HIERARCHICAL APPROXIMATIONS FOR DATA REDUCTION AND LEARNING AT MULTIPLE SCALES

Prashant Shekhar* and Abani Patra
Data Intensive Studies Center
Tufts University
Medford, MA, 02155, USA

Abstract. This paper describes a hierarchical learning strategy for generating sparse representations of multivariate datasets. The hierarchy arises from approximation spaces considered at successively finer scales. A detailed analysis of stability, convergence and behavior of error functionals associated with the approximations are presented, along with a well chosen set of applications. Results show the performance of the approach as a data reduction mechanism for both synthetic (univariate and multivariate) and a real dataset (geo-spatial). The sparse representation generated is shown to efficiently reconstruct data and minimize error in prediction. The approach is also shown to generalize well to unseen samples, extending its prospective application to statistical learning problems.

1. Introduction. Hierarchical and multiscale modeling are widely used both in data driven and physics based models [49, 30, 34, 42]. In this paper, we focus on a strategy which analyzes data at multiple scales \( s \) by constructing corresponding bases representations \( B^s \in \mathbb{R}^{n \times l_s} \) and inferring if these scale and data dependent bases are able to approximate the observed data \( f|_X \in \mathbb{R}^n \ (X \in \mathbb{R}^{n \times d}) \) to satisfy:

\[
\min_{C_s \in \mathbb{R}^{l_s}} ||f|_X - B^s C_s||_2 \leq TOL
\]

where, \( l_s \in \mathbb{N} \) is a scale enumeration chosen by the algorithm, \( TOL \) is a user defined tolerance level, and \( n \) is the number of observations in a \( d \) dimensional space \( (x_i \in \mathbb{R}^d, 1 \leq i \leq n) \). \( C_s \) in (1) is computed as the optimal coordinate of projection on \( B^s \). Besides constructing these bases at each scale, the algorithm identifies ‘representative’ data points which form the corresponding sparse data \( X_s \) with \( X_s \subseteq X \). \( \{s, X_s, C_s\} \) allow reconstruction of the original dataset and prediction (model) at any new point in the domain of interest resulting in both a sparse representation [37, 16, 45] and an efficient model [28, 12]. The code for the proposed approach is available online 1

Data sparsification seeks to find a subset \( X_s \subseteq X = \{x_1, x_2, ..., x_n\} \subseteq \Omega \ (x_i \in \mathbb{R}^d \) for \( d \) dimensional inputs) for which we have data \( f|_X = (f(x_1), f(x_2), ..., f(x_n))^T \in \mathbb{R}^n \), such that, we may obtain an acceptable reconstruction or approximation to \( f \) as in equation (1). This subset with the corresponding projection coordinate \( C_s \)

---

2020 Mathematics Subject Classification. Primary: 68W25, 65D15; Secondary: 65D12.

Key words and phrases. Sparse representation and learning, data reduction, approximation error analysis.

This work was funded by the grants NSF1821311, NSF1645053, NSF1621853.

* Corresponding author: Prashant Shekhar.

1https://github.com/pshekhar-tufts/Hierarchical_Algorithm_code.git
with respect to the basis set $B^s$, is regarded as our obtained sparse representation ($D^s_{\text{parse}} = [X_s, C_s]$). Thus, this compressed representation of the dataset acts as a model for new prediction points $X_s \subset \Omega$ and reconstruction of the original dataset $f|X$ at $X \subset \Omega$ as needed. The approximation problem \cite{14, 41} is a more traditional one seeking to construct an approximation $(Af) : \mathbb{R}^d \rightarrow \mathbb{R}$ for the underlying function $f$ which was discretely observed $(f|X)$ such that it minimizes the normed error measure $||(Af) - f||$. Following the standard notation, for a normed space $\mathcal{V}$ and any approximation $\tilde{f}$ to $f$, such that $f, \tilde{f} \in \mathcal{V}$, the quality of approximation $\tilde{f}$ is usually quantified by the error norm $||f - \tilde{f}||_{\mathcal{V}}$. $\tilde{f}$ here is usually constructed by first fixing a finite dimension approximation space $Q \subseteq \mathcal{V}$ and then computing the optimal approximation $(Af) \in Q$ by minimizing normed distance between $f$ and $Q$:

$$
\min_{f \in Q} ||f - \tilde{f}||_{\mathcal{V}} = ||f - (Af)||_{\mathcal{V}} \quad (2)
$$

We now motivate our development of a data dependent bases ($B^s \equiv B(X)^s$) \cite{22}. The Mairhuber-Curtis theorem \cite{2} states that for $d \geq 2$ and $n \geq 2$, there is no Chebyshev system $B = \{s_1, ..., s_n\}$ on $\mathbb{R}^d$, therefore for some data $X = \{x_1, x_2, ..., x_n\} \subset \mathbb{R}^d$, the Vandermonde matrix approaches singular behavior affecting the quality of approximation. As a remedy, we choose a data dependent continuous kernel function $K : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$ which is positive definite and symmetric on $\Omega$. Thus, for data points $X = \{x_1, x_2, ..., x_n\}$, the kernel function becomes $K(x_i, x_j) \equiv K(x_i, x_j)$ for $i, j \leq n$. From the literature on these functions \cite{3, 11, 50}, these kernels are reproducing in the native Hilbert Space $H$ of functions on $\Omega$ in the sense

$$<f, K(x, \cdot) >_H = f(x) \quad x \in \Omega, f \in H \quad (3)$$

The functions $K(x, \cdot)$ are the Reisz representers of the linear point functionals $\delta_x : f \mapsto f(x)$ in the dual space $H^*$ of $H$. Therefore we have the relationship (assuming the traditional definition of norm in the dual space)

$$K(x, y) = < K(x, \cdot), K(y, \cdot) >_H = < \delta_x, \delta_y >_{H^*} \quad x, y \in \Omega \quad (4)$$

For observed data at $X = \{x_1, x_2, ..., x_n\}$, one can consider the space of trial function $D_X = \text{span}\{K(\cdot, x_j) : x_j \in X\}$ as a choice for the approximation space. However from work such as \cite{22, 24} we know that the bases formed by translates of the kernel functions are highly dependent on the distribution of data points in $X$ and hence numerically ill-conditioned for many problems. For dealing with this issue, at each scale $s$, our approach considers a subset of original trial functions as bases such that

$$\Gamma^s = \text{span}\{K^s(\cdot, x_i) : x_i \in X_s\} \approx \text{span}\{K^s(\cdot, x_j) : x_j \in X\}, \quad X_s \subseteq X \quad (5)$$

These linearly independent, data derived basis functions are sampled using a Pivoted QR strategy as detailed in the following section (Algorithm 1). It should be noted here that at each scale we have a different kernel function $K^s$. Hence, effectively we are working in a different RKHS at every scale (from the uniqueness property \cite{3} of kernel functions). Let this scale dependent RKHS be represented as $\mathcal{H}_s$. Therefore at any scale $s$, the function to be approximated ($f$) is assumed to be in the RKHS $\mathcal{H}_s$. However, our approach is aimed at finding an approximation $A_s f \in \Gamma^s$ to $f$ as in (5). Hence, if we look at the problem formulation (2), then $\mathcal{H}_s$ is the space $\mathcal{V}$ and $\Gamma^s$ is our approximation space $Q$. While our overall approaches

\cite{2}Theorem 2 in [33]
are somewhat related to statistical emulators [32], the notion of scale is rarely articulated in that domain.

The research problem we handle here is fairly general and has close relationships with many other research areas. Multiresolution Analysis (MRA) implementing wavelet decomposition serves as one of the earliest literature on multiscale processing of datasets ([39], [21]). Multiscale modeling also has close relationships with geometric analysis ([18], [19], [20], [38], [48]) involving diffusion maps to analyze datasets at different scales. The inherent hierarchy in our approximation space is also motivated from a wide literature on Hierarchical Radial Basis Functions (HRBF) ([26], [8], [4], [25]). The combination of basis functions at different scales to produce desired approximation, forms the foundation of the majority of this literature. This is very similar to the idea of multiscale analysis explored in [5] and [6]. However, in the current work we show that this maybe an overkill (in terms of computation and generalization) in many of the situations, and instead we focus on one particular optimal scale which is able to model the dataset efficiently and also allows data reduction. Hierarchical approximations also find applications in data visualization [51]. Since our approach generates data driven hierarchical basis functions belonging to RKHS, therefore the proposed approach is also related to works such as [13] and [2], where the authors consider data dependent multiscale dictionaries that generalize wavelets in geometric sense with the capability to analyze non-linear geometries. The literature on modeling large physical systems through partial differential equations have utilized the idea of multilevel analysis through multigrid methods ([10], [46]). Following this idea, many research works have extended the multigrid methods to general field of machine learning and data analysis ([29], [36] [44]) closely relating it to our approach. Since our approach generates data driven basis functions centered at the produced sparse representations (essentially producing data driven unstructured grids), thus is also related to methods implementing sparse grids ([7], [31]) for dimensionality reduction of datasets. The hierarchical approach which we discuss here samples the sparse bases set using a pivoted QR decomposition. However, there is also a rich body of literature on Column Subset Selection (CSS) problem ([17], [9], [40], [23]) that can be connected with this work leading to approximate (less accurate than pivoted QR) but faster methods for generating the sparse representation.

Overall, the contributions of this paper can be summarized as follows.

- Firstly we introduce an approach which computes a scale \( s \) dependent sparse representation \( D^s_{\text{sparse}} \) of a large dataset. Besides data reduction, the algorithm also provides a model for approximations at new data points in the observation space using just \( D^s_{\text{sparse}} \), enabling any sort of prediction from a compressed version of the dataset.
- We then provide detailed analysis on convergence, establishing the optimality of the solution obtained by the algorithm, computation of point-wise error functionals and their behavior which governs the performance of the approach.
- At each scale, besides providing confidence bounds which quantify our belief in the estimation, we also provide prediction intervals which estimates the bounds in which the algorithm believes any observations not included in the sparse representation as well as any new samples to be made in the future will lie. This also makes the sparse representation more useful.
- Towards the end, we have also provided stability estimates to further establish the dependability of the approach.
2. The hierarchical approach. Developing further on the work of [5], we introduce a methodology for data reduction through efficient bases construction, exploiting the multilevel nature of the correlation structure in the data. The basic steps involved in the approach are presented as Algorithm 1, here.

