Check for updates

Deep Nets for Local Manifold Learning

Charles K. Chui[1] and Hrushikesh N. Mhaskar[2]*

[1] *Department of Mathematics, Hong Kong Baptist University, Hong Kong, Hong Kong, [2] Institute of Mathematical Sciences, Claremont Graduate University, Claremont, CA, United States*

The problem of extending a function f defined on a training data $\mathcal{C}$ on an unknown manifold $\mathbb{X}$ to the entire manifold and a tubular neighborhood of this manifold is considered in this paper. For $\mathbb{X}$ embedded in a high dimensional ambient Euclidean space $\mathbb{R}^D$, a deep learning algorithm is developed for finding a local coordinate system for the manifold **without eigen-decomposition**, which reduces the problem to the classical problem of function approximation on a low dimensional cube. Deep nets (or multilayered neural networks) are proposed to accomplish this approximation scheme by using the training data. Our methods do not involve such optimization techniques as back-propagation, while assuring optimal (a priori) error bounds on the output in terms of the number of derivatives of the target function. In addition, these methods are universal, in that they do not require a prior knowledge of the smoothness of the target function, but adjust the accuracy of approximation locally and automatically, depending only upon the local smoothness of the target function. Our ideas are easily extended to solve both the pre-image problem and the out-of-sample extension problem, with a priori bounds on the growth of the function thus extended.

Keywords: deep learning, function approximation, manifold learning, neural networks, local approximation

1. INTRODUCTION

Machine learning is an active sub-field of Computer Science on algorithmic development for learning and making predictions based on some given data, with a long list of applications that range from computational finance and advertisement, to information retrieval, to computer vision, to speech and handwriting recognition, and to structural healthcare and medical diagnosis. In terms of function approximation, the data for learning and prediction can be formulated as $\{(\mathbf{x}, f_{\mathbf{x}})\}$, obtained with an unknown probability distribution. Examples include: the Boston housing problem (of predicting the median price $f_{\mathbf{x}}$ of a home based on some vector $\mathbf{x}$ of 13 other attributes [1]) and the floor market problem [2, 3] (that deals with the indices of the wheat floor pricing in three major markets in the United States). For such problems, the objective is to predict the index $f_{\mathbf{x}}$ in the next month, say, based on a vector $\mathbf{x}$ of their values over the past few months. Other similar problems include the prediction of blood glucose level $f_{\mathbf{x}}$ of a patient based on a vector $\mathbf{x}$ of the previous few observed levels [4, 5], and the prediction of box office receipts ($f_{\mathbf{x}}$) on the date of release of a movie in preparation, based on a vector $\mathbf{x}$ of the survey results about the movie [6]. It is pointed out in Mhaskar [7], Maggioni and Mhaskar [8] and Ehler et al. [9] that all the pattern classification problems can also be viewed fruitfully as problems of function approximation. While

it is an ongoing research to allow non-numeric input $\mathbf{x}$ (e.g., [10]), we restrict our attention in this paper to the consideration of $\mathbf{x} \in \mathbb{R}^D$, for some integer $D \geq 1$.

In the following discussion, the first component $\mathbf{x}$ is considered as input, while the second component $f_\mathbf{x}$ is considered the output of the underlying process. The central problem is to estimate the conditional expectation of $f_\mathbf{x}$ given $\mathbf{x}$. Various statistical techniques and theoretical advances in this direction are well-known (see, for example [11]). In the context of neural and radial-basis-function networks, an explicit formulation of the input/output machines was pointed out in Girosi and Poggio [12] and Girosi et al. [13]. More recently, the nature of deep learning as an input/output process is formulated in the same way, as explained in LeCun et al. [14] and Rosasco et al. [15]. To complement the statistical perspective and understand the theoretical capabilities of these processes, it is customary to think of the expected value of $f_\mathbf{x}$, given $\mathbf{x}$, as a function f of $\mathbf{x}$. The question of empirical estimation in this context is to carry out the approximation of f given samples $\{(\mathbf{x}, f(\mathbf{x}))\}_{\mathbf{x} \in \mathcal{C}}$, where $\mathcal{C}$ is a finite *training data* set. In practice, because of the random nature of the data, it may be possible that there are several pairs of the form $(\mathbf{x}, f_\mathbf{x})$ in the data for the same values of $\mathbf{x}$. In this case, a statistical scheme, such as some kind of averaging of $f_\mathbf{x}$ being the simplest one, can be used to obtain a desired value $f(\mathbf{x})$ for the sample of f at $\mathbf{x}$, $\mathbf{x} \in \mathcal{C}$. From this perspective, the problem of extending f from the traning data set $\mathcal{C}$ to $\mathbf{x}$ not in $\mathcal{C}$ in machine learning is called the *generalization problem*.

We will illustrate this general line of ideas by using neural networks as an example. To motivate this idea, let us first recall a theorem originating with Kolmogorov and Arnold [16, Chapter 17, Theorem 1.1]. According to this theorem, there exist *universal* Lipschitz continuous functions $\phi_1, \cdots, \phi_{2D+1}$ and *universal* scalars $\lambda_1, \cdots, \lambda_D \in (0, 1)$, for which every continuous function $f : [0, 1]^D \to \mathbb{R}$ can be written as

$$f(\mathbf{x}) = \sum_{j=1}^{2D+1} g\left(\sum_{k=1}^{D} \lambda_k \phi_j(x_k)\right), \qquad \mathbf{x} = (x_1, \cdots, x_D) \in [0, 1]^D,$$

(1.1)

where g is a continuous function that depends on f. In other words, for a given f, only one function g has to be determin to give the representation formula (1.1) of f.

A neural network, used as an input/output machine, consists of an input layer, one or more hidden layers, and an output layer. Each hidden layer consists of a number of neurons arranged according to the network architecture. Each of these neurons has a local memory and performs a simple non-linear computation upon its input. The input layer fans out the input $\mathbf{x} \in \mathbb{R}^D$ to the neurons at the first hidden layer. The output layer typically takes a linear combination of the outputs of the neurons at the last hidden layer. The right hand side of (1.1) is a neural network with two hidden layers. The first contains D neurons, where the j-th neuron computes the sum $\sum_{k=1}^{D} \lambda_k \phi_j(x_k)$. The next hidden layer contains $2D + 1$ neurons each evaluating the function g on the output of the j-th neuron in the first hidden layer. The output layers takes the sum of the results as indicated in (1.1).

