Capturing Functions in High Dimension

Ronald DeVore
This talk will be concerned with approximating or capturing functions \( f \) of \( D \) variables with \( D \) large.
Capturing Functions in High Dimensions

- This talk will be concerned with approximating or capturing functions \( f \) of \( D \) variables with \( D \) large
- Many Application Domains: Parametric and Stochastic PDEs, Learning, Inverse problems, ...
Capturing Functions in High Dimensions

- This talk will be concerned with approximating or capturing functions $f$ of $D$ variables with $D$ large.

- Many Application Domains: Parametric and Stochastic PDEs, Learning, Inverse problems, ...

- $f$ may be Banach space valued but to make our life simple we will consider only real valued $f$. 
Capturing Functions in High Dimensions

- This talk will be concerned with approximating or capturing functions $f$ of $D$ variables with $D$ large

- Many Application Domains: Parametric and Stochastic PDEs, Learning, Inverse problems, ...

- $f$ may be Banach space valued but to make our life simple we will consider only real valued $f$

- Many reasonable settings that occur in applications
This talk will be concerned with approximating or capturing functions $f$ of $D$ variables with $D$ large.

Many Application Domains: Parametric and Stochastic PDEs, Learning, Inverse problems, ...

$f$ may be Banach space valued but to make our life simple we will consider only real valued $f$.

Many reasonable settings that occur in applications.

We are given a budget $n$ and can ask for the value of $f$ at $n$ points of our choosing - Each question is costly.
Capturing Functions in High Dimensions

- This talk will be concerned with approximating or capturing functions \( f \) of \( D \) variables with \( D \) large.

- Many Application Domains: Parametric and Stochastic PDEs, Learning, Inverse problems, ...

- \( f \) may be Banach space valued but to make our life simple we will consider only real valued \( f \).

- Many reasonable settings that occur in applications.

- We are given a budget \( n \) and can ask for the value of \( f \) at \( n \) points of our choosing - Each question is costly.

- From the answers we want to produce an accurate approximation to \( f \): For any other value of \( x \), we can cheaply produce an approximation to \( f(x) \).
Capturing Functions in High Dimensions

- This talk will be concerned with approximating or capturing functions $f$ of $D$ variables with $D$ large
- Many Application Domains: Parametric and Stochastic PDEs, Learning, Inverse problems, ...
- $f$ may be Banach space valued but to make our life simple we will consider only real valued $f$
- Many reasonable settings that occur in applications
- We are given a budget $n$ and can ask for the value of $f$ at $n$ points of our choosing - Each question is costly
- From the answers we want to produce an accurate approximation to $f$: For any other value of $x$, we can cheaply produce an approximation to $f(x)$
- Where should we query $f$?
The Challenge of the Problem

We need to assume something about $f$
The Challenge of the Problem

- We need to assume something about $f$
- Usual Model for functions is based on smoothness
The Challenge of the Problem

- We need to assume something about $f$
- Usual Model for functions is based on smoothness
- This model is not sufficient in high dimension
The Challenge of the Problem

- We need to assume something about $f$
- Usual Model for functions is based on smoothness
- This model is not sufficient in high dimension
- Curse of Dimensionality
The Challenge of the Problem

- We need to assume something about $f$
- Usual Model for functions is based on smoothness
- This model is not sufficient in high dimension
- Curse of Dimensionality
- If we only assume $f$ has $s$ orders of smoothness the best we can approximated is order $O(n^{-s/D})$ where $n$ is the number of parameters (dimension of approximation space) or number of queries of $f$ or number of computations
The Challenge of the Problem

- We need to assume something about $f$
- Usual Model for functions is based on smoothness
- This model is not sufficient in high dimension
- Curse of Dimensionality
- If we only assume $f$ has $s$ orders of smoothness the best we can approximated is order $O(n^{-s/D})$ where $n$ is the number of parameters (dimension of approximation space) or number of queries of $f$ or number of computations
- When $D$ is large $s$ would have to be very large to overcome this.
New Models For Functions

We need better models - not based solely on smoothness - that match real world functions
New Models For Functions

- We need better models - not based solely on smoothness - that match real world functions
- Popular Models: Sparsity or Compressibility
New Models For Functions

- We need better models - not based solely on smoothness - that match real world functions
- Popular Models: Sparsity or Compressibility
- $\psi_j$ (orthonormal) basis: $f = \sum_j c_j \psi_j$
New Models For Functions

- We need better models - not based solely on smoothness - that match real world functions.
- Popular Models: **Sparsity** or **Compressibility**
- $\psi_j$ (orthonormal) basis: $f = \sum_j c_j \psi_j$
- **Sparsity**: small number $k$ of coefficients are nonzero.
New Models For Functions

We need better models - not based solely on smoothness - that match real world functions

Popular Models: Sparsity or Compressibility

\[ f = \sum_j c_j \psi_j \]

\( \psi_j \) (orthonormal) basis

Sparsity: small number \( k \) of coefficients are nonzero

Compressibility: coefficients have some decay (when rearranged in decreasing size)
New Models For Functions

- We need better models - not based solely on smoothness - that match real world functions

- Popular Models: Sparsity or Compressibility

- $\psi_j$ (orthonormal) basis: $f = \sum_j c_j \psi_j$

- Sparsity: small number $k$ of coefficients are nonzero

- Compressibility: coefficients have some decay (when rearranged in decreasing size)

- Typical assumption is the coefficients are in some (weak) $\ell_p$ with $p$ small
New Models For Functions

- We need better models - not based solely on smoothness - that match real world functions

- Popular Models: Sparsity or Compressibility

  $\psi_j$ (orthonormal) basis: $f = \sum_j c_j \psi_j$

- Sparsity: small number $k$ of coefficients are nonzero

- Compressibility: coefficients have some decay (when rearranged in decreasing size)

  typical assumption is the coefficients are in some (weak) $\ell_p$ with $p$ small

- May be useful but it also suffers curse of dimensionality
We need better models - not based solely on smoothness - that match real world functions.

Popular Models: **Sparsity** or **Compressibility**

\[ f = \sum_j c_j \psi_j \]

- **Sparsity**: small number \( k \) of coefficients are nonzero
- **Compressibility**: coefficients have some decay (when rearranged in decreasing size)

Typical assumption is the coefficients are in some (weak) \( \ell_p \) with \( p \) small.

May be useful but it also suffers curse of dimensionality.