**Algorithm 1 Hierarchical approach**

1. **INPUT:**
   - Hyperparameters: $[(TOL \geq 0, T > 0, P > 1) \in \mathbb{R}^3]$
   - Dataset: $[(X \in \mathbb{R}^{n \times d}) \subset \Omega, (f|_X \in \mathbb{R}^n)]$
   - Prediction points: $[(X_\ast \in \mathbb{R}^{n \times d}) \subset \Omega]$

2. **OUTPUT:**
   - Convergence Scale: $[S_n \in \mathbb{N}]$
   - Sparse Representation ($D^S_{\text{sparse}}$): $[(X_{S_n} \in \mathbb{R}^{(l_{S_n} \times d)}), (C_{S_n} \in \mathbb{R}^{l_{S_n}})]$
   - Predictions: $[P^* \in \mathbb{R}^n]$

3. Initialize: $s = 0$
4. **while** TRUE do
5. Compute covariance kernel: $[G_s \in \mathbb{K} / \mathbb{X} \in \mathbb{K} (\epsilon_s = T/P^*)]$
6. Compute numerical Rank: $[l_s = \text{rank}(G_s)]$
7. Remove sampling Bias: $[(W = AG_s) \in \mathbb{A}^{k \times n} \text{ and } (a_{i,j} \sim N(0,1))]$
8. Generate permutation information: $[WP_R = QR]$
9. Produce bases at scale s: $[B^s = (G_s P_R)[; : l_s]], \text{ with } B^s \in \mathbb{R}^{n \times l_s}]
10. Subset the sparse representation in $X_s$
11. Compute projection coordinate: $[C_s = (B^s)\dagger f|X; B^s\dagger = (B^s T B^s)^{-1} B^s T]$
12. Generate approximation at s: $[(A_s f)|X = B^s C_s]\$
13. If $[\|f|X - (A_s f)|X\|_2 \leq TOL] : S_n = s; \text{ Break}$
14. Update scale: $[s = s + 1]$
15. **end while**
16. Compute bases for prediction at $X_s$: $[G_{S_n} \text{ centered at } X_{S_n} \text{ with } (\epsilon_{S_n} = T/P^{S_n})]$
17. Predict: $P^* = G_{S_n}^\dagger, C_{S_n}$

2.1. **Proposed algorithm.** The proposed approach (Algorithm 1) constructs a sequence of scale ($s$) dependent approximations $A_1 f, A_2 f, \ldots, A_s f$, to the unknown function $f : \mathbb{R}^d \rightarrow \mathbb{R}$ using only use a subset of dataset $X_1, X_2, \ldots, X_s$, respectively. Algorithm 1 begins by taking a dataset, where a data point $x_i \in \mathbb{R}^d$ is mapped to a functional value $f(x_i) \in \mathbb{R}$. In matrix form $f|_X = \{f_1, f_2, \ldots, f_n\} \text{ values are obtained at data points } X = \{x_1, x_2, \ldots, x_n\} (f|_X \in \mathbb{R}^n \text{ and } X \in \mathbb{R}^{n \times d})$. The scalars $[T, P, TOL] \in \mathbb{R}^3$ are the algorithmic hyperparameters defined by the user. TOL is the simple 2-norm error tolerance, P is assumed to be 2 (Based on [5]). This choice of P reduces the length scale of the kernel (K) by a factor of 0.5 at each scale increment, providing an intuitive understanding of how the support of basis functions is adapted to scale variation. If we assume the diameter of the dataset to be distance between the most distant pair of datapoints, then T is given by

$$T = 2(Diameter(X)/2)^2 \quad (6)$$

Besides these hyperparameters, the algorithm also accepts the prediction points $X_\ast \subset \Omega \subset \mathbb{R}^d$, which represent the data points at which the user wants to approximate the underlying function. We also need to choose the positive definite function
(\(K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}\)). For this paper we use the squared exponential (also referred to as Gaussian) kernel (7) for mapping the covariance structure and generating the space of trial functions \(\Gamma^s(5)\) at each scale \(s\).

\[
G_s(x, y) = \exp \left(-\frac{||x - y||^2}{\epsilon_s}\right); \quad \epsilon_s = \frac{T}{f_s^s} \tag{7}
\]

Here \(\epsilon_s\) (also known as the length scale parameter) determines the width of the bases support at a particular scale. This squared exponential kernel is very widely used in the Gaussian process literature [43] and is a popular choice for most spatial statistics problems. The learning phase (for generating \(D^s_{\text{sparse}}\)) of the proposed algorithm has been explained in \(\text{STEP } 1\) and \(\text{STEP } 2\) below. \(\text{STEP } 3\) constitutes the prediction phase which uses the produced sparse representation for the dataset. It should be noted that, it is not required to wait till the convergence of the algorithm to go to the prediction phase. In fact, it is possible to predict from the sparse representation at each scale \((D^s_{\text{sparse}})\). We have shown the prediction phase after convergence for the purpose of clarity.

\section*{STEP-1 (Getting sparse data-\(X_s\))}

Given the Dataset \(D = [X, f|\chi]\), the algorithm begins with the computation of the covariance operator \(G_s\) (7). However, based on [22, 24], the distribution of the dataset might lead to ill-conditioning of this covariance kernel. Therefore we carry out a column pivoted QR decomposition to identify the space \(\Gamma^s\) (at each scale) which represents the span of the trial functions \(K^s(\cdot, x_j), (1 \leq j \leq n)\) at scale \(s\). The QR decomposition is carried out on \(W\) for obtaining the Permutation matrix \(P_R\). \(W\) is produced by the product of a random normal matrix \(A\) with the \(G_s\). Here we have \(A \in \mathbb{R}^{l_s \times n}\) with \(l_s = \text{rank}(G_s)\) and \(k = l_s + 8\) (as in [5]) giving us 8 additional rows to account for numerical round-offs during the QR decomposition. However, this is a conservative step and even without any additional sampled rows, the algorithm was found to perform well. The permutation matrix \(P_R\) produced by the decomposition captures the information content of each column of \(W\). \(P_R\) is then used to extract independent columns with the biggest norm contributions along with the observation points these columns correspond to in the covariance kernel \((G_s)\). This set of sampled observations from the original dataset is termed as the corresponding sparse data \((X_s)\). The number of columns sampled from \(G_s\) come from its numerical rank estimated by using strategies such as a \textit{Rank Revealing QR} or a SVD decomposition.

\section*{STEP-2 (Getting projection coordinate-\(C_s\))}

Once, we have the relevant columns of \(G^s\) (denoted as \(B^s\)) which also represent the approximation subspace \(\Gamma^s\) which spans \(D_X\), the algorithm proceeds to solve the over-constrained system \((B^sC_s = f|\chi)\). We can think of it as an orthogonal projection problem where \(f|\chi\) needs to be projected on the column space of \(B^s\) and then it is required to compute the specific weighting of vectors in the basis matrix \(B^s\) which produces this projection. The Algorithm then computes the orthogonal projection \((|A_s f|\chi)\) given the required coordinates and basis vectors in \(B^s\). Once we have the scale \(s = S_a\) at which the algorithm satisfies the 2-norm condition, it produces the sparse representation \(D^s_{\text{sparse}} = [X_{S_a}, C_{S_a}]\) and proceeds to the prediction stage if required by the user. Here \(S_a\) is called the convergence scale as detailed in the following subsection.

\section*{STEP-3 (Prediction at \(X_s\) from \(D^s_{\text{sparse}}\))}

For computing functional values at unobserved location \(X_s\), basis functions are constructed by computing \(G^s_{S_a}\) (Gaussian kernel for the prediction points with the sparse data - \(X_{S_a}\)). The Prediction step
weighs the constructed bases with coordinates of orthogonal projection \( C_s (C_{S_a}) \), if the prediction is being made at the convergence scale) and linearly combines them to produce the required approximation.

Finally, before moving forward, it is worth mentioning here that the 2-norm criteria is just one of the many possible kinds of norms which can be used to measure the scale dependent approximation error.

2.2. Critical scale \((S_c)\) and convergence scale \((S_a)\). In our work, the scale at which the kernel matrix becomes numerically full rank is referred to as Critical Scale. Working with proposition 3.7 in [5], if \( \delta \) represents the precision of rank for the Gaussian kernel matrix, then we can define the numerical rank of the kernel as

\[
R_\delta(G_s) = \# \left( j : \frac{\sigma_j(G_s)}{\sigma_0(G_s)} \geq \delta \right)
\]

where \( \sigma_j(G_s) \) is the \( j^{th} \) largest singular value of \( G_s \). Also let \( |I_i| \) represents the length of the bounding box of the data in \( i^{th} (i \in [1,d]) \) dimension. Then given the length scale parameter \( \epsilon_s \), the rank of the Gaussian kernel can be bounded above as

\[
R_\delta(G_s) \leq \prod_{i=1}^{d} \left( \frac{2|I_i|}{\pi} \sqrt{\epsilon_s^{-1} \ln(\delta^{-1}) + 1} \right)
\]

Proposition 3.7 in [5] states that numerical rank of the Gaussian kernel matrix is proportional to the volume of the minimum bounding box \( B = I_1 \times I_2 \times ... \times I_d \) and to \( \epsilon^{-d/2} \). Therefore for a fixed data distribution, following relation holds

\[
R_\delta(G_s) \propto \epsilon^{-d/2} \propto P^{sd}
\]

Hence (for \( P = 2 \) and a fixed \( d \)) numerical rank of the Gaussian covariance kernel increases exponentially with the scale of study. Therefore, there exists a minimum scale \( s = S_c \), at which the kernel becomes full and continue to stay full rank as the scale is further increased. This scale is referred to as the Critical Scale \((S_c)\). Hence, based on the overall Algorithm 1, if \( X_1, X_2, ..., X_s, ... \) and so on, are the sampled sparse data at scale \( s \), they satisfy \( |X_1| \leq |X_2| \leq |X_3| \leq ... \leq |X_{S_c}| \), where \(|·|\) denotes the cardinality of a set. Therefore with increasing scales more data points are added, leading to better approximations. Thus, for any scales \( j \) and \( i \) satisfying the relation \( j \geq i \), the following relation holds

\[
||f||_X - (A_j f)|_X \|_2 \leq ||f||_X - (A_i f)|_X \|_2
\]

From approximation theory [15], we can now state that for kernel at the Critical Scale and a finite point set \( X \subset \Omega \): (a) \( G_{S_c} \) is positive definite, (b) If \( (A_{S_c} f) \in \Gamma^{S_c} \) vanishes on \( X \), then \( (A_{S_c} f) \equiv 0 \), and, (c) the approximation \( (A_{S_c} f) \) satisfying the following relation is unique

\[
(A_{S_c} f)|_X = f|_X
\]

We can now define the Convergence scale \((S_a)\) as the minimum scale \( s \) which satisfies

\[
||f||_X - (A_s f)|_X \|_2 \leq TOL
\]

Therefore \( S_a \) is the scale at which Algorithm 1, stops. It is worth noting here that based on the definition of \( S_c \) and \( S_a \), we have \( S_a \leq S_c \) and \( |X_{S_a}| \leq |X_{S_c}| \).
3. Convergence properties. Following the notations introduced in the previous sections, again let \( f \) be the function which needs to be approximated and \( A_s f \) be the approximation produced by the Algorithm 1 at scale \( s \). Correspondingly, let \( E_s^* = f - A_s f \) be the error in approximation at scale \( s \) with its 2-norm in \( \mathbb{R}^n \) as \( \|E_s^*|_2 = \|f|_2 - \|A_s f|_2 \). 