From a practical point of view, such a network is clearly hard to construct, since only the existence of the functions ϕ_j and g is known, without a numerical procedure for computing these. In the early mathematical development of neural networks during the late 1980s and early 1990s, instead of finding these functions for the representation of a given continuous function f in (1.1), the interest was to study the existence and characterization of *universal* functions $\sigma : \mathbb{R} \to \mathbb{R}$, called *activation functions* of the neural networks, such that each neuron evaluates the activation function upon an affine transform of its input, and the network is capable of approximating any desired real-valued continuous target function $f : K \to \mathbb{R}$ arbitrarily closely on K, where $K \subset \mathbb{R}^D$ is any compact set.

For example, a neural network with one hidden layer can be expressed as a function

$$\mathcal{N}(\mathbf{x}) = \mathcal{N}_n(\{\mathbf{w}_k\}, \{a_k\}, \{b_k\}; \mathbf{x}) = \sum_{k=1}^{n} a_k \sigma(\mathbf{w}_k \cdot \mathbf{x} + b_k), \quad \mathbf{x} \in \mathbb{R}^D.$$

(1.2)

Here, the hidden layer consists of n neurons, each of which has a local memory. The local memory of the k-th neuron contains the *weights* $\mathbf{w}_k \in \mathbb{R}^D$, and the *threshold* $b_k \in \mathbb{R}$. Upon receiving the input $\mathbf{x} \in \mathbb{R}^D$ from the input later, the k-th neuron evaluates $\sigma(\mathbf{w}_k \cdot \mathbf{x} + b_k)$ as its output, where σ is a non-linear activation function. The output layer is just one circuit where the coefficients $\{a_k\}$ are stored in a local memory, and that evaluates the linear combination as indicated in (1.2). Training of this network in order to learn a function f on a compact subset $K \subset \mathbb{R}^D$ to an accuracy of $\epsilon > 0$ involves finding the parameters $\{a_k\}, \{\mathbf{w}_k\}, \{b_k\}$ so that

$$\max_{\mathbf{x} \in K} |f(\mathbf{x}) - \mathcal{N}(\mathbf{x})| < \epsilon.$$

(1.3)

The most popular technique for doing this is the so called back-propagation, which seeks to find these quantities by minimizing an error functional usually with some regularization parameters. We remark that the number n of *neurons* in the approximant (1.2) must increase, if the tolerance $\epsilon > 0$ in the approximation of the target function f is required to be smaller.

From a theoretical perspective, the main attraction of neural networks with one hidden layer is their *universal approximation property* as formulated in (1.3), which overshadows the properties of their predecessors, namely: the perceptrons [17]. In particular, the question of finding sufficient conditions on the actvation function σ that ensure the universal approximation property was investigated in great detail by many authors, with emphasis on the most popular *sigmoidal function*, defined by the property $\sigma(t) \to 1$ for $t \to \infty$ and $\sigma(t) \to 0$ for $t \to -\infty$. For example, Funahashi [18] applied some discretization of an integral formula from Irie and Miyake [19] to prove the universal approximation property for some sigmoidal function σ. A similar theorem was proved by Hornik et al. [20] by using the Stone-Weierstrass theorem, and another by Cybenko [21] by applying the Hahn-Banach and Riesz Representation theorems. A constructive proof via approximation by ridge functions was given in our paper [22], with algorithm for implementation presented in our follow-up work [23]. A complete characterization of which activation

functions are allowed to achieve the universal approximation property was given later in Mhaskar and Micchelli [24] and Leshno et al. [25].

However, for neural networks with one hidden layer, one of the severe limitations to applying training algorithms based on optimization, such as back-propagation or those proposed in the book [11] of Vapnik, is that it is neccessary to know the number of neurons in $\mathcal{N}$ in advance. Therefore, one major problem in the 1990s, known as the *complexity problem*, was to estimate the number of neurons required to approximate a function to a desired accuracy. In practice, this gives rise to a trade-off: to achieve a good approximation, one needs to have a large number of neurons, which makes the implementation of the training algorithm harder.

In this regard, nearly a century of research in approximation theory suggests that the higher the order of smoothness of the target function, the smaller the number of neurons should be, needed to achieve the desired accuracy. There are many different definitions of smoothness that give rise to different estimates. For example, under the condition that the Fourier transform of the target function f satisfies $\int_{\mathbb{R}^D} |\omega \hat{f}(\omega)| d\omega < \infty$, Barron [26] proved the existence of a neural network with $\mathcal{O}(\epsilon^{-2})$ neurons that gives an $L^2([0,1]^D)$ error of $\mathcal{O}(\epsilon)$. While it is interesting to note that this number of neurons is essentially independent of the dimension D, the constants involved in the $\mathcal{O}$ term as well as the number of derivatives needed to ensure the condition on the target function may increase with D. Several authors have subsequently improved upon such results under various conditions on the activation function as well as the target function so as to ensure that the constants depend polynomially on D (e.g., [27–29] and references therein).

The most commonly understood definition of smoothness is just the number of derivatives of the target function. It is well-known from the theory of n-widths that if $r \geq 1$ is an integer, and the only a priori information assumed on the unknown target function is that it is r-times continuously differentiable function, then a stable and uniform approximation to within ϵ by neural networks must have at least a constant multiple of $\epsilon^{-D/r}$ neurons. In Mhaskar [30], we gave an explicit construction for a neural network that achieves the accuracy of ϵ using $\mathcal{O}(\epsilon^{-D/r})$ neurons arranged in a single hidden layer. It follows that this suffers from a curse of dimensionality, in that the number of neurons increases exponentially with the input dimension D. Clearly, if the smoothness r of the function increases linearly with D, as it has to in order to satisfy the condition in Barron [26], then this bound is also "dimension independent."

