\text{osc}(j) := \max\{\text{osc}(P, P') : J_{P,P'} = \{j\}\}$. Note that the above argument shows that

$$\text{osc}(j) \geq 8\epsilon, \quad \text{whenever } j \text{ is an } \epsilon\text{-visible change coordinate at scale } h. \quad (4.6)$$

We can now describe our second algorithm.

GA Algorithm 2:

- (i) We identify the maximal pairs P, P' and check if they are useful or not.
- (ii) For each maximal, useful pair P, P' we find $J_{P,P'}$ and then the set \mathcal{J} .
- (iii) We choose ℓ distinct entries j'_1, \dots, j'_ℓ in \mathcal{J} for which $\text{osc}(j)$ is largest with ties preferring the smallest j . If $\#\mathcal{J} < \ell$, we add arbitrary coordinates to have ℓ of them to arrive at $1 \leq j'_1 < \dots < j'_\ell \leq N$.

(iv) Now that we have found the ℓ potential change coordinates $\mathbf{j}' = (j'_1, \dots, j'_\ell)$, we fix a partition $\mathbf{A}' = (A'_1, \dots, A'_\ell) \in \mathcal{A}$ which separates j'_1, \dots, j'_ℓ . The existence of such a partition follows from the **Partition Assumption**. We can assume without loss of generality that $j'_1 \in A'_1, \dots, j'_\ell \in A'_\ell$. We define our approximating function

$$\hat{f} = I_{\mathbf{j}'} A_h R_{\mathbf{A}'}(f). \quad (4.7)$$

The defined \hat{f} is not necessarily unique but the results which follow will hold for any such \hat{f} .

Lemma 4.4 *Every change coordinate j_1, \dots, j_ℓ which is ϵ -visible at scale h is in the list j'_1, \dots, j'_ℓ .*

Proof: If j_r is an ϵ -visible change coordinate at scale h , then by (4.6) $\text{osc}(j_r) \geq 8\epsilon$. Thus, it is enough to show that for any $j \in \mathcal{J}$ which is not a change coordinate, we have $\text{osc}(j) < 8\epsilon$. Suppose P, P' is any maximal, useful pair for which $J_{P, P'} = \{j\}$. We use **Partition Assumption II** to find a $\mathbf{B} \in \mathcal{B}$ such that $j \in B_0$ and all the $j_1, \dots, j_\ell \in B_1$ (the reverse case is handled in the same way). Since P, P' is useful and $j \in B_0$, we have that $\nu(\mathbf{B}) = 0$ and for $Q_0 := [P, P']_{\mathbf{B}, 0}$

$$|f(P') - f(Q_0)| \leq \frac{1}{4} \text{osc}(P, P'). \quad (4.8)$$

But we also have

$$|f(P) - f(Q_0)| \leq 2\epsilon + |I_{\mathbf{j}}(g)(P) - I_{\mathbf{j}}(g)(Q_0)| = 2\epsilon, \quad (4.9)$$

where we have used the fact that $I_{\mathbf{j}}(g)(P) = I_{\mathbf{j}}(g)(Q_0)$ because B_0 does not contain any change coordinate from \mathbf{j} . Combining (4.8) and (4.9), we obtain

$$\text{osc}(P, P') = |f(P') - f(P)| \leq \frac{1}{4} \text{osc}(P, P') + 2\epsilon.$$

Hence $\text{osc}(P, P') \leq \frac{8}{3}\epsilon$. This shows that $\text{osc}(j) \leq \frac{8}{3}\epsilon$, and proves the lemma. \square

Our main result of this section is the following theorem.

Theorem 4.5 *Suppose that $f \in C(\Omega)$ and there exists a function $g \in \mathcal{A}^s$ and a vector $\mathbf{j} = (j_1, \dots, j_\ell)$ such that $\|f - I_{\mathbf{j}}(g)\|_{C(\Omega)} \leq \epsilon$. Then the function \hat{f} created by **GA Algorithm 2** satisfies*

$$\|f - \hat{f}\|_{C(\Omega)} \leq |g|_{\mathcal{A}^s} h^s + (C_0 + 1)(28\ell + 1)\epsilon, \quad (4.10)$$

where C_0 is the constant of **Approximation Property** (ii). The number of point values used in the algorithm is

$$\leq (L + 1)^\ell \#(\mathcal{A}) + 2\ell \#(\mathcal{A}) \#(\mathcal{B}).$$

Proof: The algorithm requires the values of f at each of the base points whose number is $(L+1)^\ell \#(\mathcal{A})$ and then for each maximal pair (there are $\ell \#(\mathcal{A})$ such pairs) it asks for at most $2 \#(\mathcal{B})$ padding points to determine the maximal, useful pairs.

