Choosing Kernel Shape Parameters in Partition of Unity Methods by Univariate Global Optimization Techniques

Roberto Cavoretto[1(✉)], Alessandra De Rossi[1], and Yaroslav D. Sergeyev[2,3]

[1] Department of Mathematics "Giuseppe Peano", University of Turin, Via Carlo Alberto 10, 10123 Turin, Italy
{roberto.cavoretto,alessandra.derossi}@unito.it

[2] DIMES, University of Calabria, via P. Bucci, Cubo 42-C, 87036 Rende (CS), Italy
yaro@dimes.unical.it

[3] ITMM, Lobachevsky Nizhni Novgorod State University, Gagarin Avenue 23, 603950 Nizhni Novgorod, Russia

Abstract. In this paper, we present a numerical scheme to select the kernel shape parameters within partition of unity methods. In an interpolation framework, we propose the use of a leave-one-out cross validation technique combined with efficient global optimization tools from the class of Lipschitz derivative-free methods. Numerical results highlight how this union is generally able to produce some enhancements in terms of both efficiency and accuracy.

Keywords: Kernel interpolation · Shape parameter · Global optimization

1 Introduction

The problem of determining an optimal value of the shape parameter is very important in the radial basis function (RBF) or kernel-based interpolation community. Indeed, it is well-known that the choice of such parameter influences both the accuracy and stability of the kernel interpolant [7]. In the literature, there are several strategies that can be employed for the shape parameter detection deriving from either empirical studies or consolidated optimization methods (see, e.g., [8,14]).

In [5] we proposed to use univariate Lipschitz global optimization (GO) algorithms for finding a good value of the shape parameter. The choice of the Lipschitz methods is explained by the fact that, on the one hand, they are derivative-free, that is very precious in our setting. In addition, they have shown to be very efficient in many practical applications [9–12]. More precisely, we used a well-known Leave-One-Out Cross Validation (LOOCV) technique to introduce

Y. D. Sergeyev et al. (Eds.): NUMTA 2023, LNCS 14477, pp. 207–214, 2025.
https://doi.org/10.1007/978-3-031-81244-6_17

the error (or objective) function of the optimization problem. Combining the LOOCV with efficient strategies of GO with pessimistic and optimistic improvements [13], we could increase the performance of the standard method. While in [5] we were focused on global RBF interpolation, in this article we extend our numerical method to a RBF partition of unity method (RBF-PUM), see e.g. [3,4]. This numerical scheme enables us to decompose a big problem into several smaller sub-problems, first computing the local RBF interpolants and then gathering all partial contributions in the global fit. Numerical experiments show how the new LOOCV-GO algorithm performs in comparison with a direct LOOCV competitor.

The paper is organized as follows. In Sect. 2 we briefly describe the RBF-PUM for data interpolation. Section 3 contains a presentation of the basic LOOCV technique and its extension with GO tools. In Sect. 4 we report some numerical results to show algorithm performance.

2 Kernel-Based Partition of Unity Method

Let us consider a set $X_N = \{\boldsymbol{x}_i, i = 1, \ldots, N\} \subseteq \Omega$ of distinct *data points* or *nodes*, arbitrarily distributed on a domain $\Omega \subseteq \mathbb{R}^d$, with an associated set $F_N = \{f_i = f(\boldsymbol{x}_i), i = 1, \ldots, N\}$ of *function* or *data values* that are obtained by sampling some (unknown) function $f : \Omega \to \mathbb{R}$ at the nodes $\boldsymbol{x}_i$. The *scattered data interpolation* problem consists of finding a function $s : \Omega \to \mathbb{R}$ such that $s(\boldsymbol{x}_i) = f_i$, $i = 1, \ldots, N$.