**Lemma 3.1.** For any choice of \( \text{TOL} \geq 0 \) and any norm: \( \|\cdot\| \) in \( \mathbb{R}^n \) chosen by the user, the proposed algorithm (Algorithm 1) converges in the sense 

\[
\|E_s^*|_2 = \|f|_2 - \|A_s f|_2 \leq \text{TOL} \tag{12}
\]

for some finite \( s \leq S_c \) and the corresponding produced approximation \( A_s f \). 

**Proof.** \( (A_s f)|_2 \) is the projection of \( f|_2 \) on the bases at scale \( s \) with the projection operator \( B_s B_s^T = R_s \), where \( B_s^T \) is the pseudo-inverse of basis matrix \( B_s \). Therefore 

\[
\|E_s^*|_2 = \|f|_2 - \|R_s f|_2 \|, \quad \text{where} \quad \| \cdot \| \text{is the 2-norm}\] 

From Equation (10), we know that numerical rank of bases \( B_s \) increases monotonically with scale \( s \). Hence as \( s \) approaches \( S_c \), \( R_s \) approaches \( I_n \). Thus \( \|I_n - R_s\|_2 \to 0 \) and hence \( \|E_s^*|_2 \to 0 \). This establishes the convergence of the proposed approach in 2-norm at some finite scale \( S_a \leq S_c \). From equivalence of norms on finite dimensional vector space \([35]\), this generalizes the convergence guarantee to all possible norm in \( \mathbb{R}^n \). 

In the following theorem we analyze a particular hierarchical updating scheme. 

**Theorem 3.2.** If the projection update of Algorithm 1 is written in an iterative form 

\[
(A_{s+1} f)|_X = (A_s f)|_X + \alpha_s \cdot E_s^*|_X \tag{14}
\]

where \( E_s^* = f - A_s f \). Then \( \alpha_s \) follows the bounds \( 0 < \alpha_s < 2 \) and convergence rate \( \rho_s = |1 - \alpha_s| \). 

**Proof.** Since at each scale \( s \), \( A_s f \) is generated as projection of \( f \) on space 

\[
\Gamma_s = \text{span}\{K^*(\cdot, x_i) : x_i \in X_s\} \quad X_s \subseteq X
\]

Therefore using the notation \( E_s^* = f - A_s f \), we write this update in an iterative form 

\[
(A_{s+1} f)|_X = (A_s f)|_X + \alpha_s \cdot E_s^*|_X \tag{15}
\]

From the definition of \( E_s^*|_X \), it follows that any convergent algorithm requires \( \alpha_s \geq 0 \). Rearranging (15) and taking an inner product with respect to \( E_s^*|_X \)

\[
\alpha_s = \frac{\langle (A_{s+1} f - A_s f)|_X, E_s^*|_X \rangle}{\langle E_s^*|_X, E_s^*|_X \rangle} = \frac{\langle (A_{s+1} f - A_s f)|_X, E_s^*|_X \rangle}{\|E_s^*|_X\|^2}
\]

(assuming \( \|E_s^*|_X\|^2 > 0 \)). Now, using the Cauchy Schwarz and Triangle inequalities 

\[
\alpha_s \leq \frac{\|A_{s+1} f - A_s f|_X\| \cdot \|E_s^*|_X\|}{\|E_s^*|_X\|^2} = \frac{\|A_{s+1} f - A_s f|_X\|}{\|E_s^*|_X\|} \leq 1 + \frac{\|E_{s+1}^*|_X\|}{\|E_s^*|_X\|}
\]

Therefore we get \( 0 \leq \alpha_s \leq 1 + \frac{\|E_{s+1}^*|_X\|}{\|E_s^*|_X\|} \). Now, for convergent approximations \( \|E_{s+1}^*|_X\| \leq \|E_s^*|_X\| \). Therefore for all scales, \( \alpha_s \in [0, 2] \). The end members \( \alpha_s = \{0, 2\} \) imply the iterative scheme has converged and no improvement is needed (as reflected in the stopping criterion in Algorithm 1). Hence, we remove these end
Beginning with the Inner products in the RKHS

Proof.

Now we analyze the inner product of the error of the projection \( E_s = f - A_s f \) with respect to projection at scale \( s \) in the native RKHS. The major power of the following theorem lies in the fact that as the proposed algorithm converges, we are able to upper bound this inner product as a function of the user defined tolerance (TOL). This provides the user direct control on the approximation properties on the algorithm even in the native RKHS. It should be noted again that at each scale \( s \), it is assumed that the function \( f \) to be approximated belongs to the same Hilbert space \( H_s \) and is approximated in \( \Gamma^s \), justifying the use of the reproducing property.

**Theorem 3.3.** For observed data \( f|_X = (f(x_1), f(x_2), \ldots f(x_n))^T \in \mathbb{R}^n \) on \( X \subseteq \Omega \), the approximation \( A_s f \) and its corresponding error \( E_s \) at convergence scales \( s = S_a \) satisfies

\[
\left\langle (A_{S_a} f), E_{S_a} \right\rangle_{H_{S_a}} \leq ||C_{S_a}||_{\infty} \sqrt{n}(TOL)
\]

where \( C_{S_a} = (|c_1|, |c_2|, \ldots |c_{S_a}|)^T \in \mathbb{R}^{S_a} \) contains the modulus of coordinates of projection at \( S_a \), \( n \) is the number of observations and TOL is the 2-norm convergence error tolerance in Algorithm 1.

**Proof.** Beginning with the Inner products in the RKHS \( \left\langle (A_s f), E^s \right\rangle_{H_s} \) at scale \( s \)

\[
\left\langle (A_s f), E^s \right\rangle_{H_s} = \left\langle \sum_{x_j \in X_s} K^s(\cdot, x_j) c_j, f - A_s f \right\rangle_{H_s}
\]

\[
= \left\langle \sum_{x_j \in X_s} K^s(\cdot, x_j) c_j, f \right\rangle_{H_s} - \left\langle \sum_{x_j \in X_s} K^s(\cdot, x_j) c_j, A_s f \right\rangle_{H_s}
\]

\[
= \sum_{x_j \in X_s} f(x_j) c_j - \sum_{x_j \in X_s} (A_s f)(x_j) c_j \quad \text{(Reproducing Property)}
\]

\[
\leq \sum_{x_j \in X_s} c_j \left| \left| f(x_j) - (A_s f)(x_j) \right| \right| \quad \text{(Triangle Inequality)}
\]

\[
\leq ||C_s||_{\infty} \sum_{x_j \in X_s} \left| \left| f(x_j) - (A_s f)(x_j) \right| \right| \quad (X_s \subseteq X)
\]

\[
= ||C_s||_{\infty} ||E^s|_X||_1
\]

Here \( ||C_s||_{\infty} \) is the \( \infty \) norm of \( C_s = [|c_1|, |c_2|, \ldots, |c_{S_a}|]^T \).

\[
\implies | \langle (A_s f), E^s \rangle_{H_s} | \leq ||C_s||_{\infty} ||E^s|_X||_1 \leq ||C_s||_{\infty} \sqrt{n} ||E^s|_X||_2 \quad \text{(Cauchy-Schwarz Inequality)}
\]

However, from the convergence criteria in the proposed Algorithm 1, we know \( ||E^s|_X||_2 \leq TOL \) at \( s = S_a \). Hence the theorem follows
4. Approximation properties and confidence intervals. This section provides estimates quantifying the quality of approximations generated by the proposed algorithm at each scale. The first result is a direct application of theorem (3.3)

4.1. Approximation properties.

**Corollary 1.** For observed data \( f|_X = (f(x_1), f(x_2), \ldots, f(x_n))^T \in \mathbb{R}^n \) on \( X \in \Omega \), the approximation \( (A_{s_a}f) \) produced by Algorithm 1 at the convergence scale \( (S_a) \), follows the Pythagoras Theorem in the limit \( \text{TOL} \to 0 \). i.e,

\[
||f||_{\mathcal{H}_{s_a}}^2 = ||A_{s_a}f||_{\mathcal{H}_{s_a}}^2 + ||f - A_{s_a}f||_{\mathcal{H}_{s_a}}^2
\]

**Proof.** Let \( f \) be the function to be approximated. As stated earlier, it was discretely observed at \( X \in \Omega \) leading to the restricted function \( f|_X \). Starting with the norm of the function in the Hilbert Space

\[
||f||_{\mathcal{H}_s}^2 = (A_s f, f)_s + ||f - (A_s f)||_{\mathcal{H}_s}^2 < (A_s f, f - (A_s f))_s \]

Using theorem 3.3, at \( s = S_a \) in the limit \( \text{TOL} \to 0 \), the result follows \( \square \)

Now, we will provide results related to uniqueness and quality of solution.

**Theorem 4.1.** For observed data \( f|_X = (f(x_1), f(x_2), \ldots, f(x_n))^T \in \mathbb{R}^n \) on \( X \in \Omega \), the approximation \( (A_s f) \in \Gamma^s \) in the limit \( \text{TOL} \to 0 \) produced by the proposed algorithm at the convergence scale \( s = S_a \) is

1. the unique orthogonal projection to \( f \)
2. the unique best approximation to \( f \) with respect to \( || \cdot ||_{\mathcal{H}_s} \)

**Proof.** (1): \( A_s f \) is a unique orthogonal projection of \( f \) on \( \Gamma^s \) if \( (f - A_s f) \perp \Gamma^s \). Again using the reproducing property of RKHS

\[
\left( \sum_{x_j \in X_s} c_j K^s(\cdot, x_j), f - A_s f \right)_{\mathcal{H}_s} = \left( \sum_{x_j \in X_s} c_j K^s(\cdot, x_j), f \right)_{\mathcal{H}_s} - \left( \sum_{x_j \in X_s} c_j K^s(\cdot, x_j), A_s f \right)_{\mathcal{H}_s} = \sum_{x_j \in X_s} c_j f(x_j) - \sum_{x_j \in X_s} c_j (A_s f)(x_j)
\]

The rest of the proof is similar to steps in the proof of theorem 3.3 showing the considered native Hilbert space inner product vanishes as \( \text{TOL} \to 0 \)

(2): Given \( A_s f \) as the unique orthogonal projection to \( f \), result in part 2 follows from property of projections covered in standard approximation theory [1]. \( \square \)

Now, we will analyze the error functional and pointwise error bounds. At scale \( s \), we are searching for a solution in the space \( \Gamma^s \). Our approximation is of the form

\[
(A_s f)(x) = \sum_{x_j \in X_s} K^s(x, x_j) c_j
\]

Any \( A_s f \in \Gamma^s \) can also be expressed as

\[
(A_s f)(x) = \lambda^y K^s(x, y); \quad \text{where } \lambda = \sum_{x_j \in X_s} c_j \delta_{x_j}
\]

\( \delta_x \) is the evaluational functional for \( f \), i.e. \( \delta_x(f) = f(x) \) and \( \lambda^y \) is application of linear functional \( \lambda \) on \( y \). Thus, we have the dual space \( \mathcal{H}^*_s = \left\{ \sum_{x_j \in X_s} c_j \delta_{x_j} \right\} \).
Next, we present a result which provides an inner product representation for approximation at a point $x \in \Omega$. The objective here is to show the optimality of the approximation $A_s f$ by requiring vanishing gradient for the norm of the Error Functional in $\mathcal{H}_s$.