To prove (4.10), we write $f = I_{\mathbf{j}}(g) + \eta$, where $\|\eta\|_{C(\Omega)} \leq \epsilon$. It follows that

$$\hat{f} = I_{\mathbf{j}'} A_h R_{\mathbf{A}'} I_{\mathbf{j}}(g) + I_{\mathbf{j}'} A_h R_{\mathbf{A}'}(\eta).$$

From **Approximation Property** (ii), we obtain

$$\begin{aligned} \|f - \hat{f}\|_{C(\Omega)} &= \|I_{\mathbf{j}}(g) + \eta - I_{\mathbf{j}'} A_h R_{\mathbf{A}'} I_{\mathbf{j}}(g) - I_{\mathbf{j}'} A_h R_{\mathbf{A}'}(\eta)\|_{C(\Omega)} \\ &\leq \|I_{\mathbf{j}}(g) - I_{\mathbf{j}'} A_h R_{\mathbf{A}'} I_{\mathbf{j}}(g)\|_{C(\Omega)} + \|\eta\|_{C(\Omega)} + \|I_{\mathbf{j}'} A_h R_{\mathbf{A}'}(\eta)\|_{C(\Omega)} \\ &\leq \|I_{\mathbf{j}}(g) - I_{\mathbf{j}'} A_h R_{\mathbf{A}'} I_{\mathbf{j}}(g)\|_{C(\Omega)} + (C_0 + 1)\epsilon. \end{aligned}$$

To estimate the remaining term we let $1 \leq \bar{j}_1 < \dots < \bar{j}_r \leq N$, be the indices from \mathbf{j} that are ϵ -visible at scale h (see (4.5)). Recall that our algorithm has identified each of them and hence they are all in \mathbf{j}' . We define $\phi(x_1, \dots, x_N) := I_{\mathbf{j}}(g)(\bar{x})$ where \bar{x} is identical to x in all coordinates with indices $\bar{j}_1, \dots, \bar{j}_r$ and zero otherwise. Let us first note that

$$\|I_{\mathbf{j}}(g) - \phi\|_{C(\Omega)} \leq 14\ell\epsilon. \quad (4.11)$$

Indeed, for any coordinate in \mathbf{j} that is not in $\bar{\mathbf{j}}$, incrementing just the variable corresponding to this coordinate cannot change g more than 14ϵ because these coordinates are not ϵ visible at scale h . Since we can go from \bar{x} to x with $\leq \ell$ such increments, we arrive at (4.11). This allows us to estimate

$$\begin{aligned} \|I_{\mathbf{j}}(g) - I_{\mathbf{j}'} A_h R_{\mathbf{A}'} I_{\mathbf{j}}(g)\|_{C(\Omega)} &\leq \|I_{\mathbf{j}}(g) - \phi\|_{C(\Omega)} + \|\phi - I_{\mathbf{j}'} A_h R_{\mathbf{A}'} \phi\|_{C(\Omega)} \\ &\quad + \|I_{\mathbf{j}'} A_h R_{\mathbf{A}'}(\phi - I_{\mathbf{j}}(g))\|_{C(\Omega)} \\ &\leq 14\ell(C_0 + 1)\epsilon + \|\phi - I_{\mathbf{j}'} A_h R_{\mathbf{A}'} \phi\|_{C(\Omega)}, \end{aligned}$$

where we have used **Approximation Property** (ii). Now let \mathbf{A} be the partition which separates \mathbf{j} . We can again assume $j_\nu \in A_\nu$, $\nu = 1, \dots, \ell$. Then $I_{\mathbf{j}'} A_h R_{\mathbf{A}'} \phi = I_{\mathbf{j}} A_h R_{\mathbf{A}} \phi$ because of property (v) and the fact that $\bar{j}_1, \dots, \bar{j}_r$ are in \mathbf{j} and in \mathbf{j}' and ϕ only depends on the variables indexed by $\bar{j}_1, \dots, \bar{j}_r$. Hence,