Given a *strictly positive definite* (SPD) and *symmetric* kernel $\Phi : \Omega \times \Omega \to \mathbb{R}$, we take $s \in H_\Phi(X_N) = \text{span}\{\Phi(\cdot, \boldsymbol{x}_i), \boldsymbol{x}_i \in X_N\}$. In particular, here we consider radial kernels or RBFs, assuming that there exists a function $\phi : [0, \infty) \to \mathbb{R}$ depending on a *shape parameter* $\varepsilon > 0$ such that $\Phi(\boldsymbol{x}, \boldsymbol{y}) = \phi_\varepsilon(||\boldsymbol{x} - \boldsymbol{y}||_2) := \phi(r)$, for all $\boldsymbol{x}, \boldsymbol{y} \in \Omega$.

When wanting to interpolate large data sets, the use of a global RBF interpolant is known to be computationally costly [6]. The PUM, presented in the following, enables us to overcome such issue. Thus, we firstly construct a covering $\{\Omega_j\}_{j=1}^m$ of the domain Ω with a finite number m of overlapping sub-domains Ω_j, such that such covering is *regular* [16], i.e.,

1) for every $\boldsymbol{x} \in \Omega$, the number of sub-domains Ω_j, with $\boldsymbol{x} \in \Omega_j$, is bounded by a global constant,
2) each sub-domain Ω_j satisfies an interior cone condition,
3) the local fill distances $h_{X_{N_j}}$ are uniformly bounded by the global fill distance h_{X_N}, where $X_{N_j} = X_N \cap \Omega_j$.

Once we selected weight functions w_j, $j = 1, \ldots, m$, we can define the RBF-PUM interpolant:

$$s(\boldsymbol{x}) = \sum_{j=1}^{m} s_j(\boldsymbol{x})\, w_j(\boldsymbol{x}), \quad \text{with} \quad s_j(\boldsymbol{x}) = \sum_{k=1}^{N_j} \alpha_k^j \phi(||\boldsymbol{x} - \boldsymbol{x}_k^j||_2), \tag{1}$$

where s_j is defined on the sub-domain Ω_j, N_j represents the number of points on Ω_j and $\boldsymbol{x}_k^j \in \mathcal{X}_{N_j}$, with $k = 1, \ldots, N_j$. To determine the coefficients α_k^j in (1), we need to solve m linear systems of the form

$$\mathsf{K}_j \boldsymbol{\alpha}_j = \boldsymbol{f}_j, \tag{2}$$

$(\mathsf{K}_j)_{ik} = \phi(||\boldsymbol{x}_i^j - \boldsymbol{x}_k^j||_2)$, $i, k = 1, \ldots, N_j$, being the entries of $\mathsf{K}_j \in \mathbb{R}^{N_j \times N_j}$, $\boldsymbol{\alpha}_j = (\alpha_1^j, \ldots, \alpha_{N_j}^j)^T$, and $\boldsymbol{f}_j = (f_1^j, \ldots, f_{N_j}^j)^T$. Notice that the uniqueness of the solution in (2) is ensured by the fact that the kernel Φ is SPD and symmetric.

Since the coefficients of (1) are found by imposing the local interpolation conditions, that is $R_j(\boldsymbol{x}_i^j) = f_i^j$, $i, k = 1, \ldots, N_j$, $j = 1, \ldots, m$, the functions w_j must form a *partition of unity*. Further, we require that such partition of unity is *k-stable* [16], which in particular implies that $\text{supp}(w_j) \subseteq \Omega_j$. For instance, such conditions are satisfied for the well-known *Shepard's weights*, see e.g. [15].

3 Methods for Choosing Optimal Shape Parameters

3.1 The Basic LOOCV Method

The LOOCV technique for the search of optimal values of the shape parameter ε in (1) can be briefly described as follows. Firstly, for any fixed ε and each $k = 1, \ldots, N_j$, the node $\boldsymbol{x}_k$ and the corresponding value $f(\boldsymbol{x}_k)$ are excluded from the local sets X_{N_j} and F_{N_j}, respectively. Then, the partial local RBF interpolant is constructed using only $N_j - 1$ remaining nodes and the interpolation error at the point $\boldsymbol{x}_k$ is calculated [2]. This error can be derived without solving N_j interpolation problems of dimension $N_j - 1$ by the rule

$$e_k^j(\varepsilon) = \frac{\alpha_k^j}{(\mathsf{K}_j^{-1})_{kk}}, \tag{3}$$

where α_k^j is the kth coefficient of the full local RBF interpolant s_j in (1) and $(\mathsf{K}_j^{-1})_{kk}$ is the inverse diagonal element of the matrix K_j in (2).