**Theorem 4.2.** If the error functional at scale $s$ is represented as $\varepsilon_s^x = \delta_s^x - M_s(x)^T \delta_X(x)$ such that

$$|\varepsilon_s^x(f)| = |\delta_s^x(f) - M_s(x)^T \delta_X(x)|$$

with assumed differentiability of $M_s(x) = [M_1^s(x), M_2^s(x), M_3^s(x), ..., M_n^s(x)]^T \in \mathbb{R}^n \forall x$. Then the minimizer $(\hat{M}_s(x))$ of $||\varepsilon_s^x||_{\mathcal{H}_s}$ satisfies the inner product representation

$$<f|_{X_s}, \hat{M}_s(x) >= (A_s f)(x)$$

establishing the optimality of approximation $(A_s f)$ at each scale $s$.

**Proof.** We begin by expressing the error functional norm

$$||\varepsilon_s^x||_{\mathcal{H}_s}^2 = \delta_s^x M_s(x)^T \delta_X(x) - 2M_s(x)^T \delta_s^x \delta_X(x) + M_s(x)^T \delta_X(x) \delta_s^x M_s(x)$$

Now, based on the property of dual space, we know at scale $s$,

$$<\delta_s^x, \delta_s^x >_{\mathcal{H}_s} = K^s(a, b)$$

Also, let

$$R_s(x) = \delta_X^s \delta_s^x = (K^s(x, x_1), K^s(x, x_2), ..., K^s(x, x_n))^T \in \mathbb{R}^n$$

$$G_s = \delta_X^s \delta_s^T_X \quad \text{(Gaussian kernel at scale s)}$$

Therefore, we have

$$||\varepsilon_s^x||_{\mathcal{H}_s}^2 = ||\delta_s^x||_{\mathcal{H}_s}^2 - 2M_s(x)^T R_s(x) + M_s(x)^T G_s M_s(x)$$

(20)

Differentiation yields the optimal $\hat{M}_s(x)$ that minimizes $||\varepsilon_s^x||_{\mathcal{H}_s}^2$, as the the solution of

$$G_s \hat{M}_s(x) = R_s(x)$$

(21)

From Algorithm 1, we know $G_s$ need not be full rank. Using matrix $A$ (as in Algorithm 1), constructing $W(= A G_s)$ and a Column pivoted QR decomposition $W P_s = QR$, and, applying the permutation operator $P_s$ on equation (21) yields

$$P_s R_s(x) = P_s G_s \hat{M}_s(x)$$

(22)

Since $G_s$ is a symmetric operator, sampling the first $l_s$ rows of $P_s G_s$, where $l_s$ represents the numerical rank of $G_s$, will produce $B^{sT}$, the transpose of the bases considered at scale $s$. Correspondingly, sampling the respective values in $R_s(x)$ produces the restriction of $R_s(x)$ to set $X_s$ (represented as $R_s(x)|_{X_s}$). Therefore, restricting system (22) to equations only corresponding to $X_s$

$$R_s(x)|_{X_s} = B^{sT} \hat{M}_s(x)$$

(23)

where we use the Moore-Penrose inverse for getting the optimal projection of $R_s(x)|_{X_s}$. Thus, optimal solution for $M_s(x)$ is given by

$$\hat{M}_s(x) = B^s (B^{sT} B^s)^{-1} R_s(x)|_{X_s} = B^{sT} R_s(x)|_{X_s}$$

(24)
Theorem 4.3. For observed data $f|_X = (f(x_1), f(x_2), \ldots, f(x_n))^T \in \mathbb{R}^n$ on $X \subset \Omega$, there exists at each scale, a point approximation functional $(\psi_x^*)$ for the optimal approximation $A_x f$ such that $\psi_x^*(f) = (A_x f)(x)$. This functional has a representation

$$\psi_x^* = R_s^T(x)|_X, B^{s^*} \delta_X^s$$

(25)

Also, it produces the minimized error functional in the sense

$$\min \|\varepsilon_x^*\|_{H^*_{\delta}} = \min \|\delta^*_x - \gamma^*_x\|_{H^*_{\delta}}$$

with optimal: $\gamma^*_x = \psi^*_x$

(26)

Proof. We begin with the expression for the norm of the error functional in equation (20)

$$\|\varepsilon_x^*\|_{H^*_{\delta}} = \|\delta^*_x\|_{H^*_{\delta}}^2 - 2M_s(x)^T R_s(x) + M_s(x)^T G_s M_s(x)$$

since, it is minimized when $M_s(x)$ is given by equation (24). Now, writing the optimal approximation functional at $x$ as:

$$\psi_x^*(f) = A_x f(x) = M_s^T(x) \delta_X^s(f) = R_s^T(x)|_X, B^{s^*} \delta_X^s(f)$$

which completes the proof for equation (25).

Now, considering the error functional at $x$ and putting the optimal $M_s(x)$ in equation (20), we get

$$\min \|\varepsilon_x^*\|_{H^*_{\delta}} = \|\delta^*_x\|_{H^*_{\delta}}^2 - 2R_s^T(x)|_X, B^{s^*} R_s(x) + R_s^T(x)|_X, B^{s^*} G_s B^{s^* T} R_s(x)|_X$$

Recognizing $\psi_x^*$ from equation (25) and substituting in the above equation.

$$\min \|\varepsilon_x^*\|_{H^*_{\delta}} = \|\delta^*_x\|_{H^*_{\delta}}^2 - 2 < \psi_x^*, \delta^*_x >_{H^*_{\delta}} + \|\psi_x^*\|_{H^*_{\delta}}^2 = \|\delta^*_x - \psi_x^*\|_{H^*_{\delta}}^2$$

Which completes the proof for equation (26).

Before moving forward, here we also provide a result for bounding the Error functional.

Corollary 2. For the error functional defined in Theorem 4.2, the absolute error at $x$ is bounded in the sense

$$|\varepsilon_x^*(f)| \leq \sqrt{1 - M_s(x)^T R_s(x)} \|f\|_{H_s}$$

(27)

Proof. Using the relation,

$$|\varepsilon_x^*(f)| \leq \|\varepsilon_x^*\|_{H^*_{\delta}} \|f\|_{H_s}$$

again using equation (20)

$$\|\varepsilon_x^*\|_{H^*_{\delta}} = \|\delta^*_x\|_{H^*_{\delta}}^2 - 2M_s(x)^T R_s(x) + M_s(x)^T G_s M_s(x)$$
Now, since (20) is minimized when we satisfy (21). Also since, \( \| \delta_1^s \|_{H^s_1}^2 = 1 \) (by assumed normalization). Putting the values in (20) we get

\[
\min \| \varepsilon_x^s \|_{H^s_2}^2 = 1 - M_s(x)^T R_s(x)
\]

Therefore result in equation (27) follows. 

4.2. Confidence and prediction intervals. Confidence intervals in general are a measure of our belief in the estimated approximation. Prediction intervals on the other hand refer to the bounds around the mean fit, where a future datapoint is expected to fall. This is crucial information in conjunction with the sparse representation, as even when we are not able to capture the function accurately at initial scales, we have an estimate of the expected behavior of the observations. Algorithm 1 makes predictions at each scale based on the corresponding sparse representation \( D^s_{\text{pars}} \). Here if the error in approximation is greater than user defined tolerance, i.e. \( \| f X - (A_s f) X \|_2 > \text{TOL} \), then it signifies that Approximation \( A_s f \) needs additional degrees of freedom to capture the underlying data generation process (as in principle it is the best possible approximation, given a basis set). For modeling this error value, we consider a model of form (where \( f X \) is considered as a noisy realization of a true underlying process at a fixed and finite \( X \))

\[
f(x_i) = (A_s f)(x_i) + \epsilon_x^s \quad \text{with} \quad x_i \in X \quad \text{and} \quad \epsilon_x^s \sim N(0, \sigma^2_x I)
\]  

(28)

Therefore the sampling distribution for \( f X \) would be given as

\[
p\left( f X \mid (A_s f) X, \epsilon_x \right) \sim N(B^s C_s, \sigma^2_x I)
\]

Since, we know that the projection coordinate is given as \( C_s = (B^s T B^s)^{-1} B^s T (f X) \), the covariance matrix is

\[
\text{Cov}(C_s) = (B^s T B^s)^{-1} B^s T \text{Cov}(f X) B^s (B^s T B^s)^{-1} = \sigma^2_x (B^s T B^s)^{-1}
\]

\[
\mathbb{E}[C_s] = (B^s T B^s)^{-1} B^s T \mathbb{E}[f X] = (B^s T B^s)^{-1} B^s T B^s C_s = C_s
\]

which shows an unbiased estimator. Here it should be noted that since we are modeling \( f X \) (given a fixed \( X \)), as a stochastic process. So \( \mathbb{E}[C_s] = C_s \) is true in asymptotic sense with expectation taken over the infinite realizations of \( f X \) given a fixed \( X \). Hence \( C_s \) is the best unbiased coordinate approximation at scale \( s \) given bases \( B^s \). Now for computing the confidence and prediction intervals, we start by computing the prediction basis \( (B^*_s) \) at \( x_s \in \Omega \), which is expressed as a set of Gaussians from equation (7) centered at the sparse set \( X_s \) and computed with respect to \( x_s \) (resulting in a row vector of length \( l_s \)). Then we obtain the mean and variance of the underlying process as follows

\[
\mathbb{E}[A_s f(x_s)] = B^*_s \mathbb{E}[C_s] = A_s f(x_s)
\]

\[
\text{Var}(A_s f(x_s)) = B^*_s \text{Cov}(C_s) B^*_s = \sigma^2_x B^*_s (B^s T B^s)^{-1} B^*_s T
\]