For example, for wavelet basis, such compressibility corresponds to some Besov smoothness \( f \in B^s_T(L_T) \) and again approximation is limited by \( O(n^{-s/D}) \).
HD Models

- Smoothness/Sparsity alone are usually not sufficient
HD Models

- Smoothness/Sparsity alone are usually not sufficient
- (New) approaches: Only a few variables or parameters are important
HD Models

- Smoothness/Sparsity alone are usually not sufficient
- (New) approaches: Only a few variables or parameters are important
- Manifold Learning; Laplacians on Graphs; Sensitivity Analysis; Variable Selection
HD Models

- Smoothness/Sparsity alone are usually not sufficient
- (New) approaches: Only a few variables or parameters are important
- Manifold Learning; Laplacians on Graphs; Sensitivity Analysis; Variable Selection
- Combine smoothness (sparsity) and variable reduction:
  \[ f(x) = g(\varphi(x)) \]
HD Models

- Smoothness/Sparsity alone are usually not sufficient
- (New) approaches: Only a few variables or parameters are important
- Manifold Learning; Laplacians on Graphs; Sensitivity Analysis; Variable Selection
- Combine smoothness (sparsity) and variable reduction:

\[ f(x) = g(\varphi(x)) \]

\[ \varphi : \mathbb{R}^D \rightarrow \mathbb{R}^d, \ d << D \]
HD Models

- Smoothness/Sparsity alone are usually not sufficient
- (New) approaches: Only a few variables or parameters are important
- Manifold Learning; Laplacians on Graphs; Sensitivity Analysis; Variable Selection
- Combine smoothness (sparsity) and variable reduction:
  \[ f(x) = g(\varphi(x)) \]
  \( \varphi : \mathbb{R}^D \rightarrow \mathbb{R}^d, \ d << D \)
  Perhaps \( \varphi(x) = Ax \) where \( A \) is a \( d \times D \) matrix
HD Models

- Smoothness/Sparsity alone are usually not sufficient
- (New) approaches: Only a few variables or parameters are important
- Manifold Learning; Laplacians on Graphs; Sensitivity Analysis; Variable Selection
- Combine smoothness (sparsity) and variable reduction:

\[ f(x) = g(\varphi(x)) \]

- \( \varphi : \mathbb{R}^D \rightarrow \mathbb{R}^d \), \( d << D \)
- Perhaps \( \varphi(x) = Ax \) where \( A \) is a \( d \times D \) matrix
- \( g \) is defined on \( \mathbb{R}^d \) has smoothness of order \( s \)
HD Models

- Smoothness/Sparsity alone are usually not sufficient
- (New) approaches: Only a few variables or parameters are important
- Manifold Learning; Laplacians on Graphs; Sensitivity Analysis; Variable Selection
- Combine smoothness (sparsity) and variable reduction:

\[ f(x) = g(\varphi(x)) \]

- \( \varphi : \mathbb{R}^D \rightarrow \mathbb{R}^d, \ d << D \)
- Perhaps \( \varphi(x) = Ax \) where \( A \) is a \( d \times D \) matrix
- \( g \) is defined on \( \mathbb{R}^d \) has smoothness of order \( s \)
- Parameters: \( d, D, s \), complexity of \( \varphi \)
HD Models

- Smoothness/Sparsity alone are usually not sufficient
- (New) approaches: Only a few variables or parameters are important
- Manifold Learning; Laplacians on Graphs; Sensitivity Analysis; Variable Selection
- Combine smoothness (sparsity) and variable reduction:
  \[ f(x) = g(\varphi(x)) \]
  - \( \varphi : \mathbb{R}^D \rightarrow \mathbb{R}^d, \ d << D \)
  - Perhaps \( \varphi(x) = Ax \) where \( A \) is a \( d \times D \) matrix
  - \( g \) is defined on \( \mathbb{R}^d \) has smoothness of order \( s \)
- Parameters: \( d, D, s \), complexity of \( \phi \)
- How friendly are such functions to approximation?
Recovery from Point Queries

Let assume that $f(x) = f(x_1, \ldots, x_D)$ is defined and continuous on the cube $\Omega := [0, 1]^D$ with $D$ large.
Recovery from Point Queries

Let assume that $f(x) = f(x_1, \ldots, x_D)$ is defined and continuous on the cube $\Omega := [0, 1]^D$ with $D$ large.

We shall consider two models for $f$. 
Recovery from Point Queries

- Let assume that $f(x) = f(x_1, \ldots, x_D)$ is defined and continuous on the cube $\Omega := [0, 1]^D$ with $D$ large.

- We shall consider two models for $f$:
  1. $f$ depends only on $d$ variables:
     
     $$f(x_1, \ldots, x_D) = g(x_{j_1}, \ldots, x_{j_d})$$

     where $d$ is small compared to $D$ and $g$ has some smoothness that may not be known.
Recovery from Point Queries

Let assume that $f(x) = f(x_1, \ldots, x_D)$ is defined and continuous on the cube $\Omega := [0, 1]^D$ with $D$ large.

We shall consider two models for $f$

(i) $f$ depends only on $d$ variables:

$$f(x_1, \ldots, x_D) = g(x_{j_1}, \ldots, x_{j_d}),$$

where $d$ is small compared to $D$ and $g$ has some smoothness that may not be known.

(ii) $f$ can be approximated by functions of the type (i)
Recovery from Point Queries

Let assume that $f(x) = f(x_1, \ldots, x_D)$ is defined and continuous on the cube $\Omega := [0, 1]^D$ with $D$ large.

We shall consider two models for $f$:

(i) $f$ depends only on $d$ variables:
$$f(x_1, \ldots, x_D) = g(x_{j_1}, \ldots, x_{j_d}),$$
where $d$ is small compared to $D$ and $g$ has some smoothness that may not be known.

(ii) $f$ can be approximated by functions of the type (i).

For this talk, we shall use smoothness conditions like $g \in C^s$ for some $s > 0$. 
Recovery from Point Queries

Let assume that $f(x) = f(x_1, \ldots, x_D)$ is defined and continuous on the cube $\Omega := [0, 1]^D$ with $D$ large.

We shall consider two models for $f$

(i) $f$ depends only on $d$ variables:
$$ f(x_1, \ldots, x_D) = g(x_{j_1}, \ldots, x_{j_d}), $$
where $d$ is small compared to $D$ and $g$ has some smoothness that may not be known.

(ii) $f$ can be approximated by functions of the type (i)

For this talk, we shall use smoothness conditions like $g \in C^s$ for some $s > 0$.

