LEAST SQUARES AND SMOLYAK'S ALGORITHM

JAKOB EGGL, ELIAS MINDLBERGER AND MARIO ULLRICH

ABSTRACT. We present novel, large-scale experiments for polynomial interpolation in high dimensional settings using some of the most popular algorithms available. We compare Smolyak's Algorithm (SA) on sparse grids and Least Squares (LSQ) using random data. We empirically confirm that interpolation using LSQ performs equally as good on smooth and better on non-smooth functions if SA is given n and LSQ is given $\mathcal{O}(n \log n)$ points.
Code available at: `https://github.com/th3lias/NumericalExperiments`

1. INTRODUCTION

Smolyak's Algorithm on Sparse Grids has been of high theoretical and practical interest for a long time.

2. NOTATION

We denote the indexset containing all indices $i \in \mathbb{Z}$ where $m \leq i \leq n$ for $m \leq n$ with $[m : n]$. For the set of polynomials $p \colon V \to W$ of maximal degree N and sets $V, W \subseteq \mathbb{C}^K$, $K \in \mathbb{N}$ we use the notation $p \in \mathcal{P}^N(V, W)$. **_I mean this is the general notation but do we even have polynomials that map to $\mathbb{C}^K$._** We use $f \propto g$ for functions f, g to denote $f = cg$ for a constant $c \in \mathbb{R}$ and $f \lesssim g$ to denote $f \leq cg$. With $B(X)$ we denote the closed unit ball of a normed space X, i.e. $B(X) := \{x \in X : \|x\|_X \leq 1\}$. With $X^\star$ we denote the space of linear, bounded functionals on X, i.e. $X^\star := \{x^\star \colon X \to \mathbb{R} \mid x^\star \text{ is linear and bounded}\}$.

3. SMOLYAK'S ALGORITHM

Smolyak's Algorithm is determinstic and can be used in various settings. These include but are not limited to

(1) function interpolation,
(2) numerical integration,
(3) solving ODEs and PDEs,

(4) $\cdots$

Interpolation in one dimension. In the onedimensional setting it is well known that for a given function $f \colon \mathbb{R} \to \mathbb{R}$ and a point collection $\mathbf{Z} := \{z_n\}_{n=0}^N \subseteq \mathbb{R}$, containing of $N + 1$ different points, there exists an interpolating polynomial $i \in \mathcal{P}^N(\mathbb{R}, \mathbb{R})$ such that for all $j \in [0 : N]$ we have $i(z_j) = f(z_j)$. To find the polynomial i, one can, among other available methods, choose to perform Lagrange Interpolation which yields an explicit construction of the form

$$(3.1) \qquad \mathcal{I}(\mathbf{F})(x) := \sum_{n=0}^N f(z_n)\ell_n(x) \quad \text{s.t.} \quad \ell_n(x) := \prod_{\substack{j=0 \\ j \neq n}}^N \frac{x - z_j}{z_n - z_j}$$

where $\mathbf{F} := \{[z_n, f(z_n)]\}_{n=0}^N$. We set $i := \mathcal{I}(\mathbf{F})$. Note that $\mathcal{I}$ is a mapping from data, which may be seen as arranged in a $2 \times N$ matrix, to an N-degree polynomial. Thus $\mathcal{I} \colon \mathbb{R}^{2 \times N} \to \mathcal{P}^N(\mathbb{R}, \mathbb{R})$, since each basis function $\ell_n(x)$ is a polynomial of degree N. Therefore, if $f \in \mathcal{P}^N(\mathbb{R}, \mathbb{R})$, one obtains $i = f$. Moreover, i is interpolating since $\ell_n(z_j) = \delta_n(j)$.

Even though (3.1) has nice theoretical properties, it is suboptimal for practical settings due to the following facts. (1) If a node z_n is close to a node z_j such that $j \neq n$, computing the product $\ell_n(x)$ becomes numerically unstable. (2) The addition of new data $(z, f(z))$ requires to recalculate all basis functions again in $\mathcal{O}(N^2)$ time. (3) To compute $i(x)$ in this form requires computational complexity of $\mathcal{O}(N^2)$. This does not mean that one has to abandon Lagrange Interpolation at once. We may write (3.1) in a much more stable way as