$$\begin{aligned} \|\phi - I_{\mathbf{j}'} A_h R_{\mathbf{A}'} \phi\|_{C(\Omega)} &= \|\phi - I_{\mathbf{j}} A_h R_{\mathbf{A}} \phi\|_{C(\Omega)} \\ &\leq \|\phi - I_{\mathbf{j}}(g) - I_{\mathbf{j}} A_h R_{\mathbf{A}}(\phi - I_{\mathbf{j}}(g))\|_{C(\Omega)} \\ &\quad + \|I_{\mathbf{j}}(g) - I_{\mathbf{j}} A_h R_{\mathbf{A}} I_{\mathbf{j}}(g)\|_{C(\Omega)} \\ &\leq 14\ell(C_0 + 1)\epsilon + |g|_{\mathcal{A}^s} h^s, \end{aligned}$$

which proves the theorem. \square

4.4 A non-adaptive version of GA Algorithm 2

Similarly to **GA Algorithm 1**, if we want to work non-adaptively, then we need to only make the following modification in **GA Algorithm 2**. Since we do not know which pair of base points P, P' is maximal, we need to define the padding points $[P, P']_{\mathbf{B}, \nu}$, $\mathbf{B} \in \mathcal{B}$,

$\nu = 0, 1$ for each partition \mathbf{A} , each choice of points P, P' , subordinate to this partition and each choice of a set A_i from \mathbf{A} . The set \mathcal{Q} of all such padding points has cardinality $2\ell L(L+1)^\ell \#(\mathcal{B})\#(\mathcal{A})$. We can then proceed as in **GA Algorithm 2** to find the maximal pairs P, P' and then check whether they are useful or not. As before, we see that the total number of queries needed in the non-adaptive algorithm is considerably more than in the adaptive case.

5 Constructing separating partitions

The algorithms we have given begin with a set \mathcal{A} of partitions that satisfy the **Partition Assumption**. It is important to know how large \mathcal{A} needs to be for this assumption to hold. Indeed, $\#(\mathcal{A})$ controls the size of the sets of base points \mathcal{P} and padding points \mathcal{Q} where we sample f . For the completeness of our exposition, we give a probabilistic proof that for any ℓ there exists a set of partitions \mathcal{A} which satisfies the **Partition Assumption** and has reasonable cardinality.

We have already noted that the **Partition Assumption** is known in theoretical computer science as perfect hashing. We would like to thank Professor Janos Körner for pointing out bounds for the cardinality of such sets \mathcal{A} , see [7, 9]. Here, we give a simple probabilistic argument, which we learned from Tomasz Łuczak, for obtaining upper bounds on the cardinality of \mathcal{A} that gives estimates close to the best known.

Suppose we are given N and $\ell < N$. We are interested in partitions $\mathbf{A} = (A_1, \dots, A_\ell)$ of $\{1, \dots, N\}$ consisting of ℓ disjoint sets A_1, \dots, A_ℓ . We view each A_i as a bucket which will have in it a collection of integers from $\{1, \dots, N\}$. To create such sets, we make draws from a box of balls labeled $1, \dots, \ell$. We randomly draw the first ball. If this ball has label i then the integer one is placed into the bucket A_i . We then replace the first ball and randomly draw again, receiving a ball with label i' . We place the integer two into the bucket $A_{i'}$. We continue in this way N times, thereby putting each integer from $\{1, \dots, N\}$ into one of the sets A_1, \dots, A_ℓ . This gives the first partition $\mathbf{A} = (A_1, \dots, A_\ell)$. Notice that some of the sets A_1, \dots, A_ℓ may be empty. We repeat this experiment m times, which results in m partitions. We shall decide m later.

If we are given $\mathbf{j} = (j_1, \dots, j_\ell)$, then it is easy to see that the probability that a random partition \mathbf{A} separates the entries of \mathbf{j} into distinct sets is $\ell!/\ell^\ell$. Indeed, the probability that j_i is in A_i , for each $i = 1, \dots, \ell$, is $\ell^{-\ell}$. But any permutation of the j_i will do as well and we have $\ell!$ of these. So the probability that a random partition does not separate a given \mathbf{j} is $a := (1 - \frac{\ell!}{\ell^\ell})$. Therefore, if we have m independent partitions, the probability that none of them separates \mathbf{j} is a^m . There are $\binom{N}{\ell}$ ℓ -tuples $\mathbf{j} = (j_1, \dots, j_\ell)$, $j_1 < \dots < j_\ell$. Thus, if $\binom{N}{\ell} a^m < 1$, then a set of m random partitions will separate every \mathbf{j} with positive probability.

