Kolmogorov Superposition Theorem: Univariate Encodings for Multivariate Functions

Jonas Actor [1] Matthew Knepley [2]

[1]Computational and Applied Mathematics, Rice University [2]Computer Science and Engineering, University at Buffalo

Background

In answering Hilbert's 13th problem, A. Kolmogorov proved that any continuous multivariate function can be represented by the sum of a small number of function compositions, reducing computation with multivariate functions into univariate function evaluation, function composition, and addition. Later works by Sprecher and Lorenz reformulated this statement into the following theorem, reducing the number of functions needed for representation to one outer function and one inner function.

Kolmogorov Superposition Theorem

Let $f : \mathbb{R}^n \to \mathbb{R} \in C([0,1]^n)$ where $n \geq 2$. Fix $\epsilon = \frac{1}{2n}$, and choose $\lambda_1, \ldots, \lambda_n \in \mathbb{R}$ integrally independent. Then, there exist $\psi : [-1,1] \to \mathbb{R} \in C[-1,1]$ and $\chi : \mathbb{R} \to \mathbb{R} \in C(\mathbb{R})$, such that

$$f(x_1, \ldots, x_n) = \sum_{q=0}^{2n} \chi \left(\sum_{p=1}^{n} \lambda_p \psi(x_p + q\epsilon) + nq \right).$$

The function ψ and the constants λ_p, ϵ are independent of f we aim to represent, allowing us to consider a univariate function χ in place of the multivariate function f. **We aim to use KST to create a computational scheme that, given a function f, returns its counterpart χ.** To do so involves two tasks:

1. Precompute inner function ψ, space decomposition $\mathscr{T}_j$
2. Create algorithm, given f, ψ, $\mathscr{T}_j$, constructs outer function χ

Applications

- image compression
- content-based image/video retrieval
- encryption
- PDE dimension reduction

Properties of Towns $\mathscr{T}_j$

- $\mathscr{T}_j \subset [-1,1]$ is a collection of towns (closed intervals)
- Towns refine with j uniformly i.e. $\sup_{t \in \mathscr{T}_j} Diam(t) \leq \frac{1}{2^j}$.
- We have $2n+1$ shifted copies of $\mathscr{T}_j$, denoted $\mathscr{T}_j^q = \{t + q\epsilon \ : \ t \in \mathscr{T}_j\}$.
- For any point $x \in [0,1]$, at each level of refinement j there are at least $2n$ values of q such that $x \in t_j^q$ for some town $t_j^q \in \mathscr{T}_j^q$.

Properties of Inner Function ψ

- $\psi = \lim_{j \to \infty} \psi_j, \qquad \Psi(x_1, \ldots, x_n) = \sum_{p=1}^{n} \lambda_p \psi(x_p + q\epsilon) + q$
- $\psi_j(t) = c_t \in \mathbb{Q}$ for each town $t \in \mathscr{T}_j$, interpolating linearly between towns
- For distinct towns $t_1, t_2 \in \mathscr{T}_j$, $\psi_j(t_1) \neq \psi_j(t_2)$
- Slope of ψ_j is bounded by $1 - \frac{1}{2^j}$ to achieve desired Lip(1) smoothness bound

Construction of Inner Function ψ

Refine towns: Check to make valid gaps

Need to keep each point in $[0,1]$ contained in at least $2n$ of the $2n+1$ shifted systems of towns even once we remove an open interval around each break point

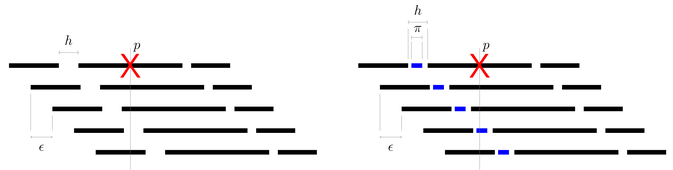

Figure 1: Sketch where a break point p falls into hole h for $n = 2$, and of adding a plug for that hole.

If we cannot break without causing some points to no longer belong to $2n$ of $2n+1$ shifted systems of towns, we add to $\mathscr{T}_j$ smaller intervals that plug those gaps.