Accordingly, in the sub-domain Ω_j the value of ε can be determined by minimizing the error function as follows:

$$\text{err}_j(\varepsilon) = \max_{k=1,\ldots,N_j} \left| e_k^j(\varepsilon) \right|, \qquad \forall j = 1, \ldots, m. \tag{4}$$

3.2 The New LOOCV-GO Method

To find the optimal value of ε, for each Ω_j, $j = 1, \ldots, m$, an optimization problem must be solved [4]. In particular, it is required to determine the point ε^* and the corresponding value err_j^*, where $\text{err}_j(\varepsilon)$ is from (4) such that

$$\text{err}_j^* = \text{err}_j(\varepsilon^*) = \min \text{err}_j(\varepsilon), \qquad \varepsilon \in [0, \varepsilon_{max}], \tag{5}$$

ε_{max} being large enough. Here, the function $\mathrm{err}_j(\varepsilon)$ can be multiextremal, non-differentiable and hard to evaluate even at one value of ε, since for each ε-value the local interpolants s_j in (1) has to be computed. Furthermore, it is assumed to be Lipschitz-continuous over the interval $[0, \varepsilon_{max}]$:

$$|\mathrm{err}(\varepsilon_1) - \mathrm{err}(\varepsilon_2)| \leq L|\varepsilon_1 - \varepsilon_2|, \qquad \varepsilon_1, \varepsilon_2 \in [0, \varepsilon_{max}],$$

where $L \in (0, \infty)$ is the Lipschitz constant. Notice that since the function $\mathrm{err}_j(\varepsilon)$ is usually ill-conditioned for $\varepsilon \to 0$ (see, e.g., [7]), the Lipcshitz constant L turns out to be very large. Nevertheless, in [5], we showed that information GO algorithms can be used successfully in combination with LOOCV in order to solve RBF interpolation problems. Now, we extend this approach to RBF-PUM by locally minimizing the LOOCV-based error function (4) on sub-domain Ω_j, $j = 1, \ldots, m$.

Adapting the procedure from [5,13], the computational scheme to solve the problem (5) using GO is briefly sketched in Algorithm 1.

Algorithm 1. Minimization of the error function using locally LOOCV

Input: The error function (4), the right limit ε_{max} of the search interval.
Step 1: *Preliminary global search* of ε over the interval $[0, \varepsilon_{max}]$;
Step 2: *Ill-conditioned region refinement* by a local search;
Step 3: *Aggregation* of results found in previous steps;
Step 4: *Restriction of the search region* over the interval $[\varepsilon_{imin}, \varepsilon_{imax}] \subset [0, \varepsilon_{max}]$;
Step 5: *Final global search* over $[\varepsilon_{imin}, \varepsilon_{imax}]$ until a stopping criterion is satisfied.
Output: The optimal shape parameter ε .

The new LOOCV-GO method that was briefly introduced above is employed hereinafter using an optimistic local improvement technique [13]. This scheme is a direct extension to RBF-PUM of the LOOCV-GOOI scheme in [5].

4 Numerical Results

In this section we show results of our numerical tests by comparing the basic LOOCV method and the new LOOCV-GO one discussed in Sect. 3. Both schemes were included in the PUM (1) for choosing optimal values of ε in the local RBF interpolants. As a starting point, we considered the adaptive PUM algorithm implemented in [2], which then was suitably modified for our scopes. Moreover, we set the code so that each sub-domain had a minimum cardinality of 20 nodes. All the experiments were carried out on a laptop with an Intel(R) Core(TM) i7-1065G7 CPU 1.50 GHz processor and 16.0 GB RAM.