While this is definitely unavoidable for neural networks with one hidden layer, the most natural way out is to achieve local approximation; i.e., given an input $\mathbf{x}$, construct a network with a uniformly bounded number of neurons that approximates the target function with the optimal rate of approximation near the point $\mathbf{x}$, preferably using the values of the function also in a neighborhood of $\mathbf{x}$. Unfortunately, this can never be achieved as we proved in Chui et al. [31]. Furthermore, we have proved in Chui et al. [32] that even if we allow each neuron to evaluate its own activation function, this local approximation fails. Therefore

the only way out is to use a neural network with more than one hidden layer, called *deep net* (for deep neural network). Indeed, local approximation can be achieved by a deep net as proved in our papers [7, 33]. In this regard, it is of interest to point out that an adaptive version of Mhaskar [7, 33] was derived in Mhaskar and Khachikyan [34] for prediction of time series, yielding as much as 150% improvement upon the state-of-the-art at that time, in the study of the floor market problem.

Of course, the curse of dimensionality is inherent to the problem itself, whether with one or more hidden layers. Thus, while it is possible to construct a deep net to approximate a function at each point arbitrarily closely by using a uniformly bounded number of neurons, the uniform approximation on an entire compact set, such as a cube, would still require an approximation at a number of points in the cube, and this number increasing exponentially with the input dimension. Equivalently, the effective number of neurons for approximation on the entire cube is still exponentially increasing with the input dimension.

In addition to the high dimensionality, another difficulty in solving the function approximation problem is that the data may be not just high dimensional but unstructured and sparse. A relatively recent idea which has been found very useful in applications, in fact, too many to list exhaustively, is to consider the points $\mathbf{x}$ as being sampled from an unknown, low dimensional sub-manifold $\mathbb{X}$ of the ambient high dimensional space $\mathbb{R}^D$. The understanding of the geometry of $\mathbb{X}$ is the subject of the bulk of modern research in the area of diffusion geometry. An introduction to this subject can be found in the special issue [35] of Applied and Computational Harmonic Analysis. The basic idea is to construct the so-called diffusion matrix from the data, and use its eigen-decomposition for finding local coordinate charts and other useful aspects of the manifold. The convergence of the eigen-decomposition of the matrices to that of the Laplace-Beltrami and other differential operators on the manifold is discussed, for example, in Belkin and Niyogi [36], Lafon [37], and Singer [38]. It is shown in Jones et al. [39, 40] that some of the eigenfunctions on the manifolds yield a local coordinate chart on the manifold. In the context of deep learning, this idea is explored as a function approximation problem in Mishne et al. [41], where a deep net is developed in order to learn the coordinate system given by the eigenfunctions.

On the other hand, while much of the research in this direction is focused on understanding the data geometry, the theoretical foundations for the problems of function approximation and harmonic analysis on such data-defined manifold are developed extensively in Maggioni and Mhaskar [8], Filbir and Mhaskar [42, 43], Mhaskar [44, 45] and Chui and Mhaskar [46]. The theory is developed more recently for kernel construction on directed graphs and analysis of functions on changing data in our paper [47]. However, a drawback of the approach based on data-defined manifolds, known as the out-of-sample extension problem, is that since the diffusion matrix is constructed entirely using the available data, the whole process must be done again if new data become available. A popular idea is then to extend the eigenfunctions to the ambient space by using the so called Nyström extension [48].

The objective of this present paper is to describe a deep learning approach to the problem of function approximation, using three groups of networks in the deep net. The lowest layer accomplishes dimensionality reduction by learning the local coordinate charts on the unknown manifold **without using any eigen-decomposition**. Having found the local coordinate system, the problem is reduced to the classical problem of approximating a function on a cube in a relatively low dimensional Euclidean space. For the next two layers, we may now apply the powerful techniques from approximation theory to approximate the target function f, given the samples on the training data set $\mathcal{C}$. We describe two approaches to construct the basis functions using multi-layered neural networks, and to construct other networks to use these basis functions in the next layer to accomplish the desired function approximation.

We summarize some of the highlights of our paper.

- We give a very simple learning method for learning the local coordinate chart near each point. The subsequent approximation process is then entirely local to each coordinate patch.
- Our method allows us to solve the pre-image problem easily; i.e., to generate a point on the manifold corresponding to a given local coordinate description.
- The learning method itself **does not involve** any optimization based technique, except probably for reducing the noise in the values of the function itself.
- We provide optimal error bounds on approximation based on the smoothness of the function, while the method itself *does not require an a priori knowledge of such smoothness.*
- Our methods can solve easily the out-of-sample extension problem. Unlike the Nyström extension process, our method does not require any elaborate construction of kernels defined on the ambient space and commuting with certain differential operators on the unknown manifold.
- Our method is designed to control the growth of the out-of-sample extension in a tubular neighborhood of $\mathbb{X}$, and is local to each coordinate patch.

This paper is organized as follows. In section 2, we describe the main ideas in our approach. The local coordinate system is described in detail in section 2.2. Having thus found a local coordinate chart around the input, the problem of function approximation reduces to the classical one. In section 2.3, we demonstrate how the popular basis functions used in this theory can be implemented using neural networks with one or more hidden layers. The function approximation methods which work with unstructured data without using optimization are described in section 2.6. In section 3, we explain how our method can be used to solve both the pre-image problem and the out-of-sample extension problem.

2. MAIN IDEAS AND RESULTS

The purpose of this paper is to develop a deep learning algorithm to learn a function $f : \mathbb{X} \rightarrow \mathbb{R}$, where $\mathbb{X}$ is a d dimensional compact Riemannian sub-manifold (without boundary) of a

Euclidean space $\mathbb{R}^D$, with $d \ll D$, given *training data* of the form $\{(\mathbf{x}_j, f(\mathbf{x}_j))\}_{j=1}^M$, $\mathbf{x}_j \in \mathbb{X}$. It is important to note that $\mathbb{X}$ itself is not known in advance; the points $\mathbf{x}_j$ are known only as D-dimensional vectors, presumed to lie on $\mathbb{X}$. In section 2.1, we explain our main idea briefly. In section 2.2, we derive a simple construction of the local coordinate chart for $\mathbb{X}$. In section 2.3, we describe the construction of a neural network with one or more hidden layers to implement two of the basis functions used commonly in function approximation. While the well known classical approximation algorithms require a specific placement of the training data, one has no control on the location of the data in the current problem. In section 2.6, we give algorithms suitable for the purpose of solving this problem.