$$(3.2) \qquad i(x) := \mathcal{I}(\mathbf{F})(x) = \frac{\sum_{n=0}^N \frac{w_n}{x - z_n} f(z_n)}{\sum_{n=0}^N \frac{w_n}{x - z_n}}$$

for $w_n \in \mathbb{R} : n \in [0 : N]$. If the weights w_n are known, computing $i(x)$ in this form admits a complexity of $\mathcal{O}(N)$. Luckily, the weights have a closed-form, analytic solution for many deterministic point sets used in practice and are hence considered to be computable in $\mathcal{O}(1)$ time. Thus, adding a new data pair $(z, f(z))$ requires total complexity of $\mathcal{O}(N)$ for the recomputation of said weights. The general identity

$$w_n = \frac{1}{\ell'(x_n)}$$

always holds. A survey on the *barycentric* form of Lagrange interpolation can be found in [**Berut˙2004**]. To specify the above, consider the

set of *equidistant* nodes $\{z_n := 2/n\}_{n=0}^{N} \subset [-1, 1]$. Then

$$(3.3) \qquad w_n = (-1)^n \binom{N}{n}.$$

For large N, (3.3) is problematic since weights $w_i, w_j : i \neq j$ now may vary by factors as large as $\mathcal{O}\left(2^N\right)$. This means that interpolation in equispaced points is susceptible to the *Runge phenomenon*. For this interpolation problem to be well posed one should use point sets with asymptotic density $\rho(x) \propto 1/\sqrt{1 - x^2}$. This is called the *Chebyshev* density.

Many point sets admit this asymptotic density, for example the Chebyshev points of the first kind are given as the roots of the $n + 1$-th Chebyshev polynomial and thus given by

$$(3.4) \qquad \mathbf{Z}_C^{(1)}(N) := \left\{ \cos \frac{(2n + 1)\pi}{2N + 2} \mid n \in [0 : N] \right\}.$$

Similarly, the Chebyshev points of the second kind are defined

$$(3.5) \qquad \mathbf{Z}_C^{(2)}(N) := \left\{ \cos \frac{n\pi}{N} \mid n \in [0 : N] \right\}.$$

In either of these cases, the weights w_n admit nice closed forms as $w_n^{(k)} : k \in \{1, 2\}$ dependent on which point set is in use as

$$(3.6) \quad w_n^{(1)} = (-1)^n \sin \frac{(2n + 1)\pi}{2n + 2}, \quad w_n^{(2)} = (-1)^n \cdot \begin{cases} 1/2 & n \in \{0, N\} \\ 1 & \text{else.} \end{cases}$$

Interpolation in d dimensions. In principle, one may just *extend* one dimensional interpolation rules to d dimensions by forming the tensor product of all interpolation rules. To specify, take the one dimensional algorithm

$$(3.7) \qquad \mathcal{I}^{(N)}(\mathbf{F}(N))(x) = \sum_{n=1}^{m(N)} b_n(x) f(z_n)$$

such that $\mathcal{I}^{(N)}(\mathbf{F}(N))$ is interpolating on the data $\mathbf{F}(N) := \{[z_n, f(z_n)]\}_{n=1}^{m(N)}$ with basis functions $b_n \in C[0, 1]$ by having $b_n(z_j) = \delta_n(j)$. Note that $m \colon \mathbb{N}_0 \to \mathbb{N}_0$ is a common abstraction for the number of points used to interpolate a given function. In the case of special interpolation methods or their corresponding underlying point set, m will be specified.

To obtain a set of d-dimensional point sets from d such one-dimensional

rules like (3.4) or (3.5), one uses the usual cartesian product
(3.8)
$$\mathbf{Z}(N, d) := \mathbf{Z}(N_1) \times \cdots \times \mathbf{Z}(N_d) := \bigcup_{n_1=1}^{m(N_1)} \cdots \bigcup_{n_d=1}^{m(N_d)} \left\{ \left(z_{n_1}^{(1)}, \ldots, z_{n_d}^{(d)} \right) \right\}$$

where $z_j^{(i)}$ denotes the j-th point of the i-th rule and $N := (N_1, \ldots, N_d) \in \mathbb{N}_0^d$ and $z_n := (z_{n_1}, \ldots, z_{n_d})$ as the usual notation for multi-indices. Using the tensor product formula, one obtains the very general expression