Our First Problem: Given a budget $n$ of point values we can ask of $f$ where should we take these samples and how well can we approximate $f$ from these?
If we know \( j := (j_1, \ldots, j_d) \) then sampling \( f \) at \((L + 1)^d\) equally spaced points in the \( d \) dimensional space spanned by the coordinate vectors \( e_{j_1}, \ldots, e_{j_d} \) we can recover \( f \) to accuracy \( C(s)\|g\|C_s L^{-s} \)
Benchmark

If we know $\mathbf{j} := (j_1, \ldots, j_d)$ then sampling $f$ at $(L + 1)^d$ equally spaced points in the $d$ dimensional space spanned by the coordinate vectors $e_{j_1}, \ldots, e_{j_d}$ we can recover $f$ to accuracy $C(s)\|g\|_{C^s} L^{-s}$.

Our problem is to sample at the fewest number of points in the case we do not know $\mathbf{j} := (j_1, \ldots, j_d)$. 
If we know $\mathbf{j} := (j_1, \ldots, j_d)$ then sampling $f$ at $(L + 1)^d$ equally spaced points in the $d$ dimensional space spanned by the coordinate vectors $e_{j_1}, \ldots, e_{j_d}$ we can recover $f$ to accuracy $C(s)\|g\|_{C^s}L^{-s}$.

Our problem is to sample at the fewest number of points in the case we do not know $\mathbf{j} := (j_1, \ldots, j_d)$.

Naively, we could consider all $d$ dimensional subspaces, take $L^d$ sample points in each.
If we know $j := (j_1, \ldots, j_d)$ then sampling $f$ at $(L + 1)^d$ equally spaced points in the $d$ dimensional space spanned by the coordinate vectors $e_{j_1}, \ldots, e_{j_d}$ we can recover $f$ to accuracy $C(s)\|g\|_{C^s} L^{-s}$.

Our problem is to sample at the fewest number of points in the case we do not know $j := (j_1, \ldots, j_d)$.

Naively, we could consider all $d$ dimensional subspaces, take $L^d$ sample points in each.

This would require $\binom{D}{d}(L + 1)^d$ points.
Benchmark

- If we know \( \mathbf{j} := (j_1, \ldots, j_d) \) then sampling \( f \) at \( (L + 1)^d \) equally spaced points in the \( d \) dimensional space spanned by the coordinate vectors \( e_{j_1}, \ldots, e_{j_d} \) we can recover \( f \) to accuracy \( C(s) \|g\| C_s L^{-s} \).

- Our problem is to sample at the fewest number of points in the case we do not know \( \mathbf{j} := (j_1, \ldots, j_d) \).

- Naively, we could consider all \( d \) dimensional subspaces, take \( L^d \) sample points in each.

- This would require \( \binom{D}{d} (L + 1)^d \) points.

- We want and can to do much better.
First Results

DeVore-Petrova-Wojtaszczyk
First Results

DeVore-Petrova-Wojtaszczyk

Theorem
(i) Assume \( f(x_1, \ldots, x_D) = g(x_{j_1}, \ldots, x_{j_d}) \). By making \( C(d, S)L^d(\log_2 D) \) adaptive point queries we can recover \( f \) by \( \hat{f} \) with the following accuracy

\[
\| f - \hat{f} \|_{C(\Omega)} \leq C(S, d)\| g^{(s)} \|_{C([0,1]^d)}L^{-s}
\]

(ii) Suppose we only know that there is a \( g \) and \( j_1, \ldots, j_d \) such that \( \| f(x_1, \ldots, x_D) - g(x_{j_1}, \ldots, x_{j_d}) \|_{C(\Omega)} \leq \epsilon \). By making \( C(d, S)L^d(\log_2 D) \) adaptive point queries we can recover \( f \) by \( \hat{f} \) to the accuracy

\[
\| f - \hat{f} \|_{C(\Omega)} \leq C(S, d)\{ \| g^{(s)} \|_{C([0,1]^d)}L^{-s} + \epsilon \}
\]
Partitions

We shall describe the points at which we query $f$. 
Partitions

- We shall describe the points at which we query \( f \).
- We say a collection \( \mathcal{A} \) of partitions \( \mathbf{A} = (A_1, \ldots, A_d) \) of \( \Lambda := \{1, 2, \ldots, D\} \) satisfy the **Partition Assumption** if...
Partitions

We shall describe the points at which we query $f$.

We say a collection $\mathcal{A}$ of partitions $A = (A_1, \ldots, A_d)$ of $\Lambda := \{1, 2, \ldots, D\}$ satisfy the Partition Assumption if

(i) For each $j = (j_1, \ldots, j_d)$, there is an $A \in \mathcal{A}$ such that no two $j_\nu$ lie in the same cell $A_i$. 
Partitions

We shall describe the points at which we query $f$

We say a collection $\mathcal{A}$ of partitions $\mathcal{A} = (A_1, \ldots, A_d)$ of $\Lambda := \{1, 2, \ldots, D\}$ satisfy the **Partition Assumption** if

1. For each $j = (j_1, \ldots, j_d)$, there is an $A \in \mathcal{A}$ such that no two $j_\nu$ lie in the same cell $A_i$

2. For each $j = (j_1, \ldots, j_k)$ and $j \neq j_\nu$, $\nu = 1, \ldots, d$, there is an $A$ such that the cell $A_i$ which contains $j$ contains none of the $j_\nu$, $\nu = 1, \ldots, d$
Partitions

- We shall describe the points at which we query $f$.

- We say a collection $\mathcal{A}$ of partitions $\mathbf{A} = (A_1, \ldots, A_d)$ of $\Lambda := \{1, 2, \ldots, D\}$ satisfy the **Partition Assumption** if
  1. For each $\mathbf{j} = (j_1, \ldots, j_d)$, there is an $A \in \mathbf{A}$ such that no two $j_\nu$ lie in the same cell $A_i$.
  2. For each $\mathbf{j} = (j_1, \ldots, j_k)$ and $j \neq j_\nu$, $\nu = 1, \ldots, d$, there is an $A$ such that the cell $A_i$ which contains $j$ contains none of the $j_\nu$, $\nu = 1, \ldots, d$.

