Approximation and Data Assimilation

Ronald DeVore

Collaborators: Peter Binev, Albert Cohen,

Wolfgang Dahmen, Guergana Petrova,

Przemek Wojtaszczyk

State of Numerical Computation

- Many pressing scientific problems challenge our computational ability
  - Atmospheric modeling: predicting climate change
  - Monitoring threat activities
  - Contaminant transport
  - Optimal engineering design
  - Medical diagnostics
  - Modeling the internet
  - Option pricing, bond valuation
  - ....
- GOAL OF NUMERICAL COMPUTATION: Build the most efficient numerical procedures to treat these problems

The Computational Task

- Generally one of the following:
  - Approximate an unknown function u - called the target function
  - Compute a quantity of interest about u such as an integral of u or the max or min of u
- An important issue is what information is available about u: This is described by a model class K
  - In numerical PDEs we know u is the solution to a PDE with known coefficients, initial values, boundary values
  - Classically, model classes describe the smoothness that is known about the target function u
  - Selection of the correct model class is a key issue since it governs best algorithms and expected performance

Outline of this Talk

- This talk will emphasize the following issues:
  - I. The Role of Approximation Theory
  - II. Finding the best approximation scheme given the model class K
  - III. Building optimal algorithms
    - We will emphasize this for problems of data fitting

Role of Approximation Theory

- Any numerical procedure is based on some form of approximation: polynomials, splines, Fourier, wavelets, etc.
- Approximation Theory aims at exactly characterizing the performance of any proposed method of approximation and thereby giving a benchmark for the optimal numerical procedure based on the chosen method of approximation
- It can also provide a guide to
  - which method of approximation to chose
  - how to build an optimal algorithm for the chosen method

Types of Approximation

- We fix a norm $\|\cdot\|_X$ in which to measure error, e.g. $\|u\|_{L_p(\Omega)} := (\int_\Omega u(x)|^p \, dx)^{1/p}$
- Approximation methods are linear or nonlinear
  - Linear methods of approximation
    - $X_0, X_1 \cdots, X_n, \cdots$ linear spaces with $\dim(X_n) = n$
    - We approximate u by the elements of X_n giving error $E_n(u)_X := E(u, X_n)_X := \inf_g \in X_n \|u - g\|$
    - If K is a model class then performance of X_n on K is $\operatorname{dist}(K, X_n) := \sup_{u \in K} E_n(u)$
    - Optimal performance on K given by the nb- width $d_n(K)_X := \inf_{\dim(Y)=n} \operatorname{dist}(K, Y)_X$
- Approximation Classes: For $\alpha > 0$ $\mathcal{A}^\alpha((X_n)_{n \geq 1}) := \{u \in X : E_n(u) \leq M n^{-\alpha}\}$
- $\|u\|_{\mathcal{A}^\alpha}$ is the smallest M