$$(3.9) \quad \left(\mathcal{I}^{(N_1)} \otimes \cdots \otimes \mathcal{I}^{(N_d)} \right)(\mathbf{F}) = \sum_{n_1=1}^{m(N_1)} \cdots \sum_{n_d=1}^{m(N_d)} (b_{n_1} \otimes \cdots \otimes b_{n_d}) f(z_n)$$

where $\mathbf{F} := \bigcup_{j=1}^{d} \mathbf{F}(N_j)$. In this case, the tensor product interpolation is a mapping of the form

$$(3.10) \quad \bigotimes_{j=1}^{d} \mathcal{I}^{(N_j)} \colon \mathbb{R}^{d \times \left(\prod_{j=1}^{d} m(N_j) \right)} \to \mathcal{P}^{(\max\{m(N_j)\})} \left(\mathbb{R}^d, \mathbb{R} \right).$$

We denote $\mathcal{I}^{(N)} := \bigotimes_{j=1}^{d} \mathcal{I}^{(N_j)}$. If the basis functions b_n are scalar-valued, their tensor product is just

$$(3.11) \quad \left(\bigotimes_{n=1}^{d} b_n \right) (x_1, \ldots, x_d) := \prod_{n \in [d]} b_n(x_n).$$

In Smolyak's Algorithm, the function $m \colon \mathbb{N} \to \mathbb{N}_0$ is used as a growth rule specifying how many points are to be used by the interpolant. Specifically, we will use a doubling rule such that

$$m(n) := \begin{cases} 2^{n-1} + 1 & n \geq 2 \\ 1 & \text{else.} \end{cases}$$

Clearly, the usual representation of the Lagrange polynomial (3.1) fits this framework. For the barycentric form, see [**Berut˙2004**], one obtains the d-variate interpolation

$$(3.12) \quad \mathcal{I}(\mathbf{F})(x) := \frac{\sum_{n_1=1}^{m(N_1)} b_{n_1}^{(1)}(x_1) \cdots \sum_{n_d=1}^{m(N_d)} b_{n_d}^{(d)}(x_d) f(z_{n_1}, \ldots, z_{n_d})}{\sum_{n_1=1}^{m(N_1)} b_{n_1}^{(1)}(x_1) \cdots \sum_{n_d=1}^{m(N_d)} b_{n_d}^{(d)}(x_d)}$$

where

$$(3.13) \quad b_{n_k}^{(k)}(x) := \frac{w_{n_k}}{x - z_{n_k}} \mid k \in [d].$$

If a rule yielding admissible point sets, like Chebyshev's rules (3.4) or (3.5), is available, the only thing left to do is to specify how many nodes $m(N_j)$ to pick in each dimension. If this number is large simultaneously

for all dimensions, (3.9) becomes intractable very quickly. Smolyak's algorithm defines a resolution-like number $q \in \mathbb{N}$ to specify how close-meshed points should be picked. It proceeds by taking all multi-indices N with 1-norm less equal to q and specifies to interpolate with the algorithm

$$(3.14) \qquad \mathcal{A}(q, d)\,(\mathbf{X}) := \sum_{\|N\|_1 \leq q} \mathcal{I}^{(N)}(\mathbf{X}(N)).$$

Note that, with this construction $q \geq d$ is necessary. The set $\mathbf{X}$ in this case is

$$(3.15) \qquad \mathbf{X}(N) := \bigcup_{n \in [d]} \mathbf{F}(N_n)$$

All points used by this algorithm are then

$$(3.16) \qquad \mathbf{X} := \mathbf{X}^{\leq}(q) := \bigcup_{\|N\|_1 \leq q} \mathbf{X}(N).$$

We call $\mathbf{X}$ a *sparse grid*. We use $\mathbf{X}^{=}(q)$ for denoting the corresponding set replaced with the rule that it takes all vectors $N_0 \in \mathbb{N}^d$ with $\|N\|_1 = q$. In case the point set is nice, for example for Chebyshev points, one obtains a *nesting* of sparse grids for increasing fineness scales: $\mathbf{X}^{\leq}(q) \subset \mathbf{X}^{\leq}(q+1)$. This construction has the advantage that in the case of applying this algorithm on a computer, storing whole grid $\mathbf{X}$ in the memory is not necessary. It is enough to use the refinements of each level-increase $q \mapsto q+1$ and interpolate at each such level separately, each time discarding the old values. We may thus define the difference operator

$$(3.17) \qquad \Delta^{(q)} := \left(\bigotimes_{\|N\|_1 = q+1} \mathcal{I}^{(N)} \right) - \left(\bigotimes_{\|N\|_1 = q} \mathcal{I}^{(N)} \right)$$

