Greedy Sparse Linear Approximations of Functionals from Nodal Data

R. Schaback

Göttingen Academy of Sciences and Humanities
Georg–August–Universität Göttingen
Institut für Numerische und Angewandte Mathematik
http://www.num.math.uni-goettingen.de/schaback/research/group.html

MAIA 2013, Erice, Italy, Sept. 2013

Overview

Linear Recovery

Optimal Recovery

Greedy Recovery

Examples

Linear Recovery

Standard Problem of Numerical Analysis

- You have data, but you want different data
- You have a few data of an otherwise unknown function, but you want different data of that function
- Examples:
  - Numerical integration
  - Numerical differentiation
  - PDE solving
- Given $\lambda_1(u), \ldots, \lambda_N(u)$, find $\mu(u)$ for continuous linear functionals $\mu, \lambda_1, \ldots, \lambda_N \in U^*$
- Assumption: Linear recovery

$$\mu(u) \approx \sum_{j=1}^{N} a_j \lambda_j(u) \quad \text{for all } u \in U$$

Linear Recovery

- Linear recovery

$$\mu(u) \approx \sum_{j=1}^{N} a_j \lambda_j(u) \quad \text{for all } u \in U$$

- Error functional

$$\epsilon_a(u) := \mu(u) - \sum_{j=1}^{N} a_j \lambda_j(u) \quad \text{for all } u \in U,\ a \in \mathbb{R}^N$$

- Sharp error bound

$$\left| \mu(u) - \sum_{j=1}^{N} a_j \lambda_j(u) \right| \leq \underbrace{\|\epsilon_a\|_{U^*}}_{????} \cdot \|u\|_u$$

- If $\|\epsilon_a\|_{U^*}$ were known: Error known in % of $\|u\|_u$
- Problem: Calculate $\|\epsilon_a\|_{U^*}$ and optimize over a

Reproducing Kernel Hilbert Spaces

- Assumption: U Hilbert space of functions on Ω with continuous and linearly independent point evaluations
- Theorem: Then u is a RKHS with some positive definite kernel $K \; : \; \Omega \times \Omega \to \mathbb{R}$

$$\begin{array}{rcl} u(x) & = & (u, K(x,\cdot))_U \text{ for all } x \in \Omega,\; u \in U \\ \mu(u) & = & (u, \mu^x K(x,\cdot))_U \text{ for all } \mu \in U^*,\; u \in U \\ (\lambda,\mu)_{U^*} & = & \lambda^x \mu^y K(x,y) \text{ for all } \lambda,\; \mu \in U^* \end{array}$$

- Consequence:

$$\begin{array}{rcl} \|\epsilon_a\|^2_{U^*} & = & \epsilon_a^x \epsilon_a^y K(x,y) \\ & = & \mu^x \mu^y K(x,y) - 2\sum_{j=1}^N a_j \; \mu^x \lambda_j^y K(x,y) \\ & & + \sum_{j=1}^N \sum_{k=1}^N a_j a_k \lambda_j^x \lambda_k^y K(x,y) \end{array}$$

- This quadratic form in a can be explicitly calculated
- Different approximations can be compared
- This quadratic form can be optimized

Sobolev Spaces

- Assumption: U Hilbert space of functions on Ω with continuous and linearly independent point evaluations
- Take $U := W_2^m(\mathbb{R}^d)$ with $m > d/2$
- Matérn–Sobolev kernel:

$$K(x,y) = \|x-y\|_2^{m-d/2} K_{m-d/2}(\|x-y\|_2)$$

  with modified Bessel function of second kind.
- For "nice" domains $\Omega \subset \mathbb{R}^d$: $W_2^m(\Omega)$ is norm–equivalent to $W_2^m(\mathbb{R}^d)$

Example: Numerical Integration in Sobolev Spaces

- 5 points in $[-1, 1]$
- Standard weights

Points	$\|\epsilon\|_1^2$	$\|\epsilon\|_2^2$	$\|\epsilon\|_3^2$
equidistant	0.103077	0.001034817	0.00016066
Gauß	0.090650164	0.000409353	0.00000298

Optimal Recovery

Optimal Recovery in RKHS

- Minimize the quadratic form

$$
\begin{aligned}
\|\epsilon_a\|_{U^*}^2 &= \epsilon_a^x \epsilon_a^y K(x,y) \\
&= \mu^x \mu^y K(x,y) - 2\sum_{j=1}^N a_j\, \mu^x \lambda_j^y K(x,y) \\
&\quad + \sum_{j=1}^N \sum_{k=1}^N a_j a_k \lambda_j^x \lambda_k^y K(x,y)
\end{aligned}
$$