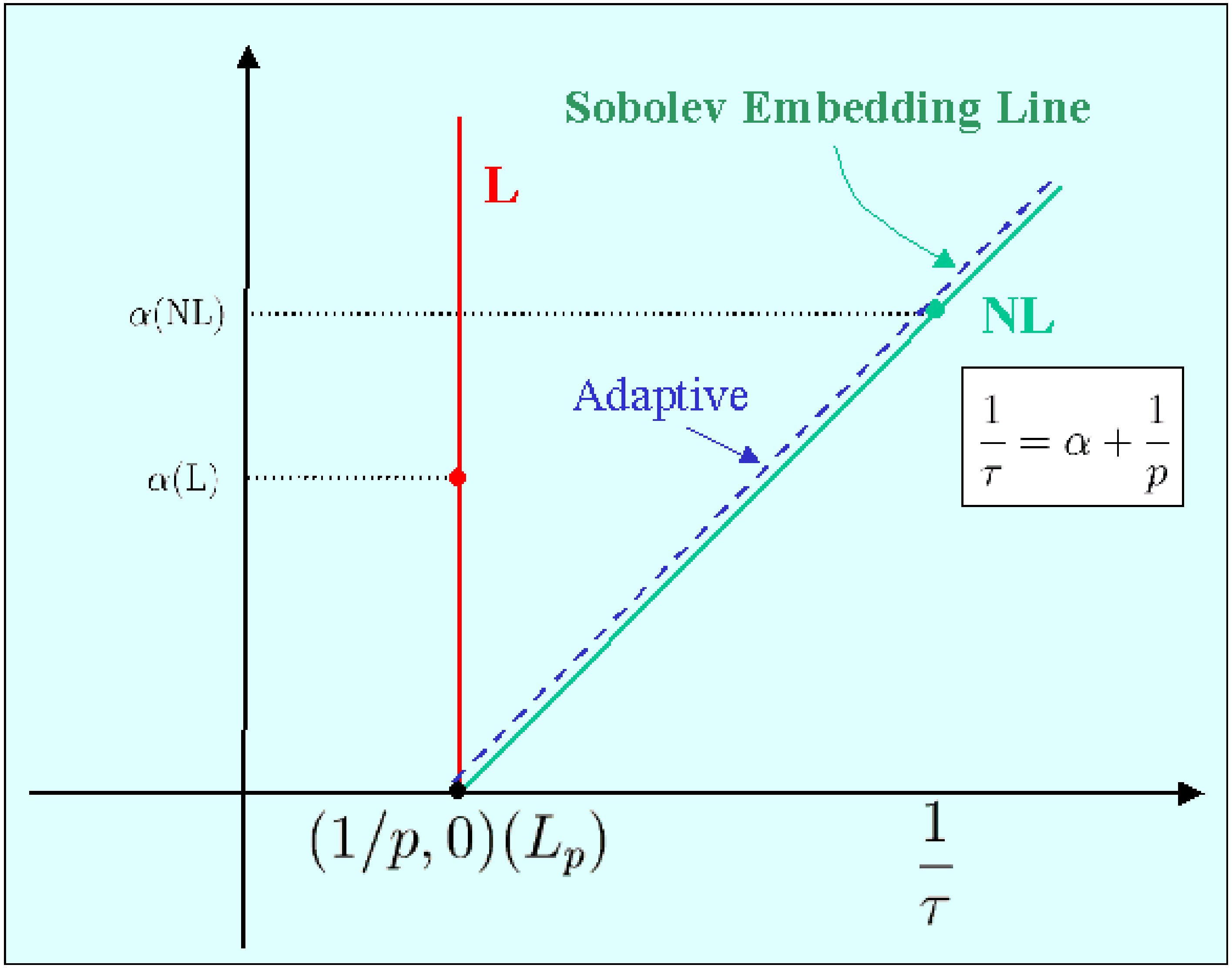
Nonlinear Approximation

- X_n is replaced by a nonlinear space Σ_n
  - n term approximation: $\mathcal{D}$ a dictionary (redundant family of functions)
    $\Sigma_n := \{g = \sum_{j=1}^n \alpha_j \phi_j : \ \phi_1, \dots, \phi_n \in \mathcal{D}\}$
  - Library approximation: $\mathcal{L} := \{Y : \dim(Y) = n\}$ with $\#(\mathcal{L}) = N$ finite $\mathrm{dist}(f, \mathcal{L})_X := \inf_{Y \in \mathcal{L}} \mathrm{dist}(f, Y)_X$
  - Piecewise Polynomial Approximation of functions on $\Omega \subset \mathbb{R}^d$: Divide Ω into n cells Ω_i depending on u and approximate u by g which is a piecewise polynomial of degree m on each cell Ω_i
  - Adaptive piecewise polynomial approximation: Generate the partition by an adaptive algorithm: typically subdividing the cell which has biggest error

Canonical results of Approximation

- Canonical results in AT characterize $\mathcal{A}^\alpha$
- Example: Approximate functions in $X := C[0,1]$
- Linear Approximation:
  - Divide $[0,1]$ into n equally spaced intervals. X_n the space of piecewise constant functions on this partition
- Nonlinear Approximation:
  - $\Sigma_n := \{S : S \text{ is piecewise constant with } n \text{ pieces}\}$
- Characterization of approximation spaces
  - $\mathcal{A}^1(X_n)_{n\geq 1}) = \text{Lip } 1$
  - $\mathcal{A}^1((\Sigma_n)_{n\geq 1}) = \text{BV} \cap C[0,1]$
- This shows the typical advantage of nonlinear approximation: need less smoothness for f

Performance in L_p- one Variable



Curse of Dimensionality

- Most of our numerical challenges involve functions that depend on many (say D) variables/parameters
- The classical numerical methods such as splines or FEM fail in this case because of the so-called curse of dimensionality
  - Suppose the assumption is that the targets function f is real valued and has smoothness (of order s)
    - Approximation theory tells us with n computations we can only capture F to accuracy $C(D,s)n^{-s/D}$ where D is the number of variables
    - When D is large than s must also be very large to guarantee any reasonable accuracy
    - No control over s which is inherent in the problem
    - So conventional assumptions on F and conventional numerical methods will not work

Example (Novak-Wozniakowski)

- To drive home the debilitating effect of high dimensions consider the following example
$\Omega := [0,1]^D, \quad X = C(\Omega), \quad \mathcal{K} := \{F : \|D^\nu F\|_{L_\infty} \le 1, \forall \nu\}$
- Any algorithm which computes for each $F \in \mathcal{K}$ an approximation $\hat{F}$ to accuracy $1/2$ in L_∞ will need at least $2^{D/2}$ FLOPS
- So if $D = 100$, we would need at least $2^{50} \asymp 10^{15}$ computations to achieve even the coarsest resolution
- This phenomenon is referred to as The Curse of Dimensionality
- How can we overcome this Curse?