2.1. Outline of the Main Idea
Our approach is the following.

1. $\mathbb{X}$ is a finite union of local coordinate neighborhoods, and $\mathbf{x}$ belongs to one of them, say $\mathbb{U}$. We find a local coordinate system for this neighborhood in terms of Euclidean distances on $\mathbb{R}^D$, say $\Phi : \mathbb{U} \rightarrow [-1, 1]^d$, where d is the dimension of the manifold. Let $\mathbf{y} = \Phi(\mathbf{x})$, and with a relabeling for notational convenience, $\{\mathbf{x}_j\}_{j=1}^K$ be the points in $\mathbb{U}$, $\mathbf{y}_j = \Phi(\mathbf{x}_j)$. This way, we have reduced the problem to approximating $g = f \circ \Phi : [-1, 1]^d \rightarrow \mathbb{R}$ at $\mathbf{y}$, given the values $\{(\mathbf{y}_j, g(\mathbf{y}_j))\}_{j=1}^K$, where $g(\mathbf{y}_j) = f(\mathbf{x}_j)$. We note that $\{\mathbf{y}_j\}$ is a subset of the unit cube of low dimensional Euclidean space, representing a local coordinate patch on $\mathbb{X}$. Thus, the problem of approximation of f on this patch is reduced that of approximation of g, a well studied classical approximation problem.
2. We will summarize the solution to this problem using neural networks with one or more hidden layers, e.g., an implementation of multivariate tensor product spline approximation using multi-layerd neural network.

Thus, our deep learning networks will have three main layers.

1. The bottom layer receives the input $\mathbf{x}$, figures out which of the points $\mathbf{x}_j$ are in the coordinate neighborhood of $\mathbf{x}$, and computes the local coordinates $\mathbf{y}$, $\mathbf{y}_j$.
2. The next several layers compute the local basis functions necessary for the approximation, for example, the B-splines and their translates using the multi-layered neural network as in Mhaskar [7].
3. The last layer receives the data $\{(\mathbf{y}_j, g(\mathbf{y}_j))\}_{j=1}^K$, and computes the approximation described in Step 2.1 above.

2.2. Local Coordinate Learning
We assume that $1 \leq d \leq D$ are integers, $\mathbb{X}$ is a d dimensional smooth, compact, connected, Riemannian sub-manifold (without boundary) of a Euclidean space $\mathbb{R}^D$, with geodesic distance ρ.

Before we discuss our own construction of a local coordinate chart on $\mathbb{X}$, we wish to motivate the work by describing a result from Jones et al. [40]. Let $\{\lambda_k^2\}_{k=0}^\infty$ be the sequence of eigenvalues of the (negative of the) Laplace-Beltrami operator on $\mathbb{X}$, and for each $k \geq 0$, ϕ_k be the eigenfunction corresponding to the

eigenvalue λ_k^2. We define a formal "heat kernel" by

$$K_t(\mathbf{x}, \mathbf{y}) = \sum_{k=0}^{\infty} \exp(-\lambda_k^2 t) \phi_k(\mathbf{x}) \phi_k(\mathbf{y}). \qquad (2.1)$$

The following result is a paraphrasing of the heat triangulation theorem proved in [40, Theorem 2.2.7] under weaker assumptions on $\mathbb{X}$.

Theorem 2.1. (cf. [40, Theorem 2.2.7]) *Let* $\mathbf{x}_0^* \in \mathbb{X}$. *There exist constants* $R > 0$, $c_1, \cdots, c_6 > 0$ *depending on* $\mathbf{x}_0^*$ *with the following property. Let* $\mathbf{p}_1, \cdots, \mathbf{p}_d$ *be* d *linearly independent vectors in* $\mathbb{R}^d$, *and* $\mathbf{x}_j^* \in \mathbb{X}$ *be chosen so that* $\mathbf{x}_j^* - \mathbf{x}_0^*$ *is in the direction of* $\mathbf{p}_j$, $j = 1, \cdots, d$, *and*

$$c_1 R \le \rho(\mathbf{x}_j^*, \mathbf{x}_0^*) \le c_2 R, \qquad j = 1, \cdots, d,$$

and $t = c_3 R^2$. *Let* $B \subset \mathbb{X}$ *be the geodesic ball of radius* $c_4 R$, *centered at* $\mathbf{x}_0^*$, *and*

$$\Phi_{jms}(\mathbf{x}) = R^d(K_t(\mathbf{x}, \mathbf{x}_1^*), \cdots, K_t(\mathbf{x}, \mathbf{x}_d^*)), \qquad \mathbf{x} \in B. \qquad (2.2)$$

Then

$$\frac{c_5}{R} \rho(\mathbf{x}_1, \mathbf{x}_2) \le \|\Phi_{jms}(\mathbf{x}_1) - \Phi_{jms}(\mathbf{x}_2)\|_d \le \frac{c_6}{R} \rho(\mathbf{x}_1, \mathbf{x}_2), \quad \mathbf{x}_1, \mathbf{x}_2 \in B. \qquad (2.3)$$

Since the paper [40] deals with a very general manifold, the mapping Φ_{jms} is not claimed to be a diffeomorphism, although it is obviously one-one on B.

We note that even in the simple case of a Euclidean sphere, an explicit expression for the heat kernel is not known. In practice, the heat kernel has to be approximated using appropriate Gaussian networks [37]. In this section, we aim to obtain a local coordinate chart that is computed directly in terms of Euclidean distances on $\mathbb{R}^D$, and depends upon $d + 2$ trainable parameters. The construction of this chart constitutes the first hidden layer of our deep learning process. As explained in the introduction, once this chart is in place, the question of function extension on the manifold reduces locally to the well studied problem of function extension on a d dimensional unit cube.