- A family of partitions which satisfy (i) are called **Perfect Hashing** in combinatorics.
Partitions

- We shall describe the points at which we query $f$
- We say a collection $\mathcal{A}$ of partitions $\mathcal{A} = (A_1, \ldots, A_d)$ of $\Lambda := \{1, 2, \ldots, D\}$ satisfy the Partition Assumption if
  - (i) For each $j = (j_1, \ldots, j_d)$, there is an $A \in \mathcal{A}$ such that no two $j_\nu$ lie in the same cell $A_i$
  - (ii) For each $j = (j_1, \ldots, j_k)$ and $j \neq j_\nu$, $\nu = 1, \ldots, d$, there is an $A$ such that the cell $A_i$ which contains $j$ contains none of the $j_\nu$, $\nu = 1, \ldots, d$
- A family of partitions which satisfy (i) are called Perfect Hashing in combinatorics
- We will use these partitions to construct query points so we want $\mathcal{A}$ that satisfy the Partition Assumption with the smallest cardinality
Controlling Cardinality of $A$

It is easy to prove using probability that there exist $A$ that satisfy (i) with $\#A \leq Cde^d \log_2 D$.
Controlling Cardinality of $A$

It is easy to prove using probability that there exist $A$ that satisfy (i) with $\#A \leq Cde^d \log_2 D$.

For small $d$ one can do this constructively, e.g. $d = 2$ use binary partitions.
Controlling Cardinality of $A$

- It is easy to prove using probability that there exist $A$ that satisfy (i) with $\#A \leq Cde^d \log_2 D$

- For small $d$ one can do this constructively, e.g. $d = 2$ use binary partitions

- It is still an open problem to determine the asymptotic behavior of the smallest perfect hashing collections when $d \geq 3$
Controlling Cardinality of $A$

- It is easy to prove using probability that there exist $A$ that satisfy (i) with $\#A \leq Cde^d \log_2 D$.

- For small $d$ one can do this constructively, e.g. $d = 2$ use binary partitions.

- It is still an open problem to determine the asymptotic behavior of the smallest perfect hashing collections when $d \geq 3$.

- To satisfy (ii) of the Partition Assumption we have to enlarge Perfect Hashing constructions. Our current constructions give $\#A \leq d^2 e^{2d} \ln D$. 
Controlling Cardinality of $\mathcal{A}$

- It is easy to prove using probability that there exist $\mathcal{A}$ that satisfy (i) with $\#\mathcal{A} \leq Cde^d \log_2 D$

- For small $d$ one can do this constructively, e.g. $d = 2$ use binary partitions

- It is still an open problem to determine the asymptotic behavior of the smallest perfect hashing collections when $d \geq 3$

- To satisfy (ii) of the Partition Assumption we have to enlarge Perfect Hashing constructions. Our current constructions give $\#\mathcal{A} \leq d^2 e^{2d} \ln D$

- Probably this could be improved
Base points $\mathcal{P}$

The first points at which we query $f$ are what we call base points.
Base points $\mathcal{P}$

- The first points at which we query $f$ are what we call base points.
- The set $\mathcal{P}$ of base points is defined as

$$P = P_A := \sum_{i=1}^{d} \alpha_i \chi_{A_i}, \quad \alpha_i \in \{0, 1/L, \ldots, 1\}, \quad A \in \mathcal{A}$$
The first points at which we query $f$ are what we call base points.

The set $P$ of base points is defined as

$$P = P_A := \sum_{i=1}^{d} \alpha_i A_i, \quad \alpha_i \in \{0, 1/L, \ldots, 1\}, \quad A \in \mathcal{A}$$

There are $(L + 1)^d \# \mathcal{A}$ points in $P$.
The first points at which we query \( f \) are what we call base points.

The set \( \mathcal{P} \) of base points is defined as
\[
P = P_A := \sum_{i=1}^{d} \alpha_i \chi_{A_i}, \quad \alpha_i \in \{0, 1/L, \ldots, 1\}, \quad A \in \mathcal{A}
\]

There are \((L + 1)^d \#\mathcal{A}\) points in \( \mathcal{P} \).

**Projection Property:** The important property of this set is that for any \( \mathbf{j} = (j_1, \ldots, j_d), 1 \leq j_1 < j_2 < \cdots < j_d \leq D \)
the projection of \( \mathcal{P} \) onto the \( d \)-dimensional space spanned by \( e_{j_1}, \ldots, e_{j_d} \) contains a uniform grid of the cube \([0, 1]^d\) with spacing \( h := 1/L \).
Base points $\mathcal{P}$

- The first points at which we query $f$ are what we call base points.

- The set $\mathcal{P}$ of base points is defined as
  \[ P = P_A := \sum_{i=1}^{d} \alpha_i \chi_{A_i}, \quad \alpha_i \in \{0, 1/L, \ldots, 1\}, \quad A \in \mathcal{A} \]

- There are $(L + 1)^d \# \mathcal{A}$ points in $\mathcal{P}$.

- **Projection Property:** The important property of this set is that for any $j = (j_1, \ldots, j_d)$, $1 \leq j_1 < j_2 < \cdots < j_d \leq D$ the projection of $\mathcal{P}$ onto the $d$-dimensional space spanned by $e_{j_1}, \ldots, e_{j_d}$ contains a uniform grid of the cube $[0, 1]^d$ with spacing $h := 1/L$.

- For any $j = (j_1, \ldots, j_d)$ and any $k$-variate function $g$ let $G_j(x_1, \ldots, x_D) := g(x_{j_1}, \ldots, x_{j_d})$.
The first points at which we query $f$ are what we call base points.

The set $\mathcal{P}$ of base points is defined as

$$P = P_A := \sum_{i=1}^{d} \alpha_i \chi_{A_i}, \quad \alpha_i \in \{0, 1/L, \ldots, 1\}, \quad A \in \mathcal{A}$$

There are $(L + 1)^d \# \mathcal{A}$ points in $\mathcal{P}$.

**Projection Property:** The important property of this set is that for any $j = (j_1, \ldots, j_d)$, $1 \leq j_1 < j_2 < \cdots < j_d \leq D$, the projection of $\mathcal{P}$ onto the $d$-dimensional space spanned by $e_{j_1}, \ldots, e_{j_d}$ contains a uniform grid of the cube $[0, 1]^d$ with spacing $h := 1/L$.

For any $j = (j_1, \ldots, j_d)$ and any $k$-variate function $g$, let $G_j(x_1, \ldots, x_D) := g(x_{j_1}, \ldots, x_{j_d})$.

If $f = G_j$ for some $j$, then knowing $f$ on $\mathcal{P}$ will determine $g$ on a uniform grid with spacing $h$. 