The Remedy

- Conventional thought is that most real world HD functions do not suffer the curse because they have properties other than traditional smoothness
  - Sparsity : F is a sum of a small number of functions from a fixed basis/frame/dictionary
  - Anisotropy/Variable Reduction: not all variables are equally important - get rid of the weak ones
  - Tensor structures: variable separability
  - Superposition: F is a composition of functions of few variables - Hilbert's 13-th problem
  - Many new approaches based on these ideas: Manifold Learning; Laplacians on Graphs; Sparse Grids; Sensitivity Analysis; ANOVA Decompositions; Tensor Formats; Discrepancy; Deep Learning (Superposition)

Finding a good Subspace

- When building a linear numerical algorithm we have the option of choosing the subspace X_n
- Suppose K is our model class for the target function u and we measure error in a Banach space norm $\|\cdot\|_X$
- The best choice for X_n given the value of n is the n-width space - but it is generally impossible to find
- As a substitute we describe a method to find a space that is almost as good as the n-width space

The (Pure) Greedy Algorithm

- $f_0 := \operatorname{argmax}\{\|f\| : f \in \mathcal{K}\}$
- If $f_0, \dots, f_{n-1}$ have been chosen, define
  - $V_n := \operatorname{span}\{f_0, \dots, f_{n-1}\}$
  - $f_n := \operatorname*{Argmax}_{f \in \mathcal{K}} \operatorname{dist}(f, V_n)_X$
- Thus at each step, the function f_n is chosen in a greedy manner
- For the purposes of numerical implementation the algorithm is usually modified to a weak greedy algorithm
- This means that at each step we weaken the selection criteria: For a fixed $\gamma \in (0, 1]$, we choose f_n so that

$$\operatorname{dist}(f_n, V_n)_X \geq \gamma \sup_{f \in \mathcal{K}} \operatorname{dist}(f, V_n)_X$$

Performance

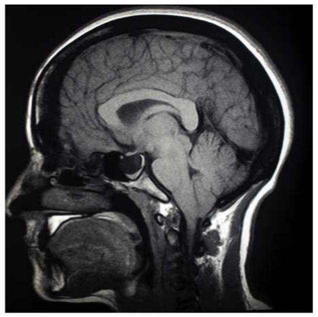
- Binev - Cohen - Dahmen - DeVore -Petrova -Wojtaszczyk prove the following theorem for the spaces $V_n, \ n \geq 1,$ generated by the weak greedy algorithm with parameter γ in the case X is a Hilbert space

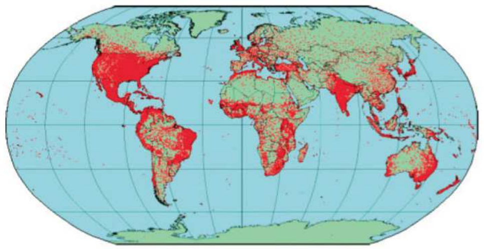
  Theorem: If $r > 0$, then there is a constant $C(r, \gamma) > 0$ such that whenever $d_n(K)_X \leq Mn^{-r}, \ n \geq 1$ we have $\mathrm{dist}(K, V_n)_X \leq C(r, \gamma)Mn^{-r}, \ n \geq 1$
- Finding the greedy space X_n requires querying K
- This is done by discretizing K to the accuracy of the current error
- In parametric PDEs the space X_n is referred to as model reduction

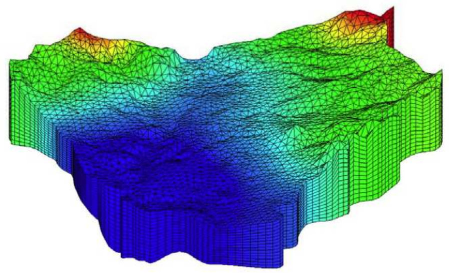
A Specific Task: Data Fitting

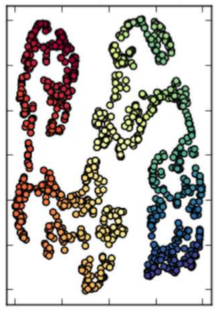
- We turn next to the following Common Scientific Problem: We are given data about some underlying function f (scientific process) and we wish to 'fit the data' to answer some question about f
- We put forward general principles that can be tailored to any specific application