To describe our constructions, we first develop some notation. In this section, it is convenient to use the notation $\mathbf{x} = (x^1, \cdots, x^D) \in \mathbb{R}^D$ rather than $\mathbf{x} = (x_1, \cdots, x_D)$, which we will use in the rest of the sections. If $1 \le d \le D$ is an integer, and $\mathbf{x} \in \mathbb{R}^d$, $\|\mathbf{x}\|_d$ denotes the Euclidean norm of $\mathbf{x}$. If $\mathbf{x} \in \mathbb{R}^D$, we will write $\pi_c(\mathbf{x}) = (x^1, \cdots, x^d)$, $\|\mathbf{x}\|_d = \|\pi_c(\mathbf{x})\|_d$. If $\mathbf{x} \in \mathbb{R}^d$, $r > 0$,

$$B_d(\mathbf{x}, r) = \{\mathbf{y} \in \mathbb{R}^d : \|\mathbf{x} - \mathbf{y}\|_d \le r\}.$$

There exists $\delta^* > 0$ with the following properties. The manifold is covered by finitely many geodesic balls such that for the center $\mathbf{x}_0^* \in \mathbb{X}$ of any of these balls, there exists a diffeomorphism, namely, the exponential coordinate map $u = (u^1, \cdots, u^D)$ from $B_d(0, \delta^*)$ to the geodesic ball around $\mathbf{x}_0^* = u(0)$ [49, p. 65]. If J is

the Jacobian matrix for u, given by $J_{i,j}(\mathbf{y}) = D_i u^j(\mathbf{y})$, $\mathbf{y} \in B_d(0, \delta^*)$, then

$$J(0) = [I_d | 0_{d, D-d}]. \qquad (2.4)$$

Further, there exists $\kappa > 0$ (independent of $\mathbf{x}^*$) such that

$$\|J(\mathbf{q}) - J(0)\| \le \kappa \|\mathbf{q}\|_d, \qquad \mathbf{q} \in B_d(0, \delta^*), \qquad (2.5)$$

where the matrix norm is the induced norm. Let $\eta^* := \min(\delta^*, 1/(2\kappa))$. Then (2.5) implies that

$$1/2 \le 1 - \kappa \|\mathbf{q}\|_d \le \|J(\mathbf{q})\| \le 1 + \kappa \|\mathbf{q}\|_d \le 2, \qquad \mathbf{q} \in B_d(0, \eta^*). \qquad (2.6)$$

In turn, this leads to

$$(1/2)\rho(u(\mathbf{p}), u(\mathbf{q})) \le \|\mathbf{p} - \mathbf{q}\|_d \le 2\rho(u(\mathbf{p}), u(\mathbf{q})), \quad \mathbf{p}, \mathbf{q} \in B_d(0, \eta^*). \qquad (2.7)$$

Let $\mathbf{x}_\ell^* = u(\mathbf{q}_\ell)$, $\ell = 1, \cdots, d$, be chosen with the following properties:

$$\|\mathbf{q}_\ell\|_d \le \eta^*, \qquad \ell = 1, \cdots, d, \qquad (2.8)$$

and, with the matrix function U defined by

$$U_{i,j}(\mathbf{q}) = u^i(\mathbf{q}) - (\mathbf{x}_j^*)^i, \qquad (2.9)$$

we have

$$\|J(0)U(0)\mathbf{y}\|_d \ge \gamma > 0, \qquad \|\mathbf{y}\|_d = 1. \qquad (2.10)$$

Any set $\{\mathbf{x}_\ell^*\}$ with these properties will be called *coordinate stars* around $\mathbf{x}_0^*$. We note that the matrix $J(0)U(0)$ has columns given by $\pi_c(\mathbf{x}_0^* - \mathbf{x}_j^*)$, $j = 1, \cdots, d$, and hence, can be computed without reference to the map u. Let

$$\beta^* := (1/2) \min\left(\frac{1}{2\kappa}, \delta^*, \frac{\gamma}{8\sqrt{d}}\right). \qquad (2.11)$$

Theorem 2.2. *Let* $\Psi(\mathbf{q}) := (\|u(\mathbf{q}) - u(\mathbf{q}_\ell)\|_D^2)_{\ell=1}^d \in \mathbb{R}^d$. *Then*

(a) Ψ *is a diffeomorphism on* $B_d(0, 2\beta^*)$. *If* $\mathbf{p}, \mathbf{q} \in B_d(0, 2\beta^*)$, $\mathbf{x} = u(\mathbf{p})$, $\mathbf{y} = u(\mathbf{q})$, *then*

$$\frac{\gamma}{2} \rho(\mathbf{x}, \mathbf{y}) \le \|\Psi(\mathbf{p}) - \Psi(\mathbf{q})\|_d \le 32\sqrt{d}\eta^* \rho(\mathbf{x}, \mathbf{y}). \qquad (2.12)$$

(b) *The function* Ψ *is a diffeomorphism from* $B_d(0, \beta^*)$ *onto* $B_d(\Psi(0), \beta^*)$.

Remark 2.1. Let $\mathbb{B} = u(B_d(0, \beta^*)) \subset \mathbb{X}$ be a geodesic ball around $\mathbf{x}_0^*$. For $\mathbf{x} \in \mathbb{B}$, we define

$$\phi(\mathbf{x}) = \Psi(u^{-1}(\mathbf{x})) = (\|\mathbf{x} - \mathbf{x}_\ell^*\|_D^2).$$

Then Theorem 2.2(b) shows that ϕ is a diffeomorphism from $\mathbb{B}$ onto $B_d(\Psi(0), \beta^*)$. Since $\Psi(0) = (\|\mathbf{x}_0^* - \mathbf{x}_\ell^*\|_D^2)$,

$$\Phi(\mathbf{x}) = \frac{\sqrt{d}}{\beta^*}(\phi(\mathbf{x}) - \Psi(0)), \qquad \mathbf{x} \in \mathbb{B}$$

maps $\mathbb{B}$ diffeomorphically onto $B_d(0, \sqrt{d}) \supset [-1,1]^d$. Let $\mathbb{U} = \Phi^{-1}([-1,1]^d)$. Then $\mathbb{U}$ is a neighborhood of $\mathbf{x}_0^*$ and Φ maps $\mathbb{U}$ diffeomorphically onto $[-1,1]^d$. We oberve that $\mathbb{X}$ is a union of finitely many neighborhoods of the form $\mathbb{U}$, so that any $\mathbf{x} \in \mathbb{X}$ belongs to at least one such neighborhood. Moreover, $\Phi(\mathbf{x})$ can be computed entirely in terms of the description of $\mathbf{x}$ in terms of its D-dimensional coordinates. $\square$

Remark 2.2. The trainable parameters are thus β^*, and the points $\mathbf{x}_0^*, \cdots, \mathbf{x}_d^*$. Since $\|J(0)\| = 1$, the condition (2.10) is satisfied if $\mathbf{x}_\ell^* - \mathbf{x}_0^*$ are along linearly independent directions as in Theorem 2.1. $\square$

Remark 2.3. Since the mapping Φ in Remark 2.1 is a quadratic polynomial in $\mathbf{x}$, it can be implemented as a neural network with a single hidden layer using the activation function given in (2.22) as described in section 2.4.