Marne2010 – p. 11/21
The base points are not sufficient to determine the change coordinates.
Padding points \( Q \)

- The base points are not sufficient to determine the change coordinates.
- To determine the change coordinates we query \( f \) at certain padding points which are adaptively chosen.
Padding points $Q$

- The base points are not sufficient to determine the change coordinates.
- To determine the change coordinates we query $f$ at certain padding points which are adaptively chosen.
- A pair of points $P, P' \in \mathcal{P}$ is said to be admissible if they are subordinate to the same partition $\mathcal{A}$ and there is a cell $A_i$ of $\mathcal{A}$ such that $P$ and $P'$ agree on all cells $A_j$, $j \neq i$ and on $A_i$, $P$ and $P'$ differ by $\pm 1/L$. 
Padding points \( Q \)

- The base points are not sufficient to determine the change coordinates.

- To determine the change coordinates we query \( f \) at certain padding points which are adaptively chosen.

- A pair of points \( P, P' \in \mathcal{P} \) is said to be admissible if they are subordinate to the same partition \( A \) and there is a cell \( A_i \) of \( A \) such that \( P \) and \( P' \) agree on all cells \( A_j \), \( j \neq i \) and on \( A_i \), \( P \) and \( P' \) differ by \( \pm 1/L \).

- There are \( \leq 2d\#(\mathcal{P}) = 2d(L + 1)^d\#(A) \) such admissible pairs.
The base points are not sufficient to determine the change coordinates.

To determine the change coordinates we query $f$ at certain padding points which are adaptively chosen.

A pair of points $P, P' \in \mathcal{P}$ is said to be admissible if they are subordinate to the same partition $A$ and there is a cell $A_i$ of $A$ such that $P$ and $P'$ agree on all cells $A_j$, $j \neq i$ and on $A_i$, $P$ and $P'$ differ by $\pm 1/L$.

There are $\leq 2d\#(\mathcal{P}) = 2d(L + 1)^d\#(A)$ such admissible pairs.

Given an admissible pair $P, P'$ associated to $A$ and $A_i$ and given any $B \in \mathcal{P}$ and $\nu \in \{1, \ldots, d\}$, we define

$$[P, P']_{B, \nu} := \begin{cases} 
    P'(j), & \text{if } j \in A_i \cap B_\nu \\
    P(j), & \text{otherwise}
\end{cases}$$
Algorithm 1

- Intended for the case where $f = G_j$ for some $j = (j_1, \ldots, j_d)$
Algorithm 1

- Intended for the case where $f = G_j$ for some $j = (j_1, \ldots, j_d)$
- Given $f$, we ask for the values of $f$ at all points in $P \cup Q$
Algorithm 1

- Intended for the case where \( f = G_j \) for some \( j = (j_1, \ldots, j_d) \)
- Given \( f \), we ask for the values of \( f \) at all points in \( \mathcal{P} \cup \mathcal{Q} \)
- Given these values, from the Projection Property we can find \( g \) on the lattice

\[
\mathcal{L}_d := \{h(i_1, \ldots, i_d) : 1 \leq i_1, \ldots, i_d \leq L\}
\]
Approximating $g$

We construct a piecewise polynomial approximation $A_{r,h}(g)$ from these values as follows.
Approximating $g$

- We construct a piecewise polynomial approximation $A_{r,h}(g)$ from these values as follows.
- For each cell $I = h^d[i_1, i_1 + 1] \times \cdots \times [i_d, i_d + 1]$, we choose a tensor product grid consisting of $r^d$ points from $hL_d$ closest to $I$. 
We construct a piecewise polynomial approximation $A_{r,h}(g)$ from these values as follows:

1. For each cell $I = h^d[i_1, i_1 + 1] \times \cdots \times [i_d, i_d + 1]$, we choose a tensor product grid consisting of $r^d$ points from $h\mathcal{L}_d$ closest to $I$.

2. We define $p_I$ as the tensor product polynomial of degree $r - 1$ which interpolates $g$ at these points.
Approximating $g$

- We construct a piecewise polynomial approximation $A_{r,h}(g)$ from these values as follows.
- For each cell $I = h^d[i_1, i_1 + 1] \times \cdots \times [i_d, i_d + 1]$, we choose a tensor product grid consisting of $r^d$ points from $h\mathcal{L}_d$ closest to $I$.
- We define $p_I$ as the tensor product polynomial of degree $r-1$ which interpolates $g$ at these points.
- Then $A_{r,h}(g)(x) := p_I(x)$, $x \in I$, for all $I$ gives an approximation to $g$ satisfying

$$
\|g - A_{r,h}g\|_{C[0,1]^k} \leq C(s)\|g\|_{C^s} h^s
$$

as long as $s \leq r$. 
Finding change coordinates

Given any admissible pair $P, P'$, let $A$ be the subordinating partition of $P$ and $P'$ and let $A_i$ be the set in $A$ where $P$ and $P'$ take differing values.
Finding change coordinates

- Given any admissible pair $P, P'$, let $A$ be the subordinating partition of $P$ and $P'$ and let $A_i$ be the set in $A$ where $P$ and $P'$ take differing values.
- We examine the values of $f$ at all the padding points $Q$ associated to this pair.
Finding change coordinates

- Given any admissible pair \( P, P' \), let \( A \) be the subordinating partition of \( P \) and \( P' \) and let \( A_i \) be the set in \( A \) where \( P \) and \( P' \) take differing values.
- We examine the values of \( f \) at all the padding points \( Q \) associated to this pair.
- We say the pair \( P, P' \) is useful if for each \( B \in A \), there is exactly one value \( \nu = \nu(B) \) where \( f([P, P']_B, \nu) = f(P') \) and for all \( \mu \neq \nu \), we have \( f([P, P']_B, \mu) = f(P) \).
Finding change coordinates

Given any admissible pair \( P, P' \), let \( A \) be the subordinating partition of \( P \) and \( P' \) and let \( A_i \) be the set in \( A \) where \( P \) and \( P' \) take differing values.

We examine the values of \( f \) at all the padding points \( Q \) associated to this pair.

We say the pair \( P, P' \) is useful if for each \( B \in A \), there is exactly one value \( \nu = \nu(B) \) where \( f([P, P']_B, \nu) = f(P') \) and for all \( \mu \neq \nu \), we have \( f([P, P']_B, \mu) = f(P) \).