Your Favorite Application


MRI


Global Temperatures


Groundwater Modeling


Manifold Learning

Data Tasks

- Two types of tasks
  - Prediction: Given any query x we want to compute $f(x)$
    - Since x is arbitrary, we need to approximate f
    - We call this the full appproximation problem
  - Quantity of Interest: calculate some narrower quantity
    - maximum/minimum of f
    - average behavior: calculate an integral of f
    - value of f at some designated point
- We seek algorithms that can be proven optimal for recovering f or answering questions of interest about f: optimal and certifiable performance

Mathematical Formulation

- Consider the full approximation problem for f
  - Form of the Data?: We assume $w_j = l_j(f), \quad j = 1, \dots, m$, where l_j are linear functionals
    - Measurement map $M(f) = w := (w_1, \dots, w_m)$
  - How to measure performance? We measure distortion by a norm $\|\cdot\|_X$ with X a Banach space
- An algorithm is a mapping $A : I\!\!R^m \mapsto X$ where $A(M(f))$ is an approximation to $f \in X$ giving error

$$E(f, A)_X := E(f, M, A)_X := \|f - A(M(f))\|_X$$

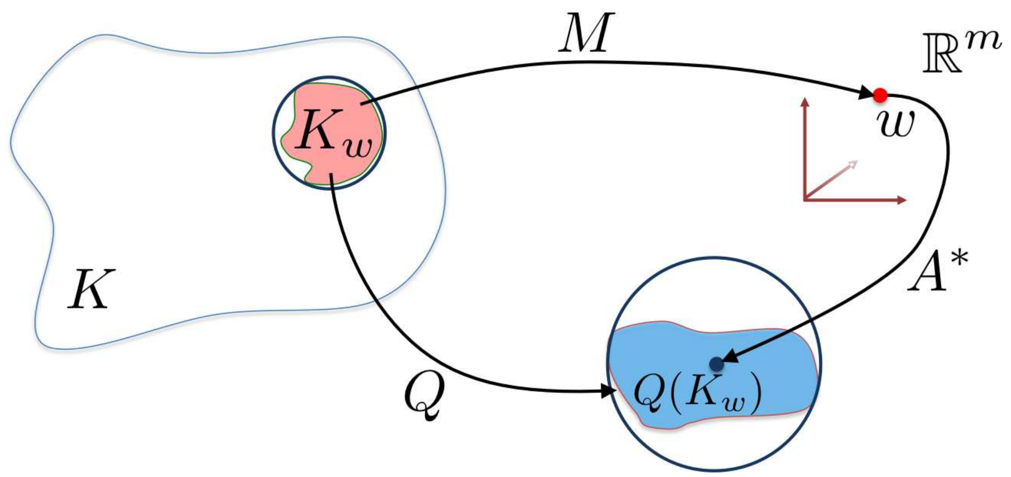
Model Classes

- With no other information we can say nothing about the error or discuss best algorithms
- To state a meaningful problem we need to have additional information about f
- This additional information is typically given in an assumption that $f \in \mathcal{K} \subset X$ with $\mathcal{K}$ a model class
- An accurate description of the model class $\mathcal{K}$ is the most important ingredient in data assimilation
- The more info we have on $\mathcal{K}$ the better we can do
- In scientific computation this is extracted by understanding the scientific process: for example, bandlimits for signals, regularity theorems for PDEs

Optimal Recovery: Best Algorithms

- $\mathcal{K}$, $\|\cdot\|_X$ fixed and consider any algorithm A
- Define $\mathcal{K}_w := \{f \in \mathcal{K} : M(f) = w\}$
  - Membership in $\mathcal{K}_w$ is all we know about f
- Pointwise error: $E(\mathcal{K}_w, M, A) := \sup_{f \in \mathcal{K}_w} \|f - A(w)\|_X$
- Worst case error:
  $E(\mathcal{K}, M, A) := \sup_{f \in \mathcal{K}} \|f - A(Mf))\|_X = \sup_{w \in \mathbb{R}^m} E(\mathcal{K}_w, M, A)$
- Optimal Performance: $\boxed{E^*(\mathcal{K}, M) := \inf_A E(\mathcal{K}, M, A)}$
- Optimal Recovery: The best algorithm A^*
  - Let $B(g_w, R_w)$ be the smallest ball that contains $\mathcal{K}_w$
  - $A^* : w \mapsto g_w$ is an algorithm that is pointwise optimal
    $E(\mathcal{K}_w, M, A^*)_X = E^*(\mathcal{K}_w, M) = R_w$

Graphic for Optimal Recovery



Not so Fast!