Example 2.1. Let $0 < a < 1$, and $M = M_a$ be the circular helix defined by

$$\mathbf{u}(s) = (\cos as, \sin as, \sqrt{1-a^2}\,s)^T.$$

Clearly, M is a one dimensional manifold, and s is the arclength parameter, measured from $(1,0,0)^T$. The curvature at any point is a^2. For any point $\mathbf{z}_0 \in M$, $U_{\mathbf{z}_0} = M$, with the diffeomorphism given by $\mathbf{u}(\tan^{-1}(\pi t/2))$, $t \in (-1,1)$. An interesting fact is that $\|\mathbf{u}(s + 2\pi) - \mathbf{u}(s)\|_3 = 2\pi\sqrt{1-a^2}$ can be made arbitrarily small by choosing a close to 1, even though the geodesic distance between $\mathbf{u}(s + 2\pi)$ and $\mathbf{u}(s)$ is 2π. Let $s_0 \in \mathbb{R}$, $s_0 + \pi/4 \le s_1 \le s_0 + 3\pi/8$, and $U := \{\mathbf{u}(s) : |s - s_0| \le \pi/8\}$. Let $\Psi(s) := \|\mathbf{u}(s) - \mathbf{u}(s_1)\|_3^2$. It is easy to calculate that

$$\Psi(s) = 2 - 2\cos(a(s - s_1)) + (1 - a^2)(s - s_1)^2,$$

so that

$$\Psi'(s) = 2a\sin(a(s - s_1)) + 2(1 - a^2)(s - s_1).$$

If $\mathbf{u}(s) \in U$, then $\pi/8 \le s_1 - s \le \pi/2$. Therefore, using the well known estimates

$$\frac{2\theta}{\pi} \le \sin(\theta) \le \theta, \qquad \theta \in [0, \pi/2],$$

we obtain that

$$|\Psi'(s)| \le 2a^2(s_1 - s) + 2(1 - a^2)(s_1 - s) \le \pi, \qquad (2.13)$$

and

$$|\Psi'(s)| \ge \frac{4a^2}{\pi}(s_1 - s) + 2(1 - a^2)(s_1 - s) \ge \frac{\pi}{8}(2 - 2a^2(1 - 2/\pi)) \ge 1/2. \qquad (2.14)$$

Hence, for any points $\mathbf{u}(t_1)$, $\mathbf{u}(t_2) \in U$, we have

$$(1/2)|t_1 - t_2| \le \left| \|\mathbf{u}(t_1) - \mathbf{u}(s_1)\|_3^2 - \|\mathbf{u}(t_2) - \mathbf{u}(s_1)\|_3^2 \right| \le \pi|t_1 - t_2|.$$

We note that the neighborhood U where this estimate holds and the constants are independent of the curvature. $\square$

The remainder of this section is devoted to the proof of Theorem 2.2.

Lemma 2.1. *Let* $\mathbf{q} \in B_d(0, \eta^*)$. *Each of the following statements hold for the matrix U defined in (2.9):*

$$\|U(\mathbf{q}) - U(0)\| \le 2\sqrt{d}\|\mathbf{q}\|_d, \qquad (2.15)$$

$$\|U(\mathbf{q})\| \le 2\sqrt{d}\,\max_{1 \le \ell \le d}\|\mathbf{q} - \mathbf{q}_\ell\|_d \le 4\sqrt{d}\eta^*, \qquad (2.16)$$

$$\|J(\mathbf{q})U(\mathbf{q}) - J(0)U(0)\| \le 4\sqrt{d}\|\mathbf{q}\|_d. \qquad (2.17)$$

With β^* as in (2.11), for $\|\mathbf{q}\|_d \le 2\beta^*$, $\|\mathbf{y}\|_d = 1$,

$$\|J(\mathbf{q})U(\mathbf{q})\mathbf{y}\|_d \ge \gamma/2. \qquad (2.18)$$

PROOF. In view of (2.6) and the mean value theorem, we have for $\|\mathbf{p}\|_d \le \eta^*$,

$$\|u(\mathbf{q}) - u(\mathbf{p})\|_D \le 2\|\mathbf{q} - \mathbf{p}\|_d. \qquad (2.19)$$

We observe further that for any integers m, ℓ, $U(\mathbf{q})_{m,\ell} - U(0)_{m,\ell} = u^m(\mathbf{q}) - u^m(0)$. Consequently, for any $\mathbf{y} \in \mathbb{R}^d$, $\|\mathbf{y}\|_d \le 1$,

$$\|(U(\mathbf{q}) - U(0))\mathbf{y}\|_D = \|u(\mathbf{q}) - u(0)\|_D \sum_{\ell=1}^{d} |y^\ell| \le 2\sqrt{d}\|\mathbf{q}\|_d\|\mathbf{y}\|_d.$$

This proves (2.15).

In view of (2.19), used with $\mathbf{q}_\ell$ in place of $\mathbf{p}$, $\ell = 1, \cdots, d$, we obtain for all $\mathbf{y} \in \mathbb{R}^d$, $\|\mathbf{y}\|_d \le 1$,

$$\left\| \sum_{\ell=1}^{d} \mathbf{y}^\ell(u(\mathbf{q}) - u(\mathbf{q}_\ell)) \right\|_D \le \sum_{\ell=1}^{d} |\mathbf{y}^\ell|\|u(\mathbf{q}) - u(\mathbf{q}_\ell)\|_D$$
$$\le 2\sqrt{d}\,\max_{1 \le \ell \le d}\|\mathbf{q} - \mathbf{q}_\ell\|_d.$$

This proves (2.16).