Add Plugs: Maintaining Bounded Slope

Adding plugs causes an increase in slope, as seen in the figure below. Slope needs to stay bounded, but we desire the largest valid plug sizes, so we solve a resulting (small) dense linear system.

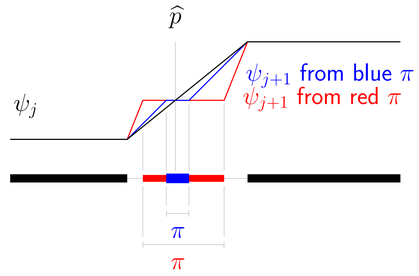

Figure 2: Sketch of how size of plugs changes the slope of ψ_{j+1}.

Once we have added in the necessary plugs, we can remove small enough open intervals around our break points, while maintaining our desired property.

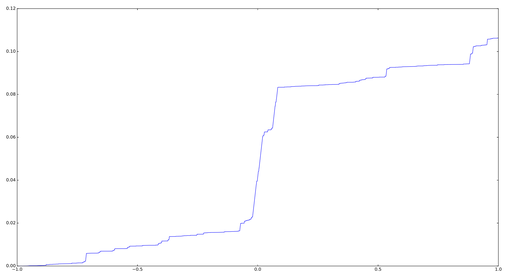

Figure 3: Function ψ after 11 levels of refinement

Construction is implemented in Python with mpmath and IntervalTree packages; resulting ψ and towns $\mathscr{T}_j$ through first few iterations are shown in figures 3 and 4.

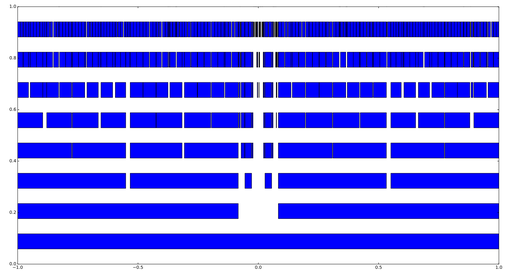

Figure 4: Towns $\mathscr{T}_j$ for $j = 0$ (bottom) to $j = 7$ (top)

Construction of Outer Function χ

- χ only as smooth as ϕ as to cancel out the non-differentiability
- Constructed iteratively by bounding oscillation and refining;

$$\chi^q = \lim_{r \to \infty} \chi_r^q, \qquad f_r(x) = \sum_{q=0}^{2n} \chi_r^q \left(\Psi^q(x) \right).$$

- Denote $M_r = \|f - f_r\|_\infty$, and oscillation of a function g on square S be given by $\omega(g, S)$.
- Linear convergence at rate $\frac{2n+1}{2n+2}$.

Procedure

Initialize $\chi_0 \equiv 0$, $f_0 \equiv 0$, and $M_0 = \|f\|_\infty$. For $r = 1, 2, \ldots$ do:

- Given χ_{r-1} and therefore f_{r-1}, refine towns until we arrive at refinement level $k_r \in \mathbb{N}$ such that

$$\forall S_{k_r}^q \in \mathscr{S}_{k_r}^q, \quad \omega(f - f_{r-1}, S_{k_r}^q) \leq \frac{1}{2n+1} M_{r-1}.$$

- Correct χ on the image of each square $S_{k_r}^q \in \mathscr{S}_{k_r}^q$ by setting for $y \in \Psi^q(S_{k_r}^q)$,

$$\chi_r(y) = \chi_{r-1}(y) + \frac{1}{n+1} \left[f(\xi_{k_r}^q) - f_{r-1}(\xi_{k_r}^q) \right],$$

where ξ_{kr}^q is the lower left corner of the specific square $S_{k_r}^q$.

- Update f_r with new definition for χ_r

Moving Forward

- Implement χ construction
- Error estimates for χ computation at each refinement stage
- Complexity analysis for ψ, χ construction

Acknowledgements

JA would like to acknowledge support from the Ken Kennedy Institute Computer Science & Engineering Enhancement Fellowship, funded by the Rice Oil & Gas HPC Conference. MGK would like to acknowlege partial support from NSF award SI2-SSI: 1450339 and U.S. DOE Contract DE-AC02-06CH11357.