- You may think that this is the end of the story
  - But finding the Chebyshev ball is a substantial problem and is only carried out in certain special settings: for certain $\mathcal{K}$ and certain distortion metrics $\|\cdot\|_X$
  - Results where optimal recovery is known are summarized in Micchelli-Rivlin
- However, there is a general setting where we can determine optimal or near optimal algorithms and we can determine a priori the optimal performance
  - This setting will also expose when one has good data or bad data

Approximation Sets

- Remember! Any algorithm will be based on some form of approximation!
- Let $V = V_n$ be the functions used in the approximation: polynomials, neural nets, wavelets, sparse sums, etc.
- Since we have chosen V we think $\mathcal{K}$ is described by the fact it is well approximated by V
- Natural Model class: Approximation set:

$$\mathcal{K} := \mathcal{K}(\epsilon, V) = \{f : \operatorname{dist}(f, V)_X \le \epsilon\}$$

- We shall describe algorithms which are optimal over all ϵ and you do not need to know ϵ

Performance estimates

- Full approximation problem: Performance determined by V and null space $\mathcal{N} := \{f \in X : \ M(f) = 0\}$ via

$$\mu(\mathcal{N}, V) := \mu(\mathcal{N}, V)_X := \sup_{\eta \in \mathcal{N}} \frac{\|\eta\|}{\mathrm{dist}(\eta, V)}$$

- When X is a Hilbert space best performance for an approximation set $\mathcal{K} = \mathcal{K}(\epsilon, V)$ is

$$\boxed{E^*(\mathcal{K}, M) = \mu(\mathcal{N}, V)\epsilon}$$

- When X is a general Banach space best performance $E(\mathcal{K}, M)$ for an approximation set $\mathcal{K} = \mathcal{K}(\epsilon, V)$ satisfies

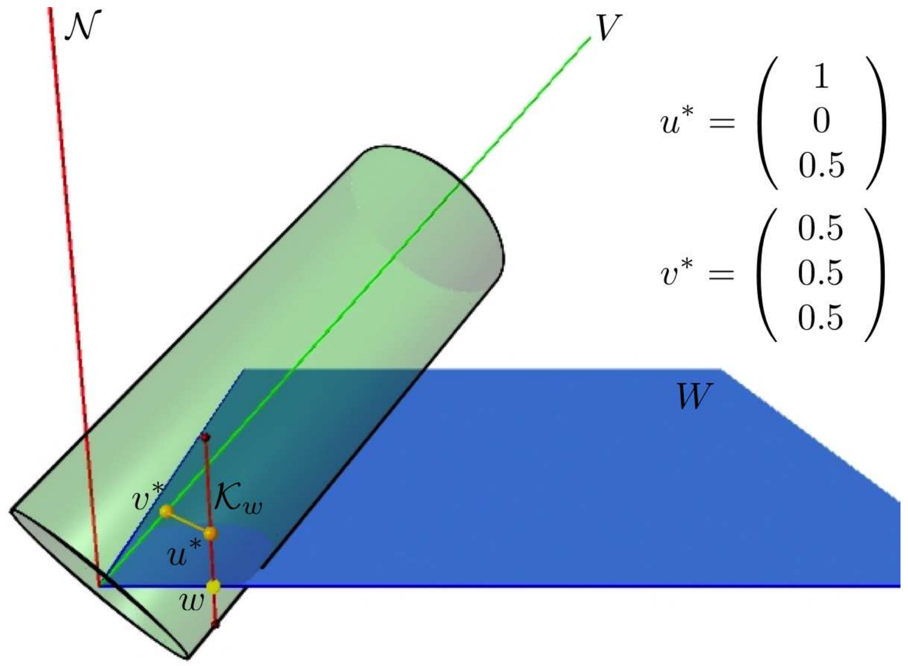
$$\mu(\mathcal{N}, V)\epsilon \leq E(\mathcal{K}, M) \leq 2\mu(\mathcal{N}, V)\epsilon$$

- Important: μ is easy to compute and (near) best algorithms can be described as will follow

A simple example

- Take X to be a Hilbert space
- If $l_j(f) = \langle f, \omega_j \rangle$, $j = 1, \dots, m$ with $(\omega_j)_{j=1}^m$ ONS, then
  - $v^*(w) := \underset{v \in V}{\operatorname{Argmin}} \|w - M(v)\|_{\ell_2}$
  - $A : w \mapsto v^*(w)$ is near optimal with constant 2
  - If $u^*(w) \in \mathcal{K}_w$ is the closest element $v^*(w)$ then $A^* : w \mapsto u^*(w)$ is linear and pointwise optimal
- Best algorithm is essentially least squares fit:
- μ can be computed by SVD of cross Grammian
- What is new?: Generally you do not see μ and performance estimates for least squares
- Note: Data is good if μ is small and bad if μ is large