For an estimated value of \( \sigma \) (\( \hat{\sigma} \)), we can write the corresponding standard deviation at \( x_s \)

\[
\text{stdev}(A_s f(x_s)) = \hat{\sigma}_s \sqrt{B^*_s (B^s T B^s)^{-1} B^*_s T}
\]

Therefore if we use t-distribution for confidence bounds, we get the following \( 100(1 - \alpha)\% \) confidence intervals for \( \mathbb{E}[A_s f(x_s)] \)

\[
(A_s f)(x_s) \pm \text{stdev}(A_s f(x_s)) \cdot t \left(1 - \frac{\alpha}{2}, n - l_s\right)
\]

(29)
Here $l_s$ is again the numerical rank of $G_s$ and $n$ is the original number of observations made. Now for prediction interval, we know, $\text{var}(f(x_s) - (A_s f)(x_s)) = \text{var}(\epsilon^s) + \text{var}(A_s f(x_s))$, therefore,

$$\text{stdev}(f(x_s) - (A_s f)(x_s)) = \hat{\sigma}_s \sqrt{1 + B_s^* (B_s^T B_s)^{-1} B_s^T}$$

Hence the $100(1 - \alpha)\%$ prediction intervals are given as

$$(A_s f)(x_s) \pm \text{stdev}(f(x_s) - (A_s f)(x_s)) \cdot t\left(1 - \frac{\alpha}{2}, n - l_s\right)$$

For the estimated value of $\sigma^2$, we use its unbiased estimation at scale $s$ given as

$$\hat{\sigma}^2 = \frac{\|f|_X - A_s f|_X\|^2}{n - l_s}$$

5. Stability properties. In this section we provide bounds related to stability of approximations obtained by the proposed algorithm. The first result bounds the approximation at scale $s$ with respect to the $L_\infty$ topology for some compact domain $\Omega \in \mathbb{R}^d$.

**Theorem 5.1.** For observed data $f|_X = (f(x_1), f(x_2), \ldots f(x_n))^T \in \mathbb{R}^n$ on $X \subset \Omega$, the approximation $A_s f$ at any scale $s$ is bounded in the $L_\infty(\Omega)$ norm as

$$\|A_s f\|_{L_\infty(\Omega)} \leq P_s^\infty \|f\|_{L_\infty(\Omega)}$$

where $P_s^\infty$ is obtained by applying the extension operator $B_s^T$ on $G_{X_s} = R_s(x)|_X R_s(x)|_{X_s}$, i.e. $D_s = B_s^T G_{X_s} B_s^T$. 

**Proof.** Expressing $(A_s f)$ in terms of the inner product as in equation (19).

$$\|A_s f\|_{L_\infty(\Omega)} = \max_{x \in \Omega} |A_s f(x)| = \max_{x \in \Omega} \left| \sum_{j \in X} \langle f(x_j) M_s^j(x) \rangle \right|$$

$$\leq \max_{x \in \Omega} \sum_{j \in X} |f(x_j) M_s^j(x)| \quad \text{(Triangle Inequality)}$$

$$\leq \max_{x \in \Omega} \sum_{j \in X} |f(x_j)| \cdot |M_s^j(x)|$$

$$\leq P_s^\infty \cdot \|f\|_{L_\infty(\Omega)} \quad \text{where} \quad P_s^\infty = \max_{x \in \Omega} \sum_{j=1}^n |M_s^j(x)|$$

Let $x^* \in \Omega$ be the data point at which $\sum_{j=1}^n |M_s^j(x)|$ is maximized. Therefore

$$P_s^\infty = \sum_{j=1}^n |M_s^j(x^*)| = \sum_{j=1}^n |\delta_x^{x^*} M_s^j| \quad \text{(Point Evaluational functional)}$$

$$\leq \sum_{j=1}^n ||\delta_x^{x^*}||_{\mathcal{H}_s} \cdot ||M_s^j||_{\mathcal{H}_s} = \sum_{j=1}^n ||M_s^j||_{\mathcal{H}_s} \quad \text{Since} \quad ||\delta_x^{x^*}||_{\mathcal{H}_s} = 1$$
Now, we know
\[< M^j_s, M^j_s >_{H_s} = e_j^T B^s^T R_s(x)|x_s \cdot R_s(x)| X_s, B^s e_j \]
\[= e_j^T B^s^T G x_s B^s e_j = e_j^T D_s e_j = D_s(j,j) \] (34)
Therefore
\[P_s^\infty \leq \sum_{j=1}^n ||M^j_s||_{H_s} = \sum_{j=1}^n \sqrt{D_s(j,j)} \] (36)

For lower bound on \(P_s^\infty\),
\[P_s^\infty = \max_{x \in \Omega} ||M_s(x)||_1 \geq \max_{x \in \Omega} ||M_s(x)||_2 \quad \text{(since } ||\cdot||_1 \geq ||\cdot||_2) \] (37)

For a tall thin matrix B, \(B^T B^\dagger = B^T\). Therefore, \(M_s(x)\) is the least squares solution of \(B^s^T M_s(x) = R_s(x)|x_s\), and,
\[||B^s^T||_2||M_s(x)||_2 \geq ||R_s(x)|x_s||_2 \] (38)
\[||M_s(x)||_2 \geq (\sigma_{s_{max}}(B^s^T))^{-1}||R_s(x)|x_s||_2 \]
where \(\sigma_{s_{max}}\) is the maximum singular value. Putting in equation (37).
\[P_s^\infty \geq \max_{x \in \Omega} (\sigma_{s_{max}}(B^s^T))^{-1}||R_s(x)|x_s||_2 \]
\[= (\sigma_{s_{max}}(B^s^T))^{-1} \max_{x \in \Omega} ||R_s(x)|x_s||_2 \]
\[\geq (\sigma_{s_{max}}(B^s^T))^{-1} \quad \text{(Because } \max_{x \in \Omega} ||R_s(x)|x_s||_2 > 1) \]
Combining equation (36) and above result, the bounds on \(P^\infty_s\) follow.

We will conclude this section by providing a bound on the approximation at any scale \(s\) at some point \(x \in \Omega\).

**Theorem 5.2.** For observed data \(f|X = (f(x_1), f(x_2), ... f(x_n))^T \in \mathbb{R}^n\) on \(X \in \Omega\), the absolute value of the approximation produced by the hierarchical algorithm \((A_s f)\) at scale \(s \leq S_a < S_c\) at any point \(x \in \Omega\) is bounded as
\[|(A_s f)(x)| \leq \sum_{j=1}^n \sqrt{D_s(j,j)} \cdot || f||_{H_s} \] (39)
Where \(D_s\) is defined as in Theorem (5.1). Also if the convergence happens at the critical scale \((S_c)\), then at convergence, bound (39) can be simplified as
\[|(A_s f)(x)| \leq n \cdot \sigma_{s_{max}}(G_{S_c}^{-1}) \cdot || f||_{H_s} \] (40)
Where \(\sigma_{s_{max}}\) is the maximum eigenvalue operator.

**Proof.** Here again we begin with the absolute value of approximation produced by the proposed algorithm at scale \(s\).
Figure 1. Univariate \{(a): Gramacy and Lee function - Test function 1 (TF1); (b): 1-D Schwefel function - TF2} and multivariate \{(c): Dropwave function - TF3; (d): 2-D Schwefel function - TF4} test functions considered for studying the performance of Algorithm 1.

\begin{align*}
|(A_s f)(x)| & 
\leq \left| \sum_{j=1}^{n} f(x_j) M_j^s(x) \right| \leq \sum_{j=1}^{n} |f(x_j)||M_j^s(x)| = \sum_{j=1}^{n} |\delta_{x_j}^r f||M_j^s(x)| \\
& \leq \sum_{j=1}^{n} ||\delta_{x_j}^r ||_{\mathcal{H}_s} ||f||_{\mathcal{H}_s} ||M_j^s||_{\mathcal{H}_s} = \sum_{j=1}^{n} ||f||_{\mathcal{H}_s} ||M_j^s||_{\mathcal{H}_s} \quad \text{since: } (||\delta_{x_j}^r ||_{\mathcal{H}_s} = 1) \\
& = \sum_{j=1}^{n} \sqrt{D_s(j,j)} ||f||_{\mathcal{H}_s}
\end{align*}

The last step was carried out using equation (35). Now if the convergence happens at \( s = S_c = S_a \), then \( D_{S_c} = G_{S_c}^{-1} \) and therefore \( D_{S_c}(j,j) = G_{S_c}^{-1}(j,j) \). Let \( g(j,j) = G_{S_c}^{-1}(j,j) \)

\begin{align*}
|(A_{S_c} f)(x)| & \leq \sum_{j=1}^{n} \sqrt{g(j,j)} ||f||_{\mathcal{H}_s} \\
& \leq \sum_{j=1}^{n} g(j,j) ||f||_{\mathcal{H}_s} \quad \text{Since } g(j,j) \geq 1 \text{ for } 1 \leq j \leq n \\
& \leq n \cdot \sigma_{\text{max}}(G_{S_c}^{-1}) ||f||_{\mathcal{H}_s}
\end{align*}
Remark 1. We note that the bounds here are conservative and depend on the data set size $n$. The assumed global overlap of the basis functions leads to the loose upper bound. However, as Figure 4 (in the next section) shows the basis functions have a rapid decay and attained bounds in practice are much smaller.

6. Results and analysis. This section analyzes the behavior of the proposed approach on a variety of datasets under different conditions. The first subsection here studies the performance on synthetic datasets. This is important, as here we know the truth, and hence quantification of performance (both in terms of reconstruction and out of sample prediction) becomes feasible. The following subsection analyzes the performance of Algorithm 1 on a real spatial dataset. Since in reality, estimating the true generalization error is not feasible (as the sampling distribution is not known), so following the standard practice, we empirically estimate this error by the performance on an unseen testing set.

6.1. Analysis on synthetic datasets. Here we have chosen a set of 4 test functions (Figure 1) from literature [47] providing our proposed algorithm, the sampled data to learn the underlying function. These test functions have been shown in Figure 1, and are mathematically expressed as:

- Test Function 1 (TF1): Gramacy and Lee Test function
  \[ f(x) = \frac{\sin(10\pi x)}{2x} + (x - 1)^4 \quad \text{where } x \in [0.5, 2.5] \]
Figure 3. Convergence measured as the decay of upper bound (equation (45)) to the inner product in the native Hilbert space between approximation $A_s f$ and the approximation error $E^s$ for considered univariate ((a): TF1 and (b): TF2) and multivariate ((c): TF3 and (d): TF4) test functions.