- Solution a^* satisfies linear system

$$\sum_{j=1}^N a_j^* \lambda_j^x \lambda_k^y K(x,y) = \mu^x \lambda_k^y K(x,y),\ 1 \le k \le N$$

- Then

$$\|\epsilon_{a^*}\|_{U^*}^2 = \mu^x \mu^y K(x,y) - \sum_{j=1}^N a_j^*\, \mu^x \lambda_j^y K(x,y)$$

Connection to Interpolation

- Theorem: The optimal recovery is equal to the μ–value of the interpolant of the data $\lambda_j(u),\ 1 \leq j \leq N$ on the span of the functions $\lambda_j^x K(x, \cdot),\ 1 \leq j \leq N$.
- Proof steps:
- There is a Lagrange basis $u_1, \ldots, u_N$ of that span with

$$\lambda_j(u_k) = \delta_{jk},\ 1 \leq j, k \leq N$$

- The interpolant of data $f_j = \lambda_j(u)$ is

$$\begin{aligned} \tilde{u}(x) &= \sum_{j=1}^N u_j(x) f_j = \sum_{j=1}^N u_j(x)\lambda_j(u) \\ \mu(u) \approx \mu(\tilde{u}) &= \sum_{j=1}^N \mu(u_j) f_j = \sum_{j=1}^N \mu(u_j)\lambda_j(u) \end{aligned}$$

- Then, as a recovery formula,

$$a_j = \mu(u_j),\ 1 \leq k \leq N$$

Connection to Interpolation II

- Optimality?
- We need

$$\sum_{j=1}^{N} \mu(u_j)\lambda_j^x \lambda_k^y K(x,y) = \mu^x \lambda_k^y K(x,y),\ 1 \leq k \leq N$$

The Lagrange property implies

$$\sum_{j=1}^{N} u_j(x)\lambda_j^x \lambda_k^y K(x,y) = \lambda_k^y K(x,y),\ 1 \leq k \leq N$$

and application of μ proves optimality.

Example: Optimal Integration in Sobolev Spaces

- 5 points in $[-1, 1]$

Points	Method	$\|\epsilon\|^2_{W_2^1(\mathbb{R})^*}$	$\|\epsilon\|^2_{W_2^2(\mathbb{R})^*}$	$\|\epsilon\|^2_{W_2^3(\mathbb{R})^*}$
equi	standard	0.103077	0.00103481	0.00016066
equi	optimal	0.101896	0.00066343	0.00001082
Gauß	standard	0.090650	0.00040935	0.00000298
Gauß	optimal	0.085169	0.00040768	0.00000296
optimal	optimal	0.063935	0.00019882	0.00000106

- Optimal points are chosen to minimize the error norm of the optimal recovery in Sobolev space

Summary of Recovery

- Consider linear recoveries of unknown data from known data
- Do this on a RKHS
- Evaluate the norm of the error functional
- This allows fair comparisons between recoveries
- Optimize the recovery weights
- Goal: Optimize the choice of data points
- Goal: Do this efficiently

Greedy Recovery

Greedy Recovery

Specialize to recovery from nodal data:

$$\mu(u) \approx \sum_{j=1}^{N} a_j^* u(x_j) \quad \text{for all } u \in U$$

with optimal weights a_j^*, but vary the points x_j

Optimization of error norm in some RKHS,
e.g. Sobolev space $W_2^m(\mathbb{R}^d),\ m > d/2$

Note: $a_j^* = a_j^*(x_1, \ldots, x_N)$

Goal: Greedy recursive method, $N \mapsto N + 1$

Goal: Sparsity via greedy selection

Interpolation

For a Lagrange basis $u_1^N, \ldots, u_N^N$: interpolant to u is

$$s_u(x) = \sum_{j=1}^{N} u_j^N(x) u(x_j) \quad \text{for all } u \in U$$

Optimal recovery is

$$\mu(u) \approx \mu(s_u) = \sum_{j=1}^{N} \underbrace{\mu(u_j^N)}_{=:a_j^*} u(x_j) \quad \text{for all } u \in U$$

Drawback: Lagrange basis is not efficiently recursive

Goal: Use Newton basis recursively

Recursive Newton Basis

Points $x_1, x_2, \dots, x_k, x_{k+1}, \dots$

Recursive Kernels (RS, 1997)

$$
\begin{aligned}
K_0(x,y) &:= K(x,y) \\
K_{k+1}(x,y) &:= K_k(x,y) - \frac{K_k(x_{k+1},x)K_k(x_{k+1},y)}{K_k(x_{k+1},x_{k+1})}
\end{aligned}
$$