Hilbert space geometry



$$u^* = \begin{pmatrix} 1 \\ 0 \\ 0.5 \end{pmatrix}$$

$$v^* = \begin{pmatrix} 0.5 \\ 0.5 \\ 0.5 \end{pmatrix}$$

Choosing V

- The above optimal estimates take the form $\|f - A(M(f)\|_X \le C\mu(\mathcal{N}, V)\,\mathrm{dist}(f, V)$
- Here there is a competition between μ and $\mathrm{dist}(f, V)$
  - Increasing the complexity of V improves $\mathrm{dist}(f, V)$ but increases $\mu(\mathcal{N}, V)$
- I want to illustrate this with a (toy) example
- $X = C(D)$ with D a domain in $\mathbb{R}^d$
- $w_j = l_j(f) = f(x_j)$ with $x_j \in D$, $j = 1, \dots, m$
- $V \subset C(D)$ a linear space of dimension $n \le m$
- $\mu(\mathcal{N}, V) = 1 + \mu(V, \mathcal{N})$ where

$$\mu(V, \mathcal{N}) = \sup_{v \in V} \frac{\|v\|_{C(D)}}{\max_{1 \le j \le m} |v(x_j)|}$$

Point values

- Near best algorithm is $v^* := \mathrm{Argmin}_{v \in V} \|w - M(v)\|_{\ell_\infty}$
- Example $X = C([0,1])$, $\xi_1, \dots, \xi_m$ equally spaced, $V = \mathcal{P}_{n-1}$ - polynomials of degree $< n$. Then it is known
  - If you choose $n = m$ then $\mu(\mathcal{N}, \mathcal{P}_m) \approx a^n, \ a > 1$
  - If $n = \sqrt{m}$ then $\mu(\mathcal{N}, \mathcal{P}_n) \le 3$
  - This gives $\|f - A(M(f))\|_C \le 3\,\mathrm{dist}(f, \mathcal{P}_{\sqrt{n}})_C$
- This points out the importance of the choice of V and the knowledge we have about f
- The above says do not interpolate unless you know f is very smooth - analytic
- Analogy with statistical learning: Do not overfit data
- computing μ tells you what overfit means

High dimension

- What happens when f depends on many variables/parameters: many features in data
    - The main issue is what is the correct model class $\mathcal{K}$ - what is the correct V to avoid the curse of dimensionality
    - Model classes $\mathcal{K}$ are proposed built on sparsity, anisotropy, variable reduction, feature selection, etc.
    - Typical V are built on highly nonlinear methods such as dictionary appproximation, neural networks
    - To have a quantitative theory (certifiable performance) we need to understand
        - Which functions are approximated well by V - if and only if theorems
        - What is $\mu(\mathcal{N}, V)$ for given data and V
        - Computational complexity of optimal algorithms

Additional Remarks

- The main references for the above are: Binev-Cohen-Dahmen-DeVore-Petrova-Wojtaszczyk (Hilbert space), DeVore -Petrova-Wojtaszczyk (Banach space)
- Closely related work emphasizing more the issue of stable computation is given by Adcock, Hansen, Shadrin, Trefethen, et al

Quantities of Interest

- A similar theory of optimal recovery exists for quantities of interest Q
- Performance now controlled by

$$\mu(\mathcal{N}, V, Q) := \mu(\mathcal{N}, V, Q)_X := \sup_{\eta \in \mathcal{N}} \frac{\|Q(\eta)\|}{\operatorname{dist}(\eta, V)}$$

- For any Banach space X we have the performance bounds

$$\boxed{\mu(\mathcal{N}, V, Q)\epsilon \leq E(Q, \mathcal{K}(\epsilon, V), M) \leq 2\mu(\mathcal{N}, V, Q)\epsilon}$$

Constructive Opt. Linear Algorithm

- When $\mathcal{K}$ is an approximation set and Q is a linear functional then one can find an optimal algorithm that is linear by constrained optimization:
- Let $\mathcal{L}_Q := \{l = \sum_{j=1}^m a_j l_j : \; l(v) = Q(v), \; v \in V\}$ and

$$l^* := \operatorname*{Argmin}_{l \in \mathcal{L}_Q} \|Q - l\|_{X^*} = \sum_{j=1}^m a_j^* l_j$$