- Test Function 2 (TF2): 1-D Schwefel Function
  \[ f(x) = 418.9829 - x \cdot \sin(\sqrt{|x|}) \quad \text{where} \quad x \in [-500, 500] \]  

- Test Function 3 (TF3): Dropwave Function
  \[ f(x, y) = -\frac{1 + \cos(12\sqrt{x^2 + y^2})}{0.5(x^2 + y^2) + 2} \quad \text{where} \quad x, y \in [-2, 2] \]  

- Test Function 4 (TF4): 2-D Schwefel Function
  \[ f(x, y) = 837.9658 - x \cdot \sin(\sqrt{|x|}) - y \cdot \sin(\sqrt{|y|}) \quad \text{where} \quad x, y \in [-500, 500] \]

After sampling the data from these test functions, for all axis (X and Y for TF1 and TF2, and X, Y and Z for TF3 and TF4 respectively) the values are normalized between -1 to 1 by dividing the measurements with the corresponding absolute maximum value along each axis. It should be noted here that for most of the analysis presented here, we have sampled 200 equidistant points for TF1 and TF2. And for TF3 and TF4, points are sampled on a 50 × 50 grid. The motivation here is to sample data points from the test functions and reconstruct the functions from the sampled data using the generated sparse representation ($D_{sparse}$). These functions were specifically chosen as they have a lot of curvature changes and multiple local minima and maxima, which makes learning the function form difficult. As mentioned before, besides reconstruction we also test the generalization (prediction at new data locations out of training sample) ability of our approach. To give a sense of the quality of performance, we also compare the results of our analysis with the performance of Algorithm 4 from [5] on the same experimental setup.

We begin with the analysis of the convergence of the algorithm on the test functions.
Figure 4. Scalewise performance of Algorithm 1 on TF2. The plots in the left column ((a), (c), (e) and (g)) show the density of basis functions at each scale, selected while identifying the sparse representation. Rank here refers to the cardinality of basis function set $B_s$. The right column ((b), (d), (f) and (h)) shows the corresponding scalewise reconstruction of the underlying function. The small dots show the samples from true function while star markers represent the sparse representation $D_{s_{\text{sparse}}}$. The solid curve is the reconstruction from the sparse representation. The fraction in bracket (for example (11/200) for (b)) shows the number of data points chosen out of 200 as $D_{s_{\text{sparse}}}$.

6.1.1. Convergence behavior. Figure 2 shows the convergence behavior of the Algorithm 1 by studying the 2-norm error of the prediction with respect to the original observations. $TOL = 10^{-2}$ was used in Algorithm 1 for generating these results. Following the notations used earlier, $S_a$ here represents the convergence scale with $S_c$ being the critical scale. Figure 2 also illustrates the proportion of dataset used at each scale $s$ for generating the approximation $A_s f$. It is essentially the proportion of the dataset used as the sparse representation. Therefore, from figure 2 we can make inferences like, for test function 1, at scale 6 with 23% of datapoints, the proposed algorithm was able to generate an approximation $A_6 f$ which had a 2-norm error of less than $10^{-5}$. It should be noted here, that based on the structure, the error measure reduces in a unique manner for all 4 test functions which have different convergence scales.

Figure 3 shows the convergence bounds from Theorem 3.3. Here we have shown the results in 1-norm, which state that at any scale $s$, in the Hilbert space
Figure 5. Scalewise performance of Algorithm 1 on TF3. Left column ((a), (c), (e) and (g)) shows the distribution of the sparse representation selected at multiple scales. Here we have shown the projection of the sparse representation (star markers) on the X-Y plain for ease of presentation. The right column ((b), (d), (f) and (h)) shows the corresponding reconstruction for the dropwave test function from the respective $D_{sparse}^s$. Again the fraction in the header of plots (right column) shows the proportion of dataset chosen as $D_{sparse}^s$.

\[ \left| \left\langle (A_s f), E^s \right\rangle_{H_s} \right| \leq ||C_s||_{\infty} ||E^s||_{X} ||1 \]  \hspace{1cm} (45)

Figure 3 plots the quantity on right in the above equation with increasing scales. The sharp drop in the this bound for all 4 test functions justifies the capacity of the algorithm to produce good approximations. Precisely, at higher scale in the native Hilbert space, the error in prediction approaches orthogonality with respect to the approximation $A_s f$.

6.1.2. Scale dependent $D_{sparse}^s$ and corresponding reconstructions. Figure 4 and 5 show the behavior of the selected sparse representation and the approximation $A_s f$ produced with increasing scales. In this subsection we have only presented results for TF2 and TF3 for analyzing the performance of the algorithm in univariate
Figure 6. Confidence and Prediction Intervals for reconstruction from $D^*_\text{sparse}$ from scale 0 to scale 5 for TF2. Along with the approximation produced at these scales (solid curve), the plots also show the sparse representation selected (star marked points) with the 95% t-confidence interval (thinner dark shaded region) and 95% t-prediction interval (broader light shaded region).

and multivariate setting respectively. Starting with figure 4, the plots in the left column show that with increasing scale, the support of the basis functions becomes narrower. This corresponds to the increasing numerical rank of the kernel matrix $G_s$ with increasing $s$. The wider support of the basis functions in the initial scales also explains the corresponding over-smoothed approximations. This scenario is similar to behavior of approximation strategies with global bases (for example polynomial based approximation). In the right column, for every scale we have mentioned the number of points chosen as sparse representation (out of 200 points). It should be noted here that the star marked points represent the $D^*_\text{sparse}$ sampled from the smaller dotted data points. Here, our motivation is not just to show that with few points, the algorithm is able to learn the underlying function. But also, that the algorithm has an inbuilt capacity to choose a small set of representative points which can appropriately capture the function structure.

Figure 5 shows the corresponding result for the dropwave wave function. Since proper visualization of the basis functions for a surface is a little challenging, so here we have just shown the location of the points chosen in the sparse representation (in X-Y plane) and the respective bases would be centered at these locations. One additional thing to be noted here is the higher density of sampling near the edges of the domain. This directly corresponds with the fact that at the edges, for matching the curvature appropriately, it needs more points as there is no scope of learning beyond the edges. The corresponding reconstruction also shows how the specific features are learned over the scales. Again, since points were sampled on a $50 \times 50$ grid. So $D^*_\text{sparse}$ consists of data points sampled from 2500 design points.
6.1.3. **Confidence and prediction intervals.** The results for this section have been shown in figure 6. The analysis is shown for scale 0 to scale 5. The smaller dotted points show the original data and the star marked points represent the sampled ones for $D_{\text{sparse}}$. The thinner bands show the 95% confidence interval on the estimated approximation at each scale. It should be noted here that, we have not carried out a full Bayesian analysis here. Instead, we have just used the fitting variance (31) as a proxy for the variance and used it for scaling our interval. Specifically here we are using t-confidence bands which are suitable for smaller datasets (as compared to Gaussian bounds) and tend to be normal in the limit of larger datasets. The thicker band show the prediction interval. The main idea to be conveyed here is that if at a scale $s$, we have our sparse representation $D_{\text{sparse}}$, then with these prediction intervals, we can make estimations of where all the deleted points and data points to be sampled in future would lie (bounds on generalization in some sense). One other way to say the same thing is, that if we are at a particular scale $s$, and if we fix the location on $x \in X$, then we can be 95% confident that the mean of $y$ values observed at that particular $x$ will lie within the thinner confidence bounds. Similarly for a fixed $x$, the broader prediction intervals show the range in which any new observation to be made in the future will lie. The results here also confirms the fact that with increasing scales both the bounds become very thin showing confidence in the produced approximation.

6.1.4. **Importance metric for design points.** This section aims at further exploring the application of the proposed algorithm. The results of the current study are presented in figure 7. The idea is based upon the requirement that besides just getting the sparse representation at each scale, sometimes we also need to define an ordering of data points in $D_{\text{sparse}}$ in decreasing order of importance with respect to efficient reconstruction of $f$. This is important because if a measurement at a design point is of very high priority, then more resources could be engaged to measure that particular observation accurately. Also if we need to further compress the data, then in what sequence the datapoints can be removed. A test case is shown in figure 7(I). It contains the importance ranking computed based on the order in which the points were sampled while getting the sparse representation. It shows that the most important point (rank 1) at scale 0 is near the center of the function (and hence should rank last in the ordering for deletion). This confirms our belief that an observation near the center is crucial to capture the behavior of the function to be approximated. The second and third most important points (rank 2 and 3 respectively) are found to be at the very end of the curve which again is logical based on the fact that algorithm needs precise information to capture the function at the edges. These results also make sense because of the inherent symmetry of the 1-D schwefel function under consideration. It should be noted that the location of these important points depends on the nature of the function under consideration, so for any other functions, the location of important points might be very different as compared to the ones obtained in figure 7(I). Also, this was the result obtained for one random run of the approach and a different run might produce a different ordering.

Now, for getting a more detailed picture of importance metric, we ran 1000 simulations of Algorithm 1 and presented the distribution (for location) of the top 3 most important points. The results are presented in figure 7(II). Here the X-axis varies from 0 to 200 because we considered 200 sampled points from TF2 as our learning set. The analysis was run for scales 0, 1, 2, 3, 5, 10 for studying the
behavior of the distribution. It could be very well seen here that all the weight of the distribution for the most important point (I:1 in top row) is concentrated at the center of the domain for the initial scales (leftmost columns). For the same scales, the second (middle row) and third (bottom row) most important points have all their mass concentrated at the edges (again more obvious in leftmost columns). However, if we move towards the critical scale - right columns(where all points are included in the sparse representation), all the points have approximately uniform importance density distribution. This is also expected as when all points are sampled, no single point is more important than the other.

6.1.5. Comparison with Algorithm 4 in Bermanis et al. As mentioned earlier, [5] was one of the major motivators for the current work. In that paper, algorithm 4 for multiscale data sampling and function extension comes close in behavior to our hierarchical approach (however the approximation strategy is different). Briefly the idea there can be summarized as follows. Suppose the approximation at scale $s$ is represented as $H_s f$. Therefore starting with scale 0, $H_0 f$ is the approximation to
HIERARCHICAL LEARNING

Figure 8. Comparative behavior of the decay of 2-norm error for the Multiscale extension algorithm from [5] (represented as ‘Berm’) and Algorithm 1 (represented as ‘Shek’) for the four test function TF1, TF2, TF3 and TF4 in (a), (b), (c) and (d) respectively.

\[ f \text{ produced at scale } 0. \text{ Thus we can write} \]

\[ (H_0 f) |_X \approx f |_X \]

However, when we move further, the error orthogonal to the search space at the previous scales becomes the target function for the next scale. Therefore,

\[ (H_1 f) |_X \approx (f - (H_0 f)) |_X \]

\[ \vdots \]

\[ (H_s f) |_X \approx (f - \sum_{i=0}^{s-1} H_i f) |_X \]

With a user defined error tolerance (err), the authors define the convergence scale \((s^*)\) as the scale satisfying

\[ \| (f - \sum_{i=0}^{s^*} H_i f) |_X \| \leq \text{err} \]

Therefore, final approximation to \( f \) is of the form

\[ f \approx H_0 f + H_1 f + \ldots + H_{s^*} f \quad (46) \]

Hence algorithm 4 in [5] has bases from all scales for the final approximation. If we take a closer look at our algorithm (Algorithm 1), we see that although our approach also samples bases at each scale, for final approximation, we just use the bases at convergence scale for approximation. This behavior enables our algorithm to have a uniform length scale with basis functions centered at data points spread out ‘nicely’ throughout the approximation domain. This nature of basis functions (obtained as a result of the pivoted QR-decomposition), makes our basis matrix \((B^*)\) well conditioned. The mixing of scales in algorithm 4 in [5] makes conditioning an issue, which is problematic while computing the coordinates of projection. This further provides a well founded motivation for our work. Furthermore, mixing the basis functions (as in [5]) at multiple scales tie the approximation very closely to the training data and hence leads to relatively worse generalization as will be seen in the next subsection.