The basic LOOCV method is applied by using a uniform grid of ε-values over the interval $[0, \varepsilon_{max}]$ in (5), with stepsize $h = \varepsilon_{max}/100$ and $\varepsilon_{max} = 10$. It is used as a comparison with the LOOCV-GO. In the latter, after careful investigation we set the parameter $\delta = 0.03$ as a stopping criterion in the GO algorithm (see [5] for details).

In the performed experiments we considered some low discrepancy Halton node sets X_N [6], which were generated for $N = 1000, 2000, 4000, 8000, 16000$, in the unit square domain $\Omega = [0, 1]^2 \subset \mathbb{R}^2$. Then, the corresponding data values were taken by the following test function [1,2]:

$$f(x_1, x_2) = \frac{1}{2} x_2 \cos^4 \left[4 \left(x_1^2 + x_2 - 1 \right) \right]. \tag{6}$$

The two PUM algorithms were tested on two SPD RBFs that are listed below together with their smoothness degrees (and related abbreviations in round brackets) [6]:

$$\phi(r) = \begin{cases} \exp(-\varepsilon^2 r^2), & \text{Gaussian } C^\infty, \quad \text{(GA)}, \\ \exp(-\varepsilon r)(\varepsilon^3 r^3 + 6\varepsilon^2 r^2 + 15\varepsilon r + 15), & \text{Matérn } C^6, \quad \text{(M6)}, \end{cases}$$

while for *Shepard's weights* w_j in (1) we used compactly supported Wendland's C^2 functions [16].

In order to measure and compare accuracy of the two PUMs, we computed the Maximum Absolute Error (MAE) and the Root Mean Square Error (RMSE), i.e.,

$$\text{MAE} = \max_{1 \le i \le M} |f(\boldsymbol{\xi}_i) - s(\boldsymbol{\xi}_i)|, \qquad \text{RMSE} = \sqrt{\frac{1}{M} \sum_{i=1}^{M} |f(\boldsymbol{\xi}_i) - s(\boldsymbol{\xi}_i)|^2},$$

where the $\boldsymbol{\xi}_i$, $i = 1, \ldots, M$, form a grid of $M = 40 \times 40$ evaluation points. Furthermore, to analyze computational efficiency of the algorithms, hereinafter we also reported CPU times (in seconds).

Table 1. Comparison between RBF-PUMs using GA kernel: LOOCV vs LOOCV-GO.

N	LOOCV			LOOCV-GO		
	MAE	RMSE	CPU time	MAE	RMSE	CPU time
1000	1.7958e−02	6.1070e−04	6.77	3.2580e−03	1.6309e−04	1.10
2000	2.7044e−03	1.0455e−04	11.92	1.1585e−03	3.3050e−05	1.67
4000	7.7035e−05	3.8772e−06	24.31	7.7370e−05	3.7637e−06	3.09
8000	2.9719e−02	7.5838e−04	52.74	3.1133e−05	9.3882e−07	6.09
16000	3.5202e−04	8.8059e−06	115.30	2.2616e−06	1.0711.e−07	13.08