To see how large we need to take m we use Stirling's formula to find

$$\binom{N}{\ell} \left(1 - \frac{\ell!}{\ell^\ell}\right)^m \leq \binom{N}{\ell} \left(1 - \frac{\sqrt{2\pi\ell}}{e^\ell}\right)^m \leq N^\ell (1 - e^{-\ell})^m. \quad (5.1)$$

If we take $m = 2\ell e^\ell \ln N$ and use the fact that $(1 - 1/x)^x \leq e^{-1}$, for $x > 1$, the right side of (5.1) is $< N^\ell e^{-2\ell \ln N} \leq N^{-\ell}$. Thus, if we take $m \geq 2\ell e^\ell \ln N$ partitions generated

randomly, then with probability greater than $1 - N^{-\ell}$, the resulting set \mathcal{A} will satisfy the **Partition Assumption**.

We can also use perfect hashing to show the existence of sets \mathcal{B} of partitions which satisfy **Partition Assumption II**. We take a perfect hashing collection \mathcal{A} which separates all selections of $\ell + 1$ distinct integers chosen from $\{1, \dots, N\}$. Each partition $\mathbf{A} = (A_1, \dots, A_{\ell+1})$ generates $\ell + 1$ partitions into two sets $(A_j, \bigcup_{i \neq j} A_i)$. It is clear that the collection \mathcal{B} of all those partitions satisfy **Partition Assumption II**. There are $(\ell + 1)\#(\mathcal{A})$ such partitions in \mathcal{B} ; We know from the above arguments that \mathcal{A} can be constructed with $\#(\mathcal{A}) \leq 2(\ell + 1)e^{\ell+1} \ln N$ and so there are constructions which give \mathcal{B} with $\#(\mathcal{B}) \leq 2(\ell + 1)^2 e^{\ell+1} \ln N$. We note that we could also use a direct probabilistic argument (similar to that above) which gives the slightly better bound $\#(\mathcal{B}) \leq (\ell + 1)2^\ell \ln N$.

It remains an interesting question to design partitions constructively which could be used in conjunction with our algorithms for general ℓ .

Acknowledgment: The authors thank the referees for useful comments about this work.

References

- [1] M. Belkin and P. Niyogi, *Laplacian Eigenmaps for Dimensionality Reduction and Data Representation*, Neural Computation, **15** (2003), 1373–1396.
- [2] E. Candès, J. Romberg, and T. Tao, *Stable signal recovery from incomplete and inaccurate measurements*, Comm. Pure and Appl. Math., **59** (2006), 1207–1223.
- [3] A. Cohen, W. Dahmen and R. DeVore, *Compressed sensing and best k -term approximation*, JAMS, **22**(1) (2009), 211–231.
- [4] R. Coifman and M. Maggioni, *Diffusion wavelets*, Appl. Comp. Harm. Anal., **21**(1) (2006), 53–94.
- [5] C. de Boor, *Quasi-interpolants and approximation power of multivariate splines*, Computation of Curves and Surfaces, Kluwer, Dordrecht, 1990, 313–345.
- [6] D. Donoho, *Compressed Sensing*, IEEE Trans. Information Theory, **52** (2006), 1289–1306.
- [7] M. Fredman and J. Komlos, *On the size of separating systems and families of perfect hash functions*, SIAM J. Alg. Disc. Meth., **5** (1984), 61–68.
- [8] E. Greenshtein, *Best subset selection, persistence in high-dimensional statistical learning and optimization under ℓ_1 constraint*, Ann. Stat., **34** (2006), 2367–2386.
- [9] J. Körner and K. Marton, *New bounds for Perfect Hashing via Information Theory*, Europ. J. Combinatorics, **9** (1988), 523–530.

- [10] J. Levesley and M. Roach, *Quasi-interpolation on compact domains*, Approximation Theory, Wavelets and Applications, S. P. Singh (ed.), Kluwer Academic Publishers, 1995, 557–566.
- [11] E. Mossel, R. O’Donnell and R. Servedio, *Learning Juntas*, Proceedings of the 35th Annual ACM Symposium on Theory of Computing (STOC) San-Diego, 2003, 206–212.
- [12] E. Novak and H. Woźniakowski, *Tractability of Multivariate Problems vol. I: Linear Information*, European Math. Soc., 2008.
- [13] R. Todor and C. Schwab, *Convergence rates for sparse chaos approximations of elliptic problems with stochastic coefficients*, IMA J. Num. Anal., **27**(2) (2007), 232–261.

Ronald DeVore, Department of Mathematics, Texas A&M University, College Station, TX, rdevore@math.tamu.edu

Guergana Petrova, Department of Mathematics, Texas A&M University, College Station, TX, gpetrova@math.tamu.edu

Przemyslaw Wojtaszczyk, Institute of Applied Mathematics, University of Warsaw, wojtaszczyk@mimuw.edu.pl