Newton basis functions (St. Müller/RS, 2009)

$$
\begin{aligned}
v_{k+1}(x) &:= \frac{K_k(x_{k+1},x)}{\sqrt{K_k(x_{k+1},x_{k+1})}} \\
K_{k+1}(x,y) &= K(x,y) - \sum_{j=1}^{k+1} v_j(x)v_j(y) \\
&= K_k(x,y) - v_{k+1}(x)v_{k+1}(y)
\end{aligned}
$$

Properties of Newton Basis

Orthonormality $(v_j, v_k)_U = \delta_{jk}$

Zeros $v_{k+1}(x_j) = 0$, $1 \leq j \leq k$

Power Function for interpolation:

$$
\begin{aligned}
P^2_{k+1}(\delta_x) &:= \|\epsilon(\delta_x; \alpha^*_1(\delta_x), \ldots, \alpha^*_{k+1}(\delta_x))\|^2_{H^*} \\
&= K_{k+1}(x, x) \qquad (M.Mouattamid, \; RS \; 09) \\
&= K_k(x, x) - v^2_{k+1}(x) \\
&= P^2_k(\delta_x) - v^2_{k+1}(x)
\end{aligned}
$$

Recursive Recovery

Functional μ, points $x_1, x_2, \dots, x_k, x_{k+1}, \dots$

Recursive Equations I

$$
\begin{aligned}
\mu^x K_0(x,y) &= \mu^x K(x,y) \\
\mu^x K_{k+1}(x,y) &= \mu^x K_k(x,y) - \frac{\mu^x K_k(x_{k+1},x) K_k(x_{k+1},y)}{K_k(x_{k+1},x_{k+1})} \\
\mu^y \mu^x K_{k+1}(x,y) &= \mu^y \mu^x K_k(x,y) - \frac{\mu^x K_k(x_{k+1},x) \mu^y K_k(x_{k+1},y}{K_k(x_{k+1},x_{k+1})} \\
&= \mu^y \mu^x K_k(x,y) - \mu(v_{k+1})^2 \\
P^2_{k+1}(\mu) &:= \|\epsilon(\mu; \alpha^*_1(\mu), \dots, \alpha^*_{k+1}(\mu))\|^2_{H^*} \\
&= \mu^x \mu^y K_{k+1}(x,y) \\
&= P^2_k(\mu) - \mu(v_{k+1})^2 \\
&= \mu^x \mu^y K(x,y) - \sum_{j=1}^{k+1} \mu(v_j)^2.
\end{aligned}
$$

Goal: Choose x_{k+1} to maximize $\mu(v_{k+1})^2$

Recursive Recovery II

Functional μ, points $x_1, x_2, \dots, x_k, x_{k+1}, \dots$

Goal: Choose x_{k+1} to maximize $\mu(v_{k+1})^2$

Recursive Equations

$$
\begin{aligned}
v_{k+1}(x) &:= \frac{K_k(x_{k+1}, x)}{\sqrt{K_k(x_{k+1}, x_{k+1})}} \\
\mu(v_{k+1}) &= \frac{\mu^x(K_k(x_{k+1}, x))}{\sqrt{K_k(x_{k+1}, x_{k+1})}}
\end{aligned}
$$

Maximize as function of z:

$$
R_k(z) := \frac{(\mu^x K_k(z, x))^2}{K_k(z, z)}
$$

Stop if R_k is small on available points

Problems: Efficiency? Stability?

Implementation: Overview

Total given points: $x_1, \dots, x_N$

Goal: Select a small subset of n points greedily

Computational Complexity: $\mathcal{O}(nN)$ for update $n - 1 \to n$

Computational Complexity: $\mathcal{O}(n^2 N)$ in total

Storage: $\mathcal{O}(nN)$ for n Newton basis functions

Similarly for interpolation of n data with
evaluation on N points via Newton basis

Implementation I

Total given points: $x_1, \dots, x_N$

Goal: Select a small subset greedily

Index set $I := \emptyset$ to collect selected point indices

Various N-vectors for steps $k = 0, 1, \dots, N$:

$$\begin{array}{rcl}
\mathbf{K}_k & := & (K_k(x_j, x_j))_{1 \le j \le N} \\
\mathbf{M}_k & := & (\mu^x K_k(x, x_j))_{1 \le j \le N} \\
\mathbf{R}_k & := & (R_k(x_1), \dots, R_k(x_N))^T = \mathbf{M}_k \;.\!\wedge 2 \;./\; \mathbf{K}_k
\end{array}$$