In view of (2.5), (2.16), (2.15), we obtain for $\|\mathbf{q}\|_d \le \eta^* \le 1/(2\kappa)$ that

$$\|J(\mathbf{q})U(\mathbf{q}) - J(0)U(0)\| =$$
$$\|(J(\mathbf{q}) - J(0))U(\mathbf{q}) + J(0)(U(\mathbf{q}) - U(0))\|$$
$$\le \|J(\mathbf{q}) - J(0)\|\|U(\mathbf{q})\| + \|J(0)\|\|U(\mathbf{q}) - U(0)\|$$
$$\le 4\sqrt{d}\eta^*\kappa\|\mathbf{q}\|_d + 2\sqrt{d}\|\mathbf{q}\|_d = 2\sqrt{d}(1 + 2\eta^*\kappa)\|\mathbf{q}\|_d$$
$$\le 4\sqrt{d}\|\mathbf{q}\|_d.$$

This proves (2.17). The estimate (2.18) follows easily from this and (2.10). $\square$

PROOF OF THEOREM 2.2. In this proof only, let $\mathcal{J}(\mathbf{q})$ be the Jacobian of Ψ: $\mathcal{J}_{i,j}(\mathbf{q}) = D_i(\|u(\mathbf{q}) - u(\mathbf{q}_j)\|^2)$. Then $\mathcal{J}(\mathbf{q}) =$

$2J(\mathbf{q})U(\mathbf{q})$. The estimate (2.10) shows that $\mathcal{J}(0)$ is invertible, and that $\|\mathcal{J}(0)^{-1}\| \leq 1/(2\gamma)$. The estimate (2.17) then shows that

$$\|\mathcal{J}(\mathbf{q}) - \mathcal{J}(0)\| \leq 1/(2\|\mathcal{J}(0)^{-1}\|), \qquad \|\mathbf{q}\|_d \leq 2\beta^*. \quad (2.20)$$

Therefore, the inverse function theorem as given in [50, Theorem 9.24 and its proof] implies that Ψ is a diffeomorphism on $B(0, 2\beta^*)$ as claimed. For $\|\mathbf{q}\|_d \leq 2\beta^*$, (2.18) shows that $\|\mathcal{J}(\mathbf{q})^{-1}\| \leq 1/\gamma$. Also, (2.16) and (2.6) show that $\|\mathcal{J}(\mathbf{q})\| \leq 16\sqrt{d}\eta^*$. Hence, the mean value theorem implies that

$$\gamma\|\mathbf{q} - \mathbf{p}\|_d \leq \|\Psi(\mathbf{q}) - \Psi(\mathbf{p})\|_d \leq 16\sqrt{d}\eta^*\|\mathbf{q} - \mathbf{p}\|_d. \quad (2.21)$$

Together with (2.7), this implies (2.12).

The part (b) follows also from [50, Theorem 9.24 and its proof] and (2.20).

□

2.3. Local Basis Functions

Having found a local coordinate map Φ on a neighborhood $\mathbb{U}$ of $\mathbf{x}$ on $\mathbb{X}$, the problem of extending f from $\{\mathbf{x}_j\} \cap \mathbb{U}$ to $\mathbb{U}$ is reduced to extending $f \circ \Phi$ from $\mathcal{C} = \{\mathbf{y}_j = \Phi(\mathbf{x}_j)\} \subset [-1,1]^d$, a classical approximation problem. There is, of course, 100+ years of research on this subject. We restrict ourselves to two examples, which can be implemented using neural networks with one or more hidden layers. One of the most popular activation function in the deep learning literature (e.g., [14]) is the rectified linear unit function

$$t_+ = \max(0, t).$$

Since this function is not continuously differentiable, there are some technical difficulties to use common algorithms like back-propagation with these activation functions. Although we do not need back-propagation in our theory, we prefer to deal with a *rectified quadratic unit function* defined for $t \in \mathbb{R}$ by

$$\sigma(t) = \begin{cases} t^2, & \text{if } t \geq 0, \\ 0, & \text{if } t < 0, \end{cases} \quad (2.22)$$

which is continuously differentiable on $\mathbb{R}$. Our theory will work in general with any activation function of order $k \geq 2$; i.e., with a function σ that satisfies

$$\lim_{t \to -\infty} \frac{\sigma(t)}{t^k} = 0, \qquad \lim_{t \to \infty} \frac{\sigma(t)}{t^k} = 1, \quad (2.23)$$

but for the sake of clarity of exposition, we will use only the activation function σ defined in (2.22).

2.4. Polynomials

The most basic class of classical approximants is the set of all polynomials. For $n > 0$, we denote the class of all algebraic polynomials of coordinatewise degree at most n in d variables by Π_n^d. (It is convenient to use the same notation also when n is not an integer; in this case, Π_n^d is just $\Pi_{\lfloor n \rfloor}^d$.)

The basic implementation of polynomials is given in [22, Proof of Theorem 3.1], where an explicit construction is given for finding the weights $\{\mathbf{w}_k\}$, the thresholds $\{b_k\}$ and the coefficients $\{a_k\}$ used in (2.24) below.

Theorem 2.3. *Let* $n > 0$, $N = 2^{\lceil \log_2 n \rceil}$, $P \in \Pi_N^d \supseteq \Pi_n^d$, *then there exist weights* $\mathbf{w}_1, \cdots, w_{dim(\Pi_N^d)}$ *and real numbers* $a_1, \cdots, a_{dim(\Pi_N^d)}, b_1, \cdots, b_{dim(\Pi_N^d)}$ *such that*

$$P(\mathbf{x}) = \sum_{k=1}^{dim(\Pi_N^d)} a_k(\mathbf{w}_k \cdot \mathbf{x} + b_k)^N, \qquad x \in \mathbb{R}^d. \quad (2.24)$$

Here, the weights $\{\mathbf{w}_k\}$ *and the thresholds* $\{b_k\}$ *are independent of* P *and the coefficients* $\{a_k\}$ *are linear functionals on* Π_N^d.

We observe that

$$t^2 = \sigma(t) + \sigma(-t), \qquad x \in \mathbb{R}, \quad (2.25)$$

while

$$t^N = ((t^2)^2 \cdots)^2, (\log_2 N \text{ times}), \quad (2.26)$$

so that the expression on the right hand side of (2.24) can be expressed as a neural network with $\log_2 N$ hidden layers.