For each such admissible and useful pair, we define
\[
J_{P, P'} := \bigcap_{B \in A} B_{\nu(B)} \cap A_i
\]
Finding change coordinates

- Given any admissible pair \( P, P' \), let \( A \) be the subordinating partition of \( P \) and \( P' \) and let \( A_i \) be the set in \( A \) where \( P \) and \( P' \) take differing values.

- We examine the values of \( f \) at all the padding points \( Q \) associated to this pair.

- We say the pair \( P, P' \) is **useful** if for each \( B \in A \), there is exactly one value \( \nu = \nu(B) \) where \( f([P, P']_B, \nu) = f(P') \) and for all \( \mu \neq \nu \), we have \( f([P, P']_B, \mu) = f(P) \).

- For each such admissible and useful pair, we define \( J_{P, P'} := \bigcap_{B \in A} B_{\nu(B)} \cap A_i \).

- Either \( J_{P, P'} = \{ j \} \) with \( j \) a change coordinate or \( J_{P, P'} = \emptyset \).
Finding change coordinates

- Given any admissible pair \( P, P' \), let \( A \) be the subordinating partition of \( P \) and \( P' \) and let \( A_i \) be the set in \( A \) where \( P \) and \( P' \) take differing values.

- We examine the values of \( f \) at all the padding points \( Q \) associated to this pair.

- We say the pair \( P, P' \) is useful if for each \( B \in A \), there is exactly one value \( \nu = \nu(B) \) where \( f([P, P']_B, \nu) = f(P') \) and for all \( \mu \neq \nu \), we have \( f([P, P']_B, \mu) = f(P) \).

- For each such admissible and useful pair, we define
  \[
  J_{P, P'} := \bigcap_{B \in A} B_{\nu(B)} \cap A_i
  \]

- Either \( J_{P, P'} = \{j\} \) with \( j \) a change coordinate or \( J_{P, P'} = \emptyset \).

- Every change coordinate which is visible on \( h\mathcal{L}_d \) appears in some \( J_{P, P'} \)
Performance of Algorithm 1

- Algorithm 1 finds all change coordinates that are visible on $\mathcal{L}_d$
Performance of Algorithm 1

- Algorithm 1 finds all change coordinates that are visible on $\mathcal{L}_d$
- The number of these may be $< d$. Complete this to a vector $j' = (j'_1, \ldots, j'_d)$ in an arbitrary way
Algorithm 1 finds all change coordinates that are visible on $\mathcal{L}_d$

The number of these may be $< d$. Complete this to a vector $j' = (j'_1, \ldots, j'_d)$ in an arbitrary way.

Define $\hat{f} := A_{r,h}(g)(x_{j'_1}, \ldots, x_{j'_d})$
Performance of Algorithm 1

- Algorithm 1 finds all change coordinates that are visible on $\mathcal{L}_d$
- The number of these may be $< d$. Complete this to a vector $j' = (j'_1, \ldots, j'_d)$ in an arbitrary way
- Define $\hat{f} := A_{r,h}(g)(x_{j'_1}, \ldots, x_{j'_d})$
- If $f = G_j$ with $g \in C^s$, $s \leq r$, then
  \[ \|f - \hat{f}\|_{C(\Omega)} \leq C(s, r)\|g\|_{C^s h^s} \]
Performance of Algorithm 1

- Algorithm 1 finds all change coordinates that are visible on $\mathcal{L}_d$
- The number of these may be $< d$. Complete this to a vector $j' = (j'_1, \ldots, j'_d)$ in an arbitrary way
- Define $\hat{f} := A_{r,h}(g)(x_{j'_1}, \ldots, x_{j'_d})$
- If $f = G_j$ with $g \in C^s$, $s \leq r$, then
  $$\|f - \hat{f}\|_{C(\Omega)} \leq C(s, r)\|g\|_{C^s h^s}$$
- The number of point values used in Algorithm 1 is $\leq 2d^2(L + 1)^d(\#(A))^2$
Performance of Algorithm 1

- Algorithm 1 finds all change coordinates that are visible on $L_d$
- The number of these may be $< d$. Complete this to a vector $j' = (j_1', \ldots, j_d')$ in an arbitrary way.
- Define $\hat{f} := A_{r,h}(g)(x_{j_1'}, \ldots, x_{j_d'})$
- If $f = G_j$ with $g \in C^s$, $s \leq r$, then
  \[ \|f - \hat{f}\|_{C(\Omega)} \leq C(s, r)\|g\|_{C^s h^s} \]
- The number of point values used in Algorithm 1 is
  \[ \leq 2d^2(L + 1)^d(\#(A))^2 \]
- There is a second algorithm (adaptive) for the case when we only know $f$ can be approximated by $g(x_{j_1}, \ldots, x_{j_d})$
A Second Model for $f$

Cohen-DeVore-Daubechies-Kerkyacharian-Picard
A Second Model for $f$

- Cohen-DeVore-Daubechies-Kerkyacharian-Picard

We shall assume that $f(x_1, \ldots, x_D) = g(a \cdot x)$, $x \in \Omega := [0, 1]^D$ where $g \in C^s[0, 1]$, $1 < \bar{s} \leq s \leq S$ and $a \in \mathbb{R}^D$. 
A Second Model for $f$

- Cohen-DeVore-Daubechies-Kerkyacharian-Picard

We shall assume that $f(x_1, \ldots, x_D) = g(a \cdot x)$, $x \in \Omega := [0, 1]^D$ where $g \in C^s[0, 1]$, $1 < \bar{s} \leq s \leq S$ and $a \in \mathbb{R}^D$

- We assume $a_i \geq 0, i = 1, \ldots, D$, and WOLOG $\sum_{i=1}^{D} a_i = 1$
A Second Model for $f$

- Cohen-DeVore-Daubechies-Kerkyacharian-Picard

We shall assume that $f(x_1, \ldots, x_D) = g(a \cdot x)$,

$x \in \Omega := [0, 1]^D$ where $g \in C^s[0, 1]$, $1 < \bar{s} \leq s \leq S$ and $a \in \mathbb{R}^D$

We assume $a_i \geq 0$, $i = 1, \ldots, D$, and WOLOG $\sum_{i=1}^{D} a_i = 1$

More generally, one could consider $f(x_1, \ldots, x_D) = g(Ax)$ with $A$ a $d \times D$ Markov matrix
A Second Model for $f$

Cohen-DeVore-Daubechies-Kerkyacharian-Picard

We shall assume that $f(x_1, \ldots, x_D) = g(a \cdot x)$, \( x \in \Omega := [0, 1]^D \) where $g \in C^s[0, 1]$, $1 < \bar{s} \leq s \leq S$ and $a \in \mathbb{R}^D$