Now, Smolyak's algorithm can be written as

$$(3.18) \qquad \mathcal{A}(q, d) := \sum_{\|N\|_1 \leq q} \Delta^{(q)}(\mathbf{X}(N))$$

or equivalently

$$(3.19) \qquad \mathcal{A}(q, d) = \mathcal{A}(q-1, d) + \sum_{\|N\|_1 = q} \mathcal{I}^{(N)}(\mathbf{X}(N))$$

with $\mathcal{I}^{(0)} = 0$.

4. Least Squares

Contrary to the construction of exactly interpolating approximants in the case of Smolyak's algorithm, least squares is a conceptually simpler algorithm. We are given the overdetermined system

$$Vz = y$$

where $V \in \mathbb{R}^{n \times m}$ with $n > m$. It is well–known that this system may be inconsistent and no exact solution exists. However, one may always pose the optimisation problem solving for $z \in \mathbb{C}^m$ with the smallest error

$$(4.1) \qquad \inf_{z \in \mathbb{R}^m} \|Vz - y\| .$$

It is further known that, in case of a full–rank matrix V, the unique solution to (4.1) is given by

$$(4.2) \qquad z^\star = \left(V^\top V\right)^{-1} V^\top y \in \mathbb{R}^m.$$

In our specific case of polynomial interpolation, V is the Vandermonde matrix, consisting of basis polynomials $b_1, b_2, \ldots, b_m$ evaluated at the n different sampled points $x_1, x_2, \ldots, x_n$ in $\Omega \subseteq \mathbb{R}^d$ and y is the vector of function values sampled from the unknown function $f \colon \Omega \to \mathbb{R}$, i.e. $y = (f(x_j))_{j=1}^n$. As for such points, the Vandermonde matrix is never singular, the solution to the approximation problem

$$\inf_p \|f - p\|$$

can analytically be expressed as

$$p^\star \colon \Omega \to \mathbb{R}, t \mapsto \sum_{j=1}^m z_j^\star b_j(t)$$

where $z^\star = (z_j \star)_{j=1}^m$ is given by (4.2).

5. Theoretical Guarantees

The notation in the following is borrowed from [**Ullrich˙2020**]. In this section we introduce a formal setting to the former considerations. That is, we consider a Hilbert space H of real-valued functions on a set D such that point evaluation

$$\delta_x : f \mapsto \int_D f \, \mathrm{d}\delta_x = f(x)$$

is a bounded, linear functional on H. The general formulation of least squares allows for a broad class of recovery problems. In our specific case, the function recovery of real–valued functions on a d–dimensional

(compact) subset D using basis functions of a k–dimensional subspace $V_k := \operatorname{span}\{b_1, \ldots, b_k\}$, we consider the specific form of least squares, given by

$$A_{n,k}(f) := \underset{g \in V_k}{\operatorname{argmin}} \sum_{i=1}^{n} \frac{|g(x_i) - f(x_i)|^2}{\varrho_k(x_i)}$$

where

$$\varrho_k(x) = \frac{1}{2}\left(\frac{1}{k}\sum_{j<k} b_{j+1}(x)^2 + \frac{1}{\sum_{j\geq k} a_j^2}\sum_{j\geq k} a_j^2 b_{j+1}(x)^2\right)$$

and $x_1, \ldots, x_n \in D$. Whenever $f \in V_k$, then, of course, $f = A_{n,k}(f)$. With

(5.1) $$e\left(A_{n,k}, H\right) := \sup_{f \in B(H)} \|f - A_{n,k}(f)\|_{L_2},$$

we denote the worst case error of $A_{n,k}$, where we measure the error of the reconstruction in the space $L_2 := L_2(D, \Sigma, \mu)$ of square integrable functions on D with respect to the measure μ, such that H is embedded into L_2. In light of this, the n–th minimal error is denoted by

$$e_n(H) := \inf_{\substack{x_1, \ldots, x_n \in D \\ \varphi_1, \ldots, \varphi_n \in L_2}} \sup_{f \in B(H)} \left\|f - \sum_{i=1}^{n} f(x_i)\varphi_i\right\|_{L_2}$$