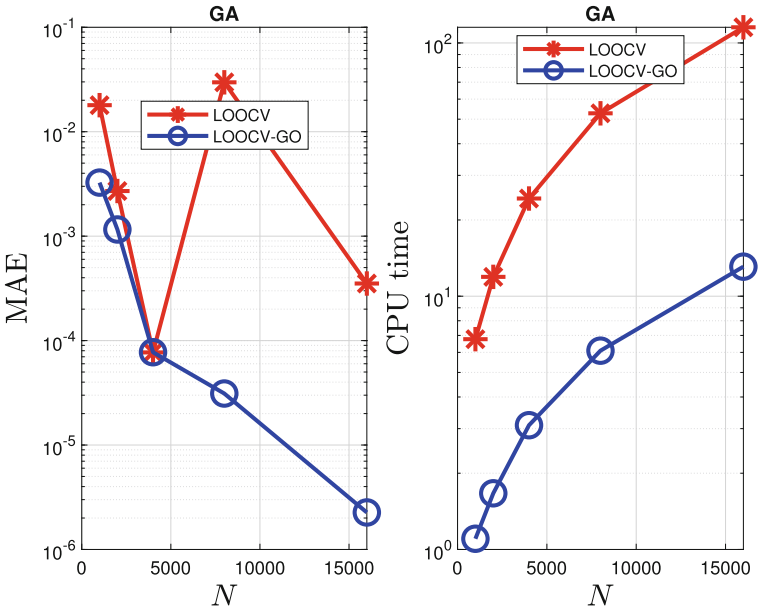

Fig. 1. Graphical comparison between LOOCV vs LOOCV-GO in RBF-PUMs: MAE (left) and CPU time (right) by using GA kernel.

Table 2. Comparison between RBF-PUMs using M6 kernel: LOOCV vs LOOCV-GO.

N	LOOCV			LOOCV-GO		
	MAE	RMSE	CPU time	MAE	RMSE	CPU time
1000	5.5744e−03	3.1127e−04	6.24	4.4277e−03	2.5981e−04	0.98
2000	4.2041e−03	1.8735e−04	14.42	1.4817e−03	7.0030e−05	1.95
4000	9.1417e−04	4.0742e−05	30.25	3.7867e−04	1.6598e−05	4.33
8000	1.5541e−04	6.7816e−06	63.69	5.2993e−05	2.8697e−06	9.20
16000	6.3631e−05	2.8531e−06	142.46	4.5011e−05	1.3458e−06	20.01

The obtained results were collected in Table 1 and Fig. 1 for GA kernel and in Table 2 and Fig. 2 for M6 kernel. In our study, we analyzed numerically algorithm performance, comparing results achieved by applying basic LOOCV and new LOOCV-GO. From this investigation, it is evident that the combination of LOOCV with GO tools leads to a general improvement in terms of efficiency, sometimes also pointing out a reduction of interpolation error.

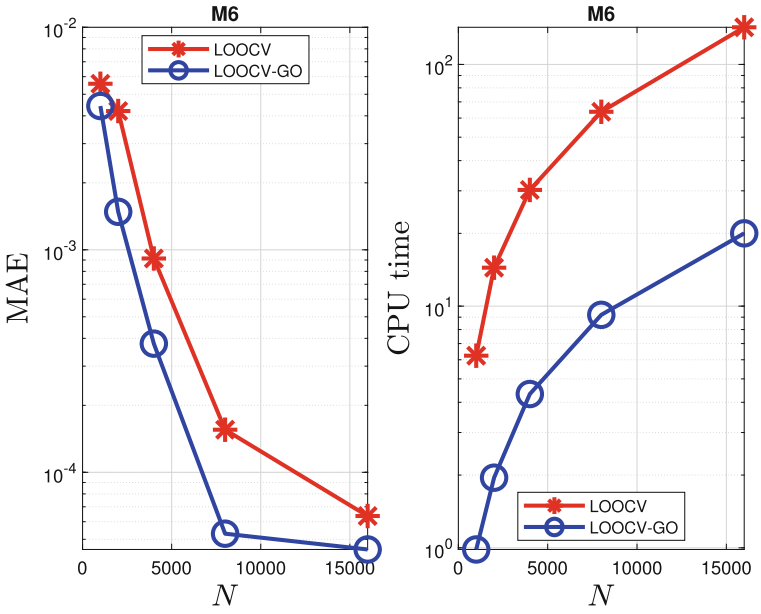

Fig. 2. Graphical comparison between LOOCV vs LOOCV-GO in RBF-PUMs: MAE (left) and CPU time (right) by using M6 kernel.