We assume $a_i \geq 0$, $i = 1, \ldots, D$, and WOLOG $\sum_{i=1}^{D} a_i = 1$

More generally, one could consider $f(x_1, \ldots, x_D) = g(Ax)$ with $A$ a $d \times D$ Markov matrix

Theorem: Assume $\|g\|_{C^s} \leq M_0$ and $\|a\|_{\ell_q} \leq M_1$. Then using $L$ point queries, we can recover $f$ by an approximant $\hat{f}$ satisfying

$$\|f - \hat{f}\|_{C} \leq C(S, \bar{s}, d, M_0, M_1)\left\{L^{-s} + \left\{\frac{\log \min(D/L,1)}{L}\right\}^{1/q-1}\right\}$$
Query Points

For $h := 1/L$, we ask for the values of $f$ at the points $ih(1, \ldots, 1), i = 0, \ldots, L$
Query Points

- For $h := 1/L$, we ask for the values of $f$ at the points $ih(1, \ldots, 1), i = 0, \ldots, L$
- This gives us the values of $g$ at $ih, i = 0, \ldots, L$ and allows us to construct $\hat{g}$ such that

$$\|g - \hat{g}\|_{C[0,1]} \leq C(s)h^s$$
Query Points

- For $h := 1/L$, we ask for the values of $f$ at the points $ih(1, \ldots, 1), i = 0, \ldots, L$

- This gives us the values of $g$ at $ih, i = 0, \ldots, L$ and allows us to construct $\hat{g}$ such that

  $$\|g - \hat{g}\|_{C[0,1]} \leq C(s)h^s$$

- We next want to approximate $a$
For $h := 1/L$, we ask for the values of $f$ at the points $ih(1, \ldots, 1), i = 0, \ldots, L$

This gives us the values of $g$ at $ih, i = 0, \ldots, L$ and allows us to construct $\hat{g}$ such that

$$\|g - \hat{g}\|_{C[0,1]} \leq C(s)h^s$$

We next want to approximate $a$

Choose $i, j$ such that $\frac{|g(ih) - g(jh)|}{|ih - jh|} =: A$ is largest
Query Points

- For $h := 1/L$, we ask for the values of $f$ at the points $ih(1,\ldots,1)$, $i = 0,\ldots,L$.
- This gives us the values of $g$ at $ih$, $i = 0,\ldots,L$ and allows us to construct $\hat{g}$ such that
  $$\|g - \hat{g}\|_{C[0,1]} \leq C(s)h^s$$
- We next want to approximate $a$.
- Choose $i, j$ such that $\frac{|g(ih) - g(jh)|}{|ih - jh|} =: A$ is largest.
- We adaptively bisect $[ih, jh]$ $L$ times always choosing the interval with largest divided difference to subdivide.
Query Points

- For $h := 1/L$, we ask for the values of $f$ at the points $ih(1, \ldots, 1), i = 0, \ldots, L$
- This gives us the values of $g$ at $ih, i = 0, \ldots, L$ and allows us to construct $\hat{g}$ such that
  \[ \|g - \hat{g}\|_{C[0,1]} \leq C(s)h^s \]
- We next want to approximate $a$
- Choose $i, j$ such that $\frac{|g(ih) - g(jh)|}{|ih - jh|} =: A$ is largest
- We adaptively bisect $[ih, jh]$ $L$ times always choosing the interval with largest divided difference to subdivide
- This gives an interval $I = [\alpha_0, \alpha_1]$ with $|I| \leq 2^{-L}$ and a point $\xi_0 \in I$ where $|g'(\xi_0)| \geq A$
Query Points

For $h := 1/L$, we ask for the values of $f$ at the points $ih(1, \ldots, 1)$, $i = 0, \ldots, L$

This gives us the values of $g$ at $ih$, $i = 0, \ldots, L$ and allows us to construct $\hat{g}$ such that

$$\|g - \hat{g}\|_{C[0,1]} \leq C(s)h^s$$

We next want to approximate $a$

Choose $i, j$ such that $\frac{|g(ih) - g(jh)|}{|ih - jh|} =: A$ is largest

We adaptively bisect $[ih, jh]$ $L$ times always choosing the interval with largest divided difference to subdivide

This gives an interval $I = [\alpha_0, \alpha_1]$ with $|I| \leq 2^{-L}$ and a point $\xi_0 \in I$ where $|g'(\xi_0)| \geq A$

$\eta$ the center of $I$
Approximating $a$

Let $\Phi$ be an $L \times D$ Bernoulli matrix with entries $\pm 1/\sqrt{L}$.
Approximating $\alpha$

- Let $\Phi$ be an $L \times D$ Bernoulli matrix with entries $\pm 1/\sqrt{L}$
- $b_1, \ldots, b_L$ the rows of $\Phi$
Approximating $\alpha$

- Let $\Phi$ be an $L \times D$ Bernoulli matrix with entries $\pm 1/\sqrt{L}$
- $b_1, \ldots, b_L$ the rows of $\Phi$
- We now ask for the value of $f$ at the points $\eta(1, 1, \ldots, 1) + \mu b_i$, $i = 1, \ldots, L$, where $\mu := \frac{\sqrt{L}\delta}{2}$
Approximating $a$

- Let $\Phi$ be an $L \times D$ Bernoulli matrix with entries $\pm 1/\sqrt{L}$
- $b_1, \ldots, b_L$ the rows of $\Phi$
- We now ask for the value of $f$ at the points $
\eta(1, 1, \ldots, 1) + \mu b_i, \ i = 1, \ldots, L$, where $\mu := \frac{\sqrt{L}\delta}{2}$
- These queries in turn gives the values $g(\eta + \mu b_i \cdot \alpha), \ i = 1, \ldots, L$. All of the points $\eta + \mu b_i \cdot \alpha$ are in $I$
Approximating $\alpha$

Let $\Phi$ be an $L \times D$ Bernoulli matrix with entries $\pm 1/\sqrt{L}$

$b_1, \ldots, b_L$ the rows of $\Phi$

We now ask for the value of $f$ at the points

$\eta(1, 1, \ldots, 1) + \mu b_i, \ i = 1, \ldots, L$, where $\mu := \frac{\sqrt{L}\delta}{2}$

These queries in turn gives the values $g(\eta + \mu b_i \cdot a)$, $i = 1, \ldots, L$. All of the points $\eta + \mu b_i \cdot a$ are in $I$