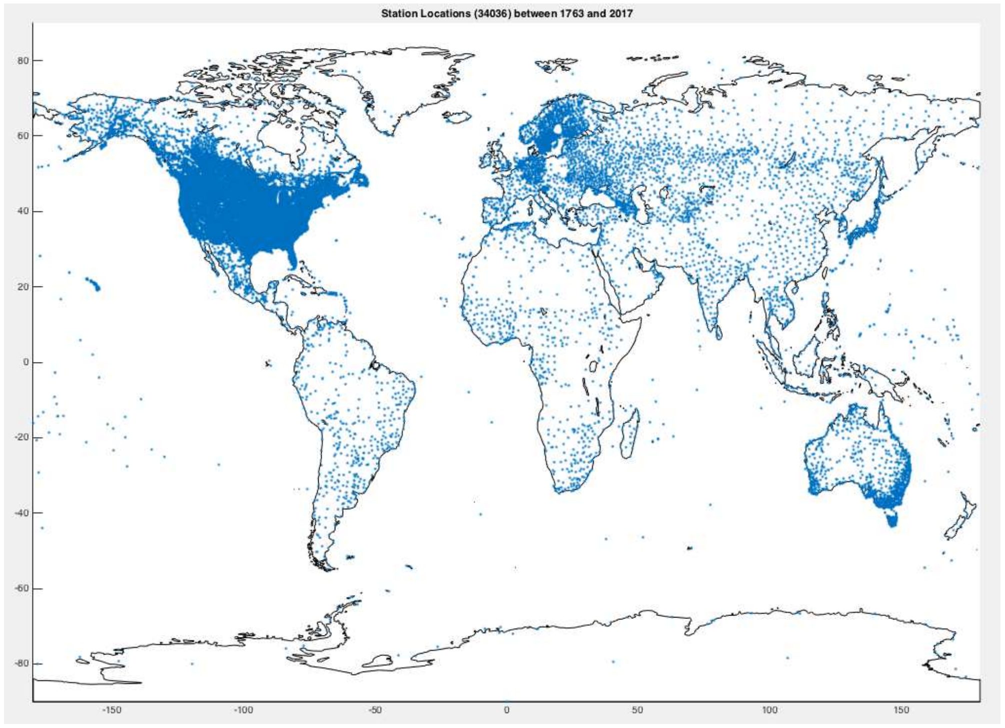
- Then $\boxed{A^* : w \mapsto \sum_{j=1}^m a_j^* w_j}$ is an optimal algorithm
- Note this may be numerically intensive constrained minimization
- Perf: $\boxed{|Q(f) - A^*(Mf)| \leq \|Q - l^*\|_{X^*} \operatorname{dist}(f, V)_X}$
- You see $\mu \leq \|Q - l^*\|_{X^*}$

Example: Quadrature

- Integration: Option trading, uncertainty quantification, Quasi-Monte Carlo, etc.
- Data are point values $l_j(f) = f(x_j),\ j = 1, \ldots, m$,
- We want to compute $Q(f) = \int_D \omega(x) f(x)\, dx,\ \ f \in \mathcal{K}(\epsilon, V)$
- The optimal quadrature on $X = C(D)$ using the points $x_j \in D$ is
  - $A^*(f) = \sum_{j=1}^m a_j^* f(x_j)$
  - $(a_j^*) := \mathrm{Argmin}\{\sum_{j=1}^m |a_j| :\ \sum_{j=1}^m a_j v(x_j) = \int_D \omega(x) v(x)\, dx\}$
- This is a constrained ℓ_1 minimization problem
- $\mu(\mathcal{N}, V, Q) = \sum_{j=1}^m |a_j^*|$
- $|\int f - A^*(M(f))| \leq \mu(\mathcal{N}, V, Q)\, \mathrm{dist}(f, V)_{C(D)}$

Example: Global Temperature

- Let $T(x,t)$ denote temperature at position x on earth and time t
- Quantity of interest $Q(T) = \int_{Year} \int_{Earth} T(x,t)\, dx\, dt$
- Roughly 14K sites from 1950 till 2017