The behavior (in terms of approximation quality) of the algorithm 4 from [5] (referred to as ‘Berm’ henceforth) is compared with Algorithm 1 (correspondingly referred to as ‘Shek’) in figure 8. Here the decay of error shows similar behavior for both the algorithms. For TF1 (in (a)), Shek shows some faster convergence. However, for TF3 and TF4 (in (c) and (d)), Berm algorithm reduces the error to
|       | Training time (s) | Prediction time (s) |
|-------|-------------------|---------------------|
|       | | | |
|       | | | |

**Table 1.** Comparison of training and prediction time for Algorithm 1 (*Shek*) and Algorithm 4 from [5] (*Berm*). For training analysis, we have considered the case of different training data sizes (|X|) for all test functions. Correspondingly, for testing prediction latency, we again have sets |X\_pred| of different cardinality. For each case, the approach taking shorter time has been boldfaced.
below machine precision faster than Shek. Although quite impressive, this doesn’t make Berm any more useful because we are already at error levels of $10^{-13}$ at such higher scales.

With comparative approximation (reconstruction) capabilities, we now move to the analysis of training and prediction latency. Detailed analysis of these results has been shown in Table 1. Starting with training, both Shek and Berm obtain approximation bases at each scale using pivoted QR decomposition. Shek models the original target function while Berm uses them to model the error orthogonal to the search space till the previous scale. In this way, the training time for the two approaches is comparable. Table 1 shows for some cases Shek takes less time, with Berm performing efficiently in other cases. The reason for these differences can be attributed to the fact that Berm has to keep track of points sampled at each scale from $s = 0$ to $s = s^*$ along with computation of error (that serves as the target at the next scale). These factors increase the training latency for Berm in most cases. However, depending on the function structure, Berm can converge relatively quickly in a few cases leading to less training time.

Now, if we think of prediction at a new design point for Berm, then we will have to iteratively compute the corresponding approximation for each considered scale as in equation (46). This is where Shek outperforms Berm by only just requiring the linear combination of bases formulated with respect to the sparse representation at the convergence scale. This characteristic of Shek allows us to talk about sparse representation of the dataset which is not possible with the definition of Berm. Table 1 shows this behavior more clearly. Here we have measured the time which each of the algorithms take for just prediction, with all the learning assumed to be performed beforehand. Here $|X_{pred}|$ shows the number of design points at which the prediction is to be made. It should be noted here that unlike training time, for prediction Shek consistently performs better for all the considered cases.

[27] presented an approach similar to Berm from [5]. They however used Delaunay triangulations to thin the dataset at each scale effectively producing a nested set of datapoints leading finally to the original set $X$. At each scale, they project the error on the current approximation space (similar to Berm). Since their approach is clearly based on similar combination of multiple scales (with Delaunay triangulations instead of pivoted QR decomposition), we have just chosen to compare to Berm (as algorithm in [27] is expected to have similar issues as compared to our algorithm). Also, since pivoted QR decomposition samples the datapoints based on the approximation power of the functions centered at them (coupling the thinning and approximation problem), it is expected to perform better than the triangulation strategies that are just aimed at thinning the dataset with approximation carried out separately.

6.1.6. Generalization analysis. In the previous experiments, we have analyzed the capability of our hierarchical approach for reconstructing the analytical test functions we have considered for this study (evaluating performance on the same data). However, one other important aspect of the learning approaches which we must also study, relates to the ability to generalize (predict) well to unseen samples (not available to the algorithm during the learning phase). Additionally, here we also study the performance of our approach when the training dataset has a non-uniform sampling (the samples are not available on well defined grids). For analyzing these two aspects, in this section we have divided the data for TF1, TF2, TF3 and TF4 (200 samples for TF1 and TF2, 2500 samples for TF3 and TF4 respectively) into training
and testing sets using 50% as training and rest 50% as testing. The training samples are randomly selected from the approximation domain providing a training dataset with non-uniform sampling. Also, since we will be testing on the remaining 50% dataset, it will quantify the ability of the approach to predict on new data points. Here we denote training and testing datasets as $D_{\text{train}}$ and $D_{\text{test}}$ respectively.

Figure 9 shows the analysis of a random run of the entire experiment. The left column ((a), (c), (e) and (g)) shows the training and testing samples from the 4 test functions. The corresponding plot on the right shows the reconstruction obtained just using the training data (in the previous experiments we used the full dataset for reconstruction). The header on the plots (b), (d), (f) and (h) show the testing root mean square error. Because of strong ‘structure’ in data for (b) and (f), the testing error is specially very small. We have then compared these results to the performance of Berm (shown in figure 10) on the same training and testing data for

Figure 9. Performance on unseen samples for TF1, TF2, TF3 and TF4 by Algorithm 1 (Shek). Here plots (a),(c),(e) and (g) ((e) and (g) just show the projection on XY plane) show the distribution of the training and testing data (50/50 % split) with corresponding plots on the right hand side ((b),(d),(f) and (g)) showing the reconstructions obtained just using the training data. The header on ((b),(d),(f) and (h)) also show the corresponding testing error.
the 4 test functions. It should be noted that (approximately) Shek gives an order of magnitude better performance as compared to Berm for predicting on unseen samples. The reason for this can again be attributed to the way Berm tries to fit very closely to the data by projecting the prediction error on the approximation space at the next scale and then mixing the scale to solve the ‘data fitting’ problem. This is also specifically evident from the the performance on TF2. In figure 9 (c), we can see a big cluster of testing data points (with very few training samples) around 0.75 to 0.9. This leads to some non smooth reconstruction by Shek in that region (in figure 9 (d)). However Berm performs considerably worse in that region (figure 10 (b)) by producing undesired spurious fluctuations. These fluctuations explain the Berm’s bad behavior on unseen samples.

For showing the consistency of better performance by our approach (Shek) as compared to Berm, we did an additional experiment shown in figure 11. Here we repeated the experiment in figure 9 and 10, 100 times (100 different random splits into training the testing) and computed the RMSE of prediction on the testing dataset \((D_{\text{test}})\). Bar plots in figure 11 (a) and (b) show the mean and standard deviation of this RMSE for both Shek and Berm. The important thing to note here is the consistent lower mean RMSE by Shek for 100 different train/test sets as
compared to *Berm* on all 4 test functions. This shows superior generalization ability irrespective of the sampling distribution. Similarly, the lower standard deviation of RMSE for all 4 test functions in (b) again shows the relatively stable nature of our algorithm with respect to how the samples were generated.

### 6.2. Application on real data.

In this section we have analyzed the performance of Algorithm 1 (*Shek*) on real observed datasets (the true underlying function is not known). Specifically, we consider a particular type of spatial dataset known as Digital Elevation Model (DEM). It is a topographical map of a particular region. The data is arranged on grids with each grid node \((x,y)\) associated with a height measurement. Here \(X\) and \(Y\) are projection coordinates on a horizontal plane (usually transformed from latitudes and longitudes). The DEM dataset we consider is shown in figure 12 (a). Also, like the previous sections, here again we compare the performance of *Shek* with respect to *Berm* by sampling the DEM (figure 12 (a)) at even indexed nodes (both along \(X\) and \(Y\) dimension) and creating a lower resolution dataset (120m) shown in figure 12 (b). We use this as our training data for prediction at deleted nodes (testing dataset).

The first experiment is shown in figure 13. Here we show the reconstruction results for scale 0, 2 and 6 using the full DEM dataset (12 (a)) for training. These specific scales were chosen so as to provide an idea of how the surface is evolving towards the starting scale and towards the end. The important thing to note here is that at scale 6, with \(D_{\text{sparse}}\) only consisting of 763 points out of 4350 points, the algorithm was able to generate a reconstruction to the DEM where the prediction error in \(\infty-norm\) is just 6.74 (compared to the range of variation observed as \(\sim [250, 500]\) in the shadebar in figure 12(a)). The compression metric of \((763/4350 \sim 17.5\%)\) clearly shows the success of the algorithm in generating a sparse representation for the dataset. Here \(\infty-norm\) was chosen as an error measure as it upper bounds the error at any individual point and gives an intuitive understanding of the performance of the algorithm. Figure 13 also shows how specific topographic features become sharper with increase in scale.

The next analysis we show here again corresponds to comparing the generalization ability of *Shek* and *Berm* with the training DEM shown in figure 12(b) and testing on the grid points that were deleted to obtain the training data. Figure 14 shows the reconstruction of the original resolution DEM (60m) from the training DEM with 120m resolution. (a) and (b) here show the reconstructions obtained by...
Figure 13. Scale dependent reconstructions for the DEM dataset in figure 12 (a). Here (a), (b) and (c) show the reconstructed contour plot for scales 0, 2 and 6 respectively. The header on these plots also show the proportion of dataset selected as the sparse representation along with the $\infty$-norm of the reconstruction error.

Figure 14. Comparison of 60m DEM reconstruction by Shek (a) and Berm (b) using the 120m DEM training data shown in figure 12 (b). The header shows the RMSE for prediction of height at the testing locations (testing RMSE for Shek: 2.74 and testing RMSE for Berm: 10.7).

Shek and Berm. Similar to the case of test functions, here again, the performance on the testing data is an order of magnitude better for Shek as compared to Berm.