$\hat{y}_i := \frac{2}{\sqrt{L}} \left[ \frac{g(\eta+\mu b_i \cdot a) - g(\eta)}{g(\alpha_0+\delta) - g(\alpha_0)} \right] = \frac{2}{\sqrt{L}} \left[ \frac{g'(\xi_1)\mu b_i \cdot a}{g'(\xi_0)\delta} \right]$

$= b_i \cdot a \left[ 1 + \frac{g'(\xi_1) - g'(\xi_0)}{g'(\xi_0)} \right] = b_i \cdot a \left[ 1 + \epsilon_i \right]$
Let $\Phi$ be an $L \times D$ Bernoulli matrix with entries $\pm 1/\sqrt{L}$

$b_1, \ldots, b_L$ the rows of $\Phi$

We now ask for the value of $f$ at the points $\eta(1, 1, \ldots, 1) + \mu b_i$, $i = 1, \ldots, L$, where $\mu := \frac{\sqrt{L}\delta}{2}$

These queries in turn gives the values $g(\eta + \mu b_i \cdot a)$, $i = 1, \ldots, L$. All of the points $\eta + \mu b_i \cdot a$ are in $I$

$$\hat{y}_i := \frac{2}{\sqrt{L}} \left[ \frac{g(\eta + \mu b_i \cdot a) - g(\eta)}{g(\alpha_0 + \delta) - g(\alpha_0)} \right] = \frac{2}{\sqrt{L}} \left[ \frac{g'((\xi_1)\mu b_i \cdot a)}{g'((\xi_0)\delta)} \right]$$

$$= b_i \cdot a \left[ 1 + \frac{g'(\xi_1) - g'(\xi_0)}{g'(\xi_0)} \right] = b_i \cdot a \left[ 1 + \epsilon_i \right]$$

$$|\epsilon_i| \leq CA^{-1}2^{-L}M_0L^{-\bar{s}}$$
Compressed sensing allows us to decode

\[ \hat{a}_i := \arg \min_{\Phi z = \hat{y}_i} \| z \|_1 \]
Compressed sensing allows us to decode
\[ \hat{a}_i := \arg\min_{\Phi z = \hat{y}_i} \|z\|_{\ell_1} \]
\[ \hat{a} := (\hat{a}_1, \ldots, \hat{a}_D) \]
Compressed sensing allows us to decode
\[ \hat{a}_i := \arg\min_{\Phi z = \hat{y}_i} \| z \|_{\ell_1} \]
\[ \hat{a} := (\hat{a}_1, \ldots, \hat{a}_D) \]
\[ \| a - \hat{a} \|_{\ell_1} \leq C \{ \frac{\log(D/L)}{L} \}^{1/q - 1} + LM_0 A^{-1} 2^{-\ell \bar{s}} \]
Compressed sensing allows us to decode
\[ \hat{a}_i := \arg\min_{\Phi z = \hat{y}_i} \| z \|_{\ell_1} \]
\[ \hat{a} := (\hat{a}_1, \ldots, \hat{a}_D) \]
\[ \| a - \hat{a} \|_{\ell_1} \leq C \left\{ \frac{\log(D/L)}{L} \right\}^{1/q-1} + L M_0 A^{-1} 2^{-\ell \bar{s}} \]
\[ \hat{f}(x) := \hat{g}(\hat{a} \cdot x) \text{ satisfies Theorem} \]
Compressed sensing allows us to decode
\[ \hat{a}_i := \text{argmin}_{\Phi z = \hat{y}_i} \| z \|_{\ell_1} \]
\[ \hat{a} := (\hat{a}_1, \ldots, \hat{a}_D) \]
\[ \| a - \hat{a} \|_{\ell_1} \leq C \left( \frac{\log(D/L)}{L} \right)^{1/q-1} + LM_0 A^{-1} 2^{-\ell s} \]
\[ \hat{f}(x) := \hat{g}(\hat{a} \cdot x) \] satisfies Theorem
Case \( A \leq M_0 L^{-s} \) then \( g \) does not vary
Compressed sensing allows us to decode

\[ \hat{a}_i := \arg\min_{\Phi z = \hat{y}_i} \| z \|_{\ell_1} \]

\[ \hat{a} := (\hat{a}_1, \ldots, \hat{a}_D) \]

\[ \| a - \hat{a} \|_{\ell_1} \leq C\left\{ \frac{\log(D/L)}{L} \right\}^{1/q - 1} + LM_0 A^{-1/2 - \ell \bar{s}} \]

\[ \hat{f}(x) := \hat{g}(\hat{a} \cdot x) \] satisfies Theorem

Case \( A \leq M_0 L^{-s} \) then \( g \) does not vary

Case \( A \geq M_0 L^{-s} \) then

\[ |f(x) - \hat{f}(x)| \leq |g(a \cdot x) - g(\hat{a} \cdot x)| + |g(\hat{a} \cdot x) - \hat{g}(\hat{a} \cdot x)| \leq M_0 \| a - \hat{a} \|_{\ell_1} + \| g - \hat{g} \|_{C[0,1]} \]
Final Remarks

- The result cannot be improved (save for the constant)
Final Remarks

- The result cannot be improved (save for the constant)
- To achieve $L^{-s}$ we need $O(L)$ points
Final Remarks

- The result cannot be improved (save for the constant)
- To achieve $L^{-s}$ we need $O(L)$ points
- By considering the functions $a \cdot x$, $\|a\|_{\ell_q} \leq M_1$ and lower bounds for Gelfand widths (Foucart, Rauhut, Pajor, Ullrich) we need $O(L)$ points for the second term accuracy
Final Remarks

- The result cannot be improved (save for the constant)
- To achieve $L^{-s}$ we need $O(L)$ points
- By considering the functions $a \cdot x, \|a\|_{\ell_q} \leq M_1$ and lower bounds for Gelfand widths (Foucart, Rauhut, Pajor, Ullrich) we need $O(L)$ points for the second term accuracy
- Why $\bar{s} > 1$?
Final Remarks

- The result cannot be improved (save for the constant)
- To achieve $L^{-s}$ we need $O(L)$ points
- By considering the functions $a \cdot x$, $\|a\|_{\ell_q} \leq M_1$ and lower bounds for Gelfand widths (Foucart, Rauhut, Pajor, Ullrich) we need $O(L)$ points for the second term accuracy
- Why $\bar{s} > 1$?
- We do not have the stability we had in the first setting