Easy initialization for $k = 0$

Find maximum of $\mathbf{R}_0$. insert point index into I

Further N-vectors :

$$\begin{array}{rcl}
\mathbf{k}_1 & := & (K_0(x_{I(1)}, x_j))_{1 \le j \le N} \\
\mathbf{V}_1 & := & \mathbf{k}_1./.\sqrt{\mathbf{K}_0}
\end{array}$$

Implementation II

Recursion $n \Rightarrow n+1$:

Given: N–vectors $\mathbf{K}_n$, $\mathbf{M}_n$, $\mathbf{V}_1, \ldots, \mathbf{V}_n$,
Index set I with n point indices

Maximize $\mathbf{R}_n = \mathbf{M}_n\ .^{\wedge 2}\ ./\ \mathbf{K}_n$, if max. is not too small
and add new point index into$I$

Now I is $I = \{y_1, \ldots, y_{n+1}\}$. Use

$$
\begin{aligned}
K_n(y_{n+1}, x_j) &= \underbrace{K(y_{n+1}, x_j)} - \sum_{k=1}^{n} v_k(y_{n+1}) v_k(x_j), \\
\mathbf{k}_{n+1} &= \overbrace{\mathbf{k}_{n+1,0}} - \sum_{k=1}^{n} e^T_{I(n+1)} \mathbf{V}_k \mathbf{V}_k \\
\mathbf{V}_{n+1} &:= \mathbf{k}_{n+1}./\sqrt{\mathbf{K}_n} \\
\mathbf{K}_{n+1} &= \mathbf{K}_n - \mathbf{V}_{n+1}.^{\wedge 2}
\end{aligned}
$$

Implementation III

Update of $\mathbf{M}_{n+1} = (\mu^x K_{n+1}(x, x_j))_{1 \le j \le N}$:

Use

$$\begin{aligned} \mu^x K_{n+1}(x, x_j) &= \mu^x K_n(x, x_j) - \frac{\mu^x K_n(y_{n+1}, x) K_n(y_{n+1}, x_j)}{K_n(y_{n+1}, y_{n+1})} \\ \mathbf{M}_{n+1} &= \mathbf{M}_n - \frac{\mathbf{M}_{n,I(n+1)}}{\mathbf{K}_{n,I(n+1)}} \mathbf{k}_{n+1} \end{aligned}$$

To get the error norm:

Update $P_{n+1}^2(\mu) = P_n^2(\mu) - R_n^2(y_{n+1})$
starting from $P_0^2(\mu) := \mu^x \mu^y K(x, y)$

Implementation IV

No recursion on the weights, so far.

After n steps:

$\tilde{\mathbf{M}}_0$:= truncation of N–vector $\mathbf{M}_0$ to the selected n points

$\tilde{\mathbf{V}}$:= truncation of matrix $(\mathbf{V}_1, \ldots, \mathbf{V}_n)$ to selected n points

Then solve two $n \times n$ triangular systems

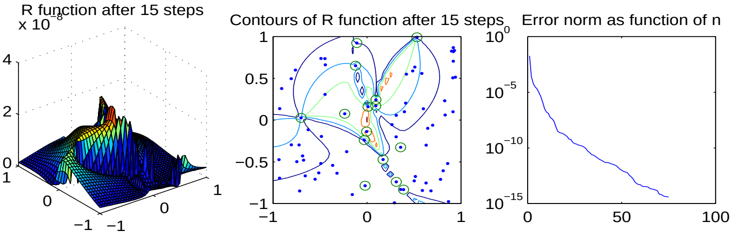
$$\begin{aligned} \tilde{\mathbf{V}}\tilde{\mathbf{z}} &= \tilde{\mathbf{M}}_0 \\ \tilde{\mathbf{V}}^T\tilde{\mathbf{a}} &= \tilde{\mathbf{z}} \end{aligned}$$

for the vector $\mathbf{a}$ of the n optimal weights.