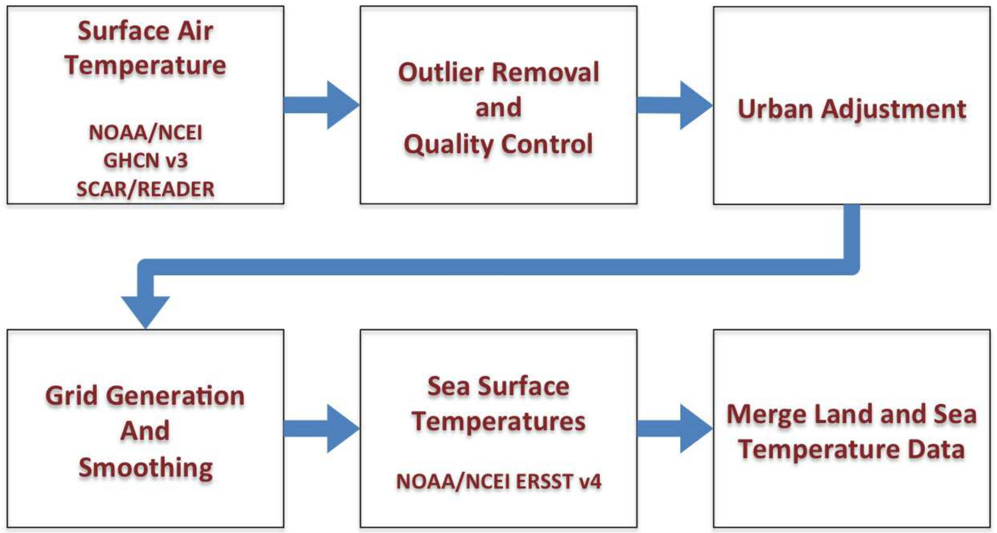


Obstacles to Mathematical Analysis

- Life would be good if
  - We knew the right model class for $T(x, t)$ - the right V
  - if data sites, equipment, and measuring times did not change each year
- Current algorithms use models based on pw polynomials - not clear what space
- We will use spherical harmonics
- We compare Spherical Harmonics versus GISTemp (NASA) on their adjusted data set
- We can compute μ for spherical harmonics but not for GISTemp

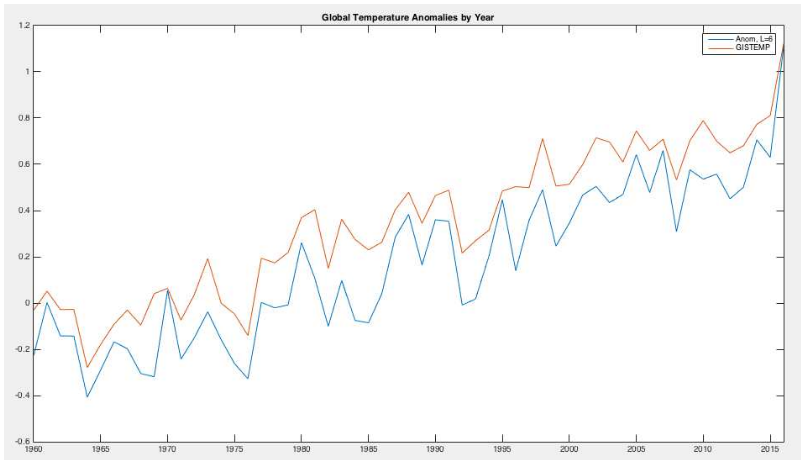
Current Algorithms

- There are many algorithms
- The following flowchart gives the main steps of the NOAA and NASA algorithms using piecewise polynomials on a uniform grid

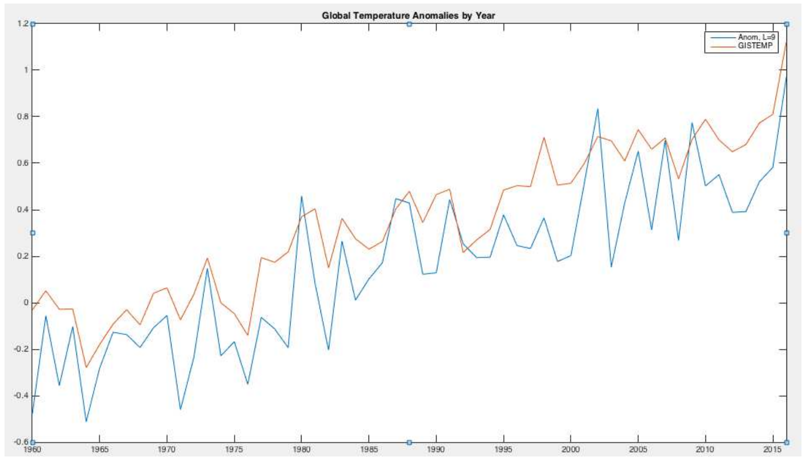


- Impossible to analyze accuracy because of the ad hoc adjustments to the data

Comparison:GISTempvs. SH6



Comparison: GISTemp vs. SH9



Typical Growth of μ

- Are we computing global temperature?
  - This would require proving validity of our model class: would require analysis from physical principles
  - Also depends on behavior of μ

n	3	6	9	12	15	18
μ	1	1.03	2.61	24.13	223.50	2779.85

- We see that even if we justify our model class, we need to restrict the size of n