7. Conclusion. In this paper, we have introduced a hierarchical method for learning a sequence of sparse representations for a large dataset. The hierarchy comes from approximation spaces defined with respect to a scale parameter. Principally, the proposed approach has been shown to be useful for data reduction applications coupled with learning a model for representing the data. The paper provides analysis that explains and studies the theoretical properties of the proposed approach (with the code available online). We derive bounds for stability, convergence and behavior of error functionals. In the results section, we have shown the performance of the approach as data reduction mechanism on both synthetic and a real dataset (geo-spatial). The sparse model generated by the presented approach is also shown to efficiently reconstruct the data, while minimizing error in prediction. Besides reconstruction, we have also analyzed the ability of the proposed approach to generalize to unseen samples and compared the performance with algorithm 4 from [5]. Our approach is shown to consistently provide, approximately an order of magnitude better performance on the testing datasets (both real and synthetic) with considerably better stability behavior.
Though the results shown in this paper depict the efficiency of the approach on a variety of datasets and settings, there are several areas in which the presented algorithm can be improved. Firstly, the implementation of the algorithm can be made more efficient by either optimizing the operations in the algorithm or by handling chunks of data at a time. Secondly, generation of sparse representation for noisy datasets poses another set of unique challenges which will be addressed in a companion paper under preparation. For this case, properly capturing the uncertainty in the generated sparse approximations becomes very crucial. Finally the hyperparameters like $P$ in (Algorithm 1) can be explored and studied further for making the approach behave better. This can be done by even utilizing additional information which is not directly observed but is inherently known (like the physics of a system). Hence, in essence the work presented addresses the need for efficient learning methods for large datasets and opens up new interesting approaches.

REFERENCES

[1] N. I. Achieser, *Theory of Approximation*, Frederick Ungar Publishing Co., New York, 1956.
[2] W. K. Allard, G. Chen and M. Maggioni, Multi-scale geometric methods for data sets II: Geometric multi-resolution analysis, *Appl. Comput. Harmon. Anal.*, 32 (2012), 435–462.
[3] N. Aronszajn, Theory of reproducing kernels, *Trans. Amer. Math. Soc.*, 68 (1950), 337–404.
[4] F. Belloccio, N. Borghese, S. Ferrari and V. Piuri, Kernel regression in HRBF networks for surface reconstruction, 2008 IEEE International Workshop on Haptic Audio Visual Environments and Games, (2008), 160–165.
[5] A. Bermanis, A. Averbuch and R. R. Coifman, Multiscale data sampling and function extension, *Appl. Comput. Harmon. Anal.*, 34 (2013), 15–29.
[6] A. Bermanis, G. Wolf and A. Averbuch, Diffusion-based kernel methods on euclidean metric measure spaces, *Appl. Comput. Harmon. Anal.*, 41 (2016), 190–213.
[7] B. Bohn, J. Garcke and M. Griebel, A sparse grid based method for generative dimensionality reduction of high-dimensional data, *J. Comput. Phys.*, 309 (2016), 1–17.
[8] N. A. Borghese and S. Ferrari, Hierarchical rbf networks and local parameters estimate, *Neurocomputing*, 19 (1998), 259–283.
[9] C. Boutsidis, M. W. Mahoney and P. Drineas, An improved approximation algorithm for the column subset selection problem, *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, SIAM, (2009), 968–977.
[10] W. L. Briggs, V. E. Henson and S. F. McCormick, *A Multigrid Tutorial*, 2nd edition, SIAM, 2000.
[11] M. D. Buhmann, *Radial Basis Functions: Theory and Implementations*, Vol. 12, Cambridge university press, 2003.
[12] J. C. Carr, W. R. Fright and R. K. Beatson, Surface interpolation with radial basis functions for medical imaging, *IEEE Transactions on Medical Imaging*, 16 (1997), 96–107.
[13] G. Chen, A. V. Little and M. Maggioni, Multi-resolution geometric analysis for data in high dimensions, in *Excursions in Harmonic Analysis*, Springer, 1 (2013), 259–285.
[14] E. W. Cheney, *Introduction to Approximation Theory*, McGraw-Hill Book Co., New York-Toronto, Ont.-London, 1966.
[15] W. Cheney and W. Light, *A Course in Approximation Theory*, Vol. 101, American Mathematical Society, 2009.
[16] C. H. Chou, B. H. Kuo and F. Chang, The generalized condensed nearest neighbor rule as a data reduction method, 18th International Conference on Pattern Recognition, 2 (2006), 556–559.
[17] A. Çivril and M. Magdon-Ismail, On selecting a maximum volume sub-matrix of a matrix and related problems, *Theoret. Comput. Sci.*, 410 (2009), 4801–4811.
[18] R. R. Coifman, S. Lafon, A. B. Lee, M. Maggioni, B. Nadler, F. Warner and S. W. Zucker, Geometric diffusions as a tool for harmonic analysis and structure definition of data: Diffusion maps, *Proceedings of the National Academy of Sciences*, 102 (2005), 7426–7431.
[19] R. R. Coifman, S. Lafon, A. B. Lee, M. Maggioni, B. Nadler, F. Warner and S. W. Zucker, Geometric diffusions as a tool for harmonic analysis and structure definition of data: Multi-scale methods, *Proceedings of the National Academy of Sciences*, 102 (2005), 7432–7437.
[20] R. R. Coifman and M. Maggioni, Diffusion wavelets, *Appl. Comput. Harmon. Anal.*, 21 (2006), 53–94.
[21] I. Daubechies, *Ten Lectures on Wavelets*, CBMS-NSF Regional Conference Series in Applied Mathematics, 61. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1992.
[22] S. De Marchi and R. Schaback, Stability of kernel-based interpolation, *Adv. Comput. Math.*, 32 (2010), 155–161.
[23] A. Deshpande and S. Vempala, Adaptive sampling and fast low-rank matrix approximation, in Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, Springer, (2006), 292–303.
[24] G. E. Fasshauer and J. G. Zhang, Preconditioning of radial basis function interpolation systems via accelerated iterated approximate moving least squares approximation, in Progress on Meshless Methods, Springer, (2009), 57–75.
[25] S. Ferrari, F. Bellocchio, V. Piuri and N. A. Borghese, A hierarchical rbf online learning algorithm for real-time 3-d scanner, *IEEE Transactions on Neural Networks*, 21 (2009), 275–285.
[26] S. Ferrari, M. Maggioni and N. A. Borghese, Multiscale approximation with hierarchical radial basis functions networks, *IEEE Transactions on Neural Networks*, 15 (2004), 178–188.
[27] M. S Floater and A. Iske, Multistep scattered data interpolation using compactly supported radial basis functions, *J. Comput. Appl. Math.*, 73 (1996), 65–78.
[28] T. E. Fricker, J. E. Oakley and N. M. Urban, Multivariate Gaussian process emulators with nonseparable covariance structures, *Technometrics*, 55 (2013), 47–56.
[29] M. Galun, R. Basri and I. Yavneh, Review of methods inspired by algebraic-multigrid for data and image analysis applications, *Numer. Math. Theory Methods Appl.*, 8 (2015), 283–312.
[30] M. Gavish, B. Nadler and R. R. Coifman, Multiscale wavelets on trees, graphs and high dimensional data: Theory and applications to semi supervised learning, *ICML*, (2010), 367–374.
[31] M. Griebel and A. Hullmann, A sparse grid based generative topographic mapping for the dimensionality reduction of high-dimensional data, *Modeling, Simulation and Optimization of Complex Processes-HPSC*, Springer, (2012), 51–62.
[32] M. Gu and J. O. Berger, Parallel partial Gaussian process emulation for computer models with massive output, *Ann. Appl. Stat.*, 10 (2016), 1317–1347.
[33] A. Iske, Scattered data approximation by positive definite kernel functions, *Rend. Semin. Mat. Univ. Politec. Torino*, 69 (2011), 217–246.
[34] P. Koumoutsakos, Multiscale flow simulations using particles, *Annu. Rev. Fluid Mech.*, 37 (2005), 457–487.
[35] E. Kreyszig, *Introductory Functional Analysis with Applications*, Vol. 1, John Wiley & Sons, New York-London-Sydney, 1978.
[36] D. Kushnir, M. Galun and A. Brandt, Efficient multilevel eigensolvers with applications to data analysis tasks, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32 (2009), 1377–1391.
[37] T. Lane and C. E. Brodley, Temporal sequence learning and data reduction for anomaly detection, *CCS ’98: Proceedings of the 5th ACM Conference on Computer and communications Security*, (1998), 150–158.
[38] M. Maggioni, J. C. Bremer Jr, R. R. Coifman and A. D. Szlam, Biorhorthogonal diffusion wavelets for multiscale representation on manifolds and graphs, in *Wavelets XI*, International Society for Optics and Photonics, 5914, (2005), 59141M.
[39] S. Mallat, A theory for multiresolution signal decomposition: The wavelet representation, *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 11 (1989), 674–693.
[40] S. Paul, M. Magdon-Ismail and P. Drineas, Column selection via adaptive sampling, in *Advances in Neural Information Processing Systems*, (2015), 406–414.
[41] M. J. D. Powell, *Approximation Theory and Methods*, Cambridge university press, 1981.
[42] A. Quarteroni and A. Veneziani, Analysis of a geometrical multiscale model based on the coupling of ODEs and PDEs for blood flow simulations, *Multiscale Model. Simul.*, 1 (2003), 173–195.
[43] C. E. Rasmussen and C. K. I Williams, *Gaussian Process for Machine Learning* Adaptive Computation and Machine Learning. MIT Press, Cambridge, MA, 2006.
[44] E. Sadrifaridpour, S. Jeereddy, K. Kennedy, A. Luckow, T. Razzaghi and I. Safro, Algebraic multigrid support vector machines, preprint, (2016), arXiv:1611.05487.
[45] L. Shih, J. D. Rennie, Y. H. Chang and D. R. Karger, Text bundling: Statistics based data-reduction, in *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, (2003), 696–703.

[46] K. Stüben, A review of algebraic multigrid, in *Numerical Analysis: Historical Developments in the 20th Century*, Elsevier, (2001), 331–359.

[47] S. Surjanovic and D. Bingham, *Virtual Library of Simulation Experiments: Test Functions and Datasets*, Retrieved January 23, 2019, from http://www.sfu.ca/~ssurjano.

[48] A. D. Szlam, M. Maggioni, R. R. Coifman and J. C. Bremer Jr, Diffusion-driven multiscale analysis on manifolds and graphs: top-down and bottom-up constructions, in *Wavelets XI*, International Society for Optics and Photonics, 5914 (2005), 59141D.

[49] J. T. Vogelstein, Y. Park, T. Ohyama, R. A. Kerr, J. W. Truman, C. E. Priebe and M. Zlatic, Discovery of brainwide neural-behavioral maps via multiscale unsupervised structure learning, *Science*, 344 (2014), 386–392.

[50] H. Wendland, *Scattered Data Approximation*, Cambridge Monographs on Applied and Computational Mathematics, Vol. 17, Cambridge university press, 2005.

[51] Q. Wu, T. Xia, C. Chen, H.-Y. S. Lin, H. Wang and Y. Yu, Hierarchical tensor approximation of multi-dimensional visual data, *IEEE Transactions on Visualization and Computer Graphics*, 14 (2007), 186–199.

E-mail address: prashant.shekhar@tufts.edu
E-mail address: abani.patra@tufts.edu