Examples

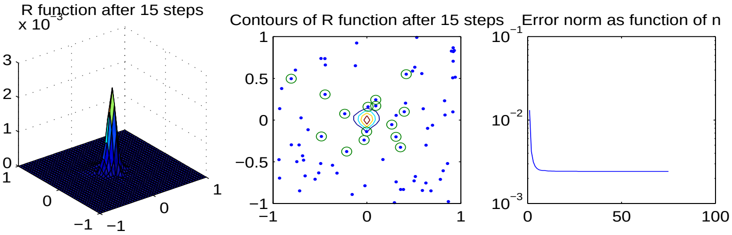
Interpolation I

Interpolation at origin
Offering 75 scattered points
Taking 15 optimal points
Gaussian kernel



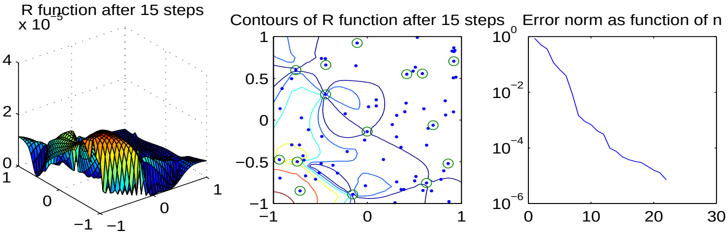
Interpolation II

Interpolation at origin
Offering 75 scattered points
Taking 15 optimal points
C^2 Wendland kernel



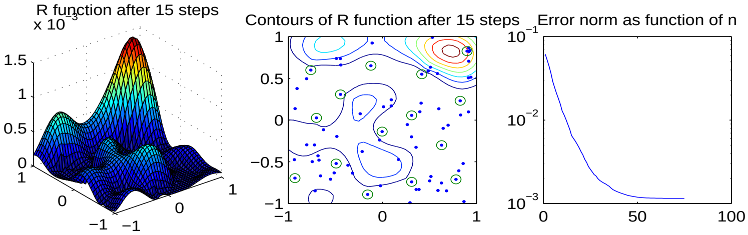
Integration I

Integration over $[-1, +1]$
Offering 75 scattered points
Taking 15 optimal points
Gaussian kernel



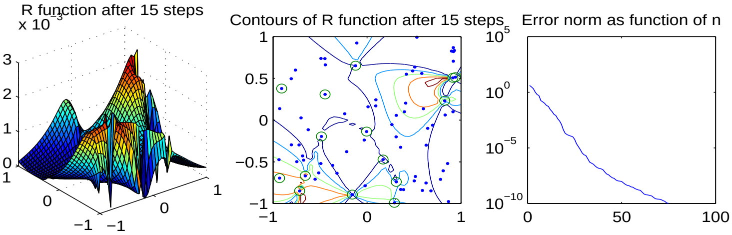
Integration II

Integration over $[-1, +1]$
Offering 75 scattered points
Taking 15 optimal points
C^2 Wendland kernel



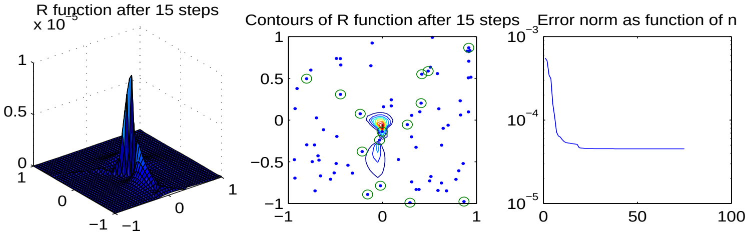
Laplacian I

Greedy recovery of Δu at the origin
Offering 75 scattered points
Taking 15 optimal points
Gaussian kernel



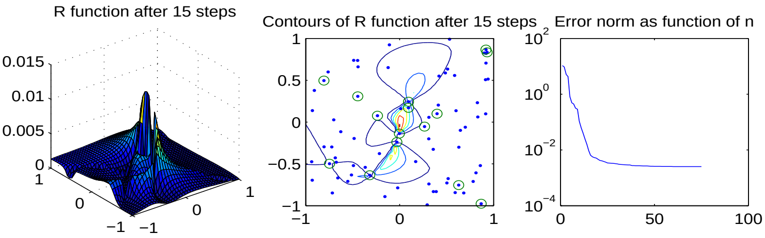
Laplacian II

Greedy recovery of Δu at the origin
Offering 75 scattered points
Taking 15 optimal points
C^4 Wendland kernel



Laplacian III

Greedy recovery of Δu at the origin
Offering 75 scattered points
Taking 15 optimal points
Sobolev–Matérn kernel $r^5 K_5(r)$



Thank You!

For references, see

http://www.num.math.uni-goettingen.de/schaback
and in particular
.../research/papers/GSLAoFfND.pdf

Thank You!

For references, see

http://www.num.math.uni-goettingen.de/schaback