Acknowledgements. The authors sincerely thank the reviewers for their valuable and constructive comments. The research of R. Cavoretto and A. De Rossi has been supported by the Grant For Internationalization (GFI) 2022 Project "New Approximation and Artificial Intelligence/Machine Learning Tools for Modeling of Traffic Networks" funded by the University of Turin. The work has then been supported by the INdAM–GNCS 2022 project "Computational methods for kernel-based approximation and its applications", code CUP_E55F22000270001, and by the INdAM–GNCS 2023 project "Approssimazione ed integrazione multivariata con applicazioni ad equazioni integrali", code CUP_E53C22001930001. Moreover, the work has been supported by the Spoke "FutureHPC & BigData" of the ICSC – Centro Nazionale di Ricerca in "High-Performance Computing, Big Data and Quantum Computing", funded by European Union – NextGenerationEU. This research has been accomplished within the RITA "Research ITalian network on Approximation" and the UMI Group TAA "Approximation Theory and Applications". The research of Y.D. Sergeyev has been supported by the GNCS–INdAM 2023 project "Numerical high-precision algorithms for solving optimization problems and ODEs with applications".

References

1. Bozzini, M., Lenarduzzi, L., Rossini, M.: Polyharmonic splines: an approximation method for noisy scattered data of extra-large size. Appl. Math. Comput. **216**, 317–331 (2010)
2. Cavoretto, R.: Adaptive radial basis function partition of unity interpolation: a bivariate algorithm for unstructured data. J. Sci. Comput. **87**, 41 (2021)

3. Cavoretto, R., De Rossi, A.: Error indicators and refinement strategies for solving Poisson problems through a RBF partition of unity collocation scheme. Appl. Math. Comput. **369**, 124824 (2020)
4. Cavoretto, R., De Rossi, A., Erb, W.: Partition of unity methods for signal processing on graphs. J. Fourier Anal. Appl. **27**, 66 (2021)
5. Cavoretto, R., De Rossi, A., Mukhametzhanov, M.S., Sergeyev, Ya.D.: On the search of the shape parameter in radial basis functions using univariate global optimization methods. J. Global Optim. **79**, 305–327 (2021)
6. Fasshauer, G.E.: Meshfree Approximation Methods with MATLAB. World Scientific, Singapore (2007)
7. Fasshauer, G., McCourt, M.: Kernel-based Approximation Methods using MATLAB. World Scientific, Singapore (2015)
8. Golbabai, A., Mohebianfar, E., Rabiei, H.: On the new variable shape parameter strategies for radial basis functions. Comput. Appl. Math. **34**, 691–704 (2015)
9. Grishagin, V.A., Israfilov, R.A.: Global search acceleration in the nested optimization scheme. In: AIP Conference Proceedings **1738**, pp. 400010:1–4 (2016)
10. Kvasov, D.E., Menniti, D., Pinnarelli, A., Sergeyev, Y.D., Sorrentino, N.: Tuning fuzzy power-system stabilizers in multi-machine systems by global optimization algorithms based on efficient domain partitions. Electr. Power Syst. Res. **78**(7), 1217–1229 (2008)
11. Kvasov, D.E., Sergeyev, Y.D.: Lipschitz global optimization methods in control problems. Autom. Remote. Control. **74**(9), 1435–1448 (2013). https://doi.org/10.1134/S0005117913090014
12. Sergeyev, Y.D., Daponte, P., Grimaldi, D., Molinaro, A.: Two methods for solving optimization problems arising in electronic measurements and electrical engineering. SIAM J. Optim. **10**(1), 1–21 (1999)
13. Sergeyev, Y.D., Mukhametzhanov, M.S., Kvasov, D.E., Lera, D.: Derivative-free local tuning and local improvement techniques embedded in the univariate global optimization. J. Optim. Theory Appl. **171**, 186–208 (2016)
14. Scheuerer, M.: An alternative procedure for selecting a good value for the parameter c in RBF-interpolation. Adv. Comput. Math. **34**, 105–126 (2011)
15. Wendland, H.: Fast evaluation of radial basis functions: methods based on partition of unity. In: Chui, C.K., et al. (eds.) Approximation Theory X: Wavelets, Splines, and Applications, pp. 473–483. Vanderbilt University Press, Nashville (2002)
16. Wendland, H.: Scattered Data Approximation. Cambridge University Press, Cambridge (2005)