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.

Contents

⋆ Equal Contribution.

1. Introduction

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

2. Notation

We use $[N]_0 := \{0, 1, \ldots, N\}$ and $[N]$ for the set $\{1, \ldots, N\}$. For the set of polynomials $p : V \to W$ of maximal degree N and sets $V, W \subseteq \mathbf{C}^K$, $K \in \mathbf{N}$ we use the notation $p \in \mathcal{P}^N(V, W)$. We use $f \propto g$ for functions f, g to denote $f = cg$ for a constant $c \in \mathbf{R}$ and $f \lesssim g$ to denote $f \leq cg$.

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 any collection of points $\mathbf{Z} := \{z_n\}_{n=0}^N \subseteq \mathbf{R}$ and known function values $f(z_n) : n \in [N]_0$ for $f : \mathbf{R} \to \mathbf{R}$ there is always a unique interpolating polynomial i, i.e. $i \in \mathcal{P}^N(\mathbf{R}, \mathbf{R}) : i(z_j) = f(z_j)$ s.t. $j \in [N]_0$. To find the polynomial i, one can, among other available methods, choose to perform Lagrange Interpolation which yields an explicit construction of the form

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

where $\mathbf{F} := \{[z_n, f(z_n)]\}_{n=1}^N$. 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 $\mathcal{I} : \mathbf{R}^{2 \times N} \to \mathcal{P}^N(\mathbf{R}, \mathbf{R})$. since each basis function $\ell_n(x)$ is a polynomial of degree N. Thus, if $f \in \mathcal{P}^N(\mathbf{R}, \mathbf{R})$, 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 $n \neq j$, computing the product $\ell_n(x)$ becomes numerically unstable. (2) The addition of new data $(z, f(z))$ requires to recalculate all basis functions anew 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 eq. (3.1) in a much more stable way as

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

for $w_n \in \mathbf{R} : n \in [N]_0$. If the weights w_n are known, computing $i(x)$ in this form admits a complexity of $\mathcal{O}(N)$. Luckily, the weights have closed-form, analytic solutions 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

$$w_n = \frac{1}{\prod_{\substack{j=0 \\ j \neq n}}^{N} x_n - x_j} = \frac{1}{\ell'(x_n)}$$

always holds. To specify the above, consider the set of *equidistant* nodes $\{z_n := 2/n\} \subset [-1, 1]$. Then

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

For large N, eq. (3.3) is problematic since weights $w_j, w_n : n \neq j$ 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 $\rho(x) \propto 1/\sqrt{1 - x^2}$. Many point sets admit this asymptotic density, for example the Chebyshev points of the first kind are given as

$$\mathbf{Z}_C^{(1)}(N) := \left\{ z_n^{(1)} := \cos \frac{(2n+1)\,\pi}{2n+2} \mid n \in [N]_0 \right\}, \tag{3.4}$$

similarly, the Chebyshev points of the second kind are

$$\mathbf{Z}_C^{(2)}(N) := \left\{ z_n^{(2)} := \cos \frac{n\pi}{N} \mid n \in [N]_0 \right\}. \tag{3.5}$$

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

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

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

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

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 $b_n \in C[0,1]$ such that $b_n(z_j) = \delta_n(j)$. To obtain a set of multidimensional point sets from one dimensional rules like eq. (3.4) or eq. (3.5), one uses the usual cartesian product

$$\mathbf{Z}(N,d) := \mathbf{Z}(N_1) \times \cdots \times \mathbf{Z}(N_d) := \bigcup_{j=1}^{d} \bigcup_{n_j=1}^{m(N_j)} \{(z_{n_1}, \ldots, z_{n_d})\} \tag{3.8}$$

where $z_n := (z_{n_1}, \ldots, z_{n_d})$ and $N := (N_1, \ldots, N_d) \in \mathbf{N}^d$ is the usual multi-index. Using the tensor product formula, one obtains the very general expression

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

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

$$\bigotimes_{j=1}^{d} \mathcal{I}^{(N_j)} : \mathbf{R}^{(d+1)\times\left(\sum_{j=1}^{d} m(N_j)\right)} \to \mathcal{P}^{(\max\{m(N_j)\})}\left(\mathbf{R}^d, \mathbf{R}\right). \tag{3.10}$$

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

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

In Smolyak's Algorithm, the function $m : \mathbf{N} \to \mathbf{N}$ 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

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

Clearly, the usual representation of the Lagrange polynomial (3.1) fits this framework. For the barycentric form, one obtains the d-variate interpolation

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

where

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

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 N_j (or $m(N_j)$) to pick in each direction. If this number is large simultaneously for all dimensions, eq. (3.9) becomes intractable very quickly. Smolyak's algorithm defines a resolution-like number $q \in \mathbf{N}$ to specify how close-meshed points should be picked. It proceeds by taking all multi-indices N with 1-norm less than or equal to q and specifies to interpolate with the algorithm

$$\mathcal{A}(q,d)\,(\mathbf{X}) := \sum_{\|N\|_1 \le q} \left(\mathcal{I}^{(N_1)} \otimes \cdots \otimes \mathcal{I}^{(N_d)}\right)(\mathbf{X}(N)). \tag{3.14}$$

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

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

All points used by this algorithm are then

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

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 $N : \|N\|_1 = q$. In case the point set is nice, for example for Chebyshev points, one obtains a *nesting* of sparse grids for increasing fineness scales: $\mathbf{X}^{\le}(q) \subset \mathbf{X}^{\le}(q+1)$. This construction has the advantage that the practicioner does not need to save the whole grid $\mathbf{X}$ at once. 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

$$\Delta^{(N)} := \left(\mathcal{I}^{N_1} \otimes \cdots \otimes \mathcal{I}^{N_d}\right) - \left(\mathcal{I}^{N_1 - 1} \otimes \cdots \otimes \mathcal{I}^{N_d - 1}\right). \tag{3.17}$$

Now, Smolyak's algorithm can be written as

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

or equivalently

$$\mathcal{A}(q,d) = \mathcal{A}(q-1,d) + \sum_{\|N\|_1 = q} \Delta^{(N)}\left(\mathbf{X}(N)\right) \tag{3.19}$$

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 $V \in \mathbf{R}^{n\times m}$ 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 $z \in \mathbf{C}^m$ with the smallest error

$$\inf_{z\in\mathbf{R}^m} \|Vz - y\| \,. \tag{4.1}$$

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

$$z^\star = \left(V^\top V\right)^{-1} V^\top y.$$

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 sampled points $x_1, x_2, \ldots, x_n$ in $\Omega \subseteq \mathbf{R}^d$ and y is a vector of function values sampled from the unknown function $f : \Omega \to \mathbf{R}$, i.e. $y = \left(f(x_j)\right)_{j=1}^n$. In this case, the Vandermonde matrix is never singular as long as the sampled points do not overlap. Hence, the solution to the approximation problem

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

can analytically be expressed as

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

where $z^\star = \left(z_j\right)_{j=1}^m$ is given by eq. (4.1).

5. Theoretical Guarantees

6. Experimental Findings

7. Conclusion

Institute of Analysis, Johannes Kepler University Linz, Austria.
Email address: `jakob.eggl@jku.at`

Institute of Analysis, Johannes Kepler University Linz, Austria.
Email address: `elias.mindlberger@jku.at`

Institute of Analysis, Johannes Kepler University Linz, Austria.
Email address: `mario.ullrich@jku.at`