and can be understood as the worst case error of the optimal algorithm using n function values. We get the clear inequality $e_n(H) \leq e(A_{n,k}, H)$ for any point set $\{x_1, \ldots, x_n\}$. With

$$a_n(H) := \inf_{\substack{h_1^\star, \ldots, h_n^\star \in H^\star \\ \varphi_1, \ldots, \varphi_n \in L_2}} \sup_{f \in B(H)} \left\|f - \sum_{i=1}^{n} h_i^\star(f)\varphi_i\right\|_{L_2}$$

we denote the n-th approximation number, which is the worst-case error of an optimal algorithm that uses the n best arbitrary linear and bounded functionals as information about the unknown. This quantity is equal to the n-th singular value of the embedding id: $H \to L_2$.

The following is known since [**Krieg˙2020**].

Theorem 5.1 (Krieg–Ullrich). *There exist constants $C, c > 0$ and a sequence of natural numbers (k_n) with each $k_n \geq cn/\log(n+1)$ and for any $n \in \mathbb{N}$ and measure space (D, Σ, μ), and any RKHS H of real-valued functions on D embedded into $L_2(D, \Sigma, \mu)$, we have*

$$e_n(H) \leq \sqrt{\frac{C}{k_n}\sum_{j\geq k_n} a_j(H)^2}.$$

In particular, for

(5.2) $$a_n(H) \lesssim n^{-s} \log^{\alpha+s}(n)$$

with $s > 1/2, \alpha \in \mathbb{R}$, this implies

$$e_n(H) \lesssim n^{-s} \log^{\alpha+s}(n).$$

The following follows from [**Ullrich˙2020**].

Theorem 5.2 (Ullrich). *Given $n \geq 2$ and $c > 0$, let*

$$k_n := \left\lfloor \frac{n}{2^8(2+c)\log n} \right\rfloor,$$

then, for any measure space (D, Σ, μ) and any RKHS H of real-valued functions on D, embedded into $L_2(D, \Sigma, \mu)$, it holds that

$$e_n\left(A_{n,2k_n}, H\right) \leq \sqrt{\frac{2}{k_n} \sum_{j > k_n} a_j(H)^2}$$

with probability at least $1 - 8n^{-c}$.

Examples. In particular, (5.2) is satisfied for the approximation numbers on the Sobolev space of dominating mixed smoothness,

$$H := H^s_{\mathrm{mix}}\left(\mathbb{T}^d\right)$$

$$:= \left\{ f \in L_2\left(\mathbb{T}^d\right) : \|f\|_H^2 := \sum_{m \in \mathbb{N}_0^d} \prod_{j=1}^d (1 + |m_j|^{2s}) \langle f, b_m \rangle_{L_2}^2 < \infty \right\}$$

where $b_m := \otimes_{j=1}^d b_{m_j}^{(1)}$ and $m = (m_1, \ldots, m_d)$ with

$$b_{2m}^{(1)} := \sqrt{2}\cos(2\pi mx)$$
$$b_{2m-1}^{(1)} := \sqrt{2}\sin(2\pi mx)$$

and $b_0^{(1)} := 1$. This satisfies the assumption for $s > 1/2$. In particular, we can say

(5.3) $$e_n\left(H^s_{\mathrm{mix}}\left(\mathbb{T}^d\right)\right) \lesssim n^{-s} \log^{sd}(n)$$

whenever $s > 1/2$. This disproves a previously posted conjecture (Conjecture 5.26) in [**Dung˙Temlyakov˙Ullrich˙2018**] and shows that Smolyak's algorithm is not optimal in this case. The surprising fact is that, despite an optimal, deterministic construction of the point sets used for reconstruction being unknown, random i.i.d. points suffice for a reconstruction error that is on the order of optimal points, with probability tending to 1. We verify this by our experimental findings, presented in Section 6 with a much better relative number of points

used for LSQ function recovery vs. SA recovery than guaranteed in this section, i.e. better constants than explicitly known before. It remains an open problem to rigorously improve upon the constants in (5.3).

6. Experimental Findings

7. Conclusion

J.E., Johannes Kepler University Linz; `jakob.eggl@jku.at`;
E.M., Johannes Kepler University Linz; `elias.mindlberger@jku.at`;
M.U., Johannes Kepler University Linz; `mario.ullrich@jku.at`