We note that a neural network with one hidden layer is given in Mhaskar [30], but using a C^∞ activation function σ; e.g., $\sigma(t) = (1 + e^{-t})^{-1}$. This uses the fact that for $\mathbf{w}, \mathbf{x} \in \mathbb{R}^d$ and $b \in \mathbb{R}$, such that none of the derivatives $\sigma^{(j)}(b), j = 0, 1, \cdots$, equal to 0,

$$\left.\frac{\partial^{|\mathbf{k}|}}{\partial \mathbf{w}^{\mathbf{k}}} \sigma(\mathbf{w} \cdot \mathbf{x} + b)\right|_{\mathbf{w}=0} = \mathbf{x}^{\mathbf{k}} \sigma^{|\mathbf{k}|}(b). \quad (2.27)$$

A finite difference scheme to implement this differentiation yields a neural network with one hidden layer, containing exactly $dim(\Pi_n^d)$ neurons, and should be stable for C^∞ functions. If stability is a greater concern, then one may use other numerical differentiation schemes to implement this formula, e.g., spectral methods [51].

2.5. B-Splines

For $t \in \mathbb{R}$, and integer $m \geq 1$, let

$$t_+^m = \begin{cases} t^m, & \text{if } t \geq 0, \\ 0, & \text{otherwise.} \end{cases}$$

A tensor product cardinal B-spline at $\mathbf{y} \in [-1, 1]^d$ is defined by

$$N_m(\mathbf{y}) = \frac{1}{(m-1)!^d} \sum_{\substack{\mathbf{k} \in \mathbb{Z}, \\ \mathbf{k} \geq 0, \, |\mathbf{k}|_\infty \leq m}} (-1)^{|\mathbf{k}|_1} \prod_{j=1}^{d} \binom{m}{k_j} \prod_{j=1}^{d} (y_j - k_j)_+^m.$$

$$(2.28)$$

It is explained in Mhaskar [7, 33] that the quantity $N_m(\mathbf{y})$ can be computed using a neural network with a sigmoidal function of order $m - 1$ consisting of finitely many neurons arranged in multiple hidden layers (the number of neurons and layers depending on m and d alone). Thus, if m is a power of 2, then each of the terms $(y_j - k_j)_+^m$ can be implemented as an iterated power of $(y_j - k_j)_+^2$ (cf. (2.26).) The product of d such expressions can be implemented using either Theorem 2.3 as a network with mulitple hidden layer and utilizing the rectified quadratic unit function as the activation function, or a discretization of the formula (2.27) using a C^∞ sigmoidal function as explained in section 2.4.

2.6. Function Approximation

In this section, if $\mathbf{y} \in \mathbb{R}^d$, $|\mathbf{y}|_\infty$ is the ℓ^∞ norm of $\mathbf{y}$.

2.6.1. Spline Based Approximation

In [52, section 4.5], [53], a quasi-interpolatory spline function is defined by

$$Q_m(f)(\mathbf{y}) = \sum_{\mathbf{k} \in \mathbb{Z}^d} \lambda_m^*(f(\cdot + \mathbf{k}))N_m(\mathbf{y} + m/2 + \mathbf{k}), \qquad \mathbf{y} \in \mathbb{R}^d,$$

(2.29)

where λ_m^* are compactly supported linear functionals, designed specifically to ensure that $Q_m(P) = P$ for every polynomial P of coordinatewise degree at most $m - 1$ in d variables. With $Q_{m,h}(f)(\mathbf{y}) = Q_m(f(h(\cdot)))(\mathbf{y}/h)$, $h > 0$, one has the approximation bound for small h:

$$\max_{\mathbf{y} \in [-1,1]^d} |Q_{m,h}(f)(\mathbf{y}) - f(\mathbf{y})| = \mathcal{O}(h^m). \qquad (2.30)$$

The linear functionals λ_m^* are based on *finitely many* samples of f at the grid points in a compact subset of $\mathbb{Z}^d$. In our context, the data for approximating f is not in this form. Therefore, we may use the following algorithm given in Mhaskar et al. [54], where we assume that λ_m^* is scaled so as to be supported on $[-1,1]^d$.

Given: A set $\mathcal{C} = \{\boldsymbol{\xi}_j\}$ of points in $[-1,1]^d$. Let

$$\delta(\mathcal{C}) = \max_{\mathbf{x} \in [-1,1]^d} \min_{\boldsymbol{\xi} \in \mathcal{C}} |\mathbf{x} - \boldsymbol{\xi}|_\infty,$$

and $\delta(\mathcal{C})$ be sufficiently small.

Objective: To find real numbers $\{a_{\boldsymbol{\xi}}\}_{\boldsymbol{\xi} \in \mathcal{C}}$ such that the functional

$$\gamma(f) := \sum_{\boldsymbol{\xi} \in \mathcal{C}} a_{\boldsymbol{\xi}} f(\boldsymbol{\xi}) \qquad (2.31)$$

satisfies

$$\gamma(P) = \lambda_m^*(P), \quad \text{if } P \text{ is } d - \text{variate polynomial of}$$
$$\text{coordinatewise degree } \leq m - 1.$$

Steps:

1. Divide $[-1,1]^d$ into congruent subcubes of side not exceeding $2\delta(\mathcal{C})$.
2. Choose $\mathcal{C}_0 \subseteq \mathcal{C}$, so that each subcube has exactly one point of $\mathcal{C}_0$.
3. Solve the following (underdetermined) system of equations for the unknowns $a_{\boldsymbol{\xi}}$, $\boldsymbol{\xi} \in \mathcal{C}_0$.

$$\sum_{\boldsymbol{\xi} \in \mathcal{C}} a_{\boldsymbol{\xi}} \boldsymbol{\xi}^{\mathbf{k}} = \lambda_m^*((\cdot)^{\mathbf{k}}), \qquad |\mathbf{k}|_\infty \leq m - 1. \quad (2.32)$$

4. Set $a_{\boldsymbol{\xi}} := 0$ if $\boldsymbol{\xi} \in \mathcal{C} \setminus \mathcal{C}_0$.
5. Output $(a_{\boldsymbol{\xi}})_{\boldsymbol{\xi} \in \mathcal{C}}$.

Substituting γ in place of λ_m^* in the definition of $Q_m(f)$ yields the desired spline approximation

$$\tilde{Q}_m(f)(\mathbf{y}) = \sum_{\mathbf{k} \in \mathbb{Z}^d} \gamma(f(\cdot + \mathbf{k}))N_m(\mathbf{y} + m/2 + \mathbf{k}), \qquad \mathbf{y} \in \mathbb{R}^d,$$

(2.33)

and it is proved in Mhaskar et al. [54] that the estimate (2.30) holds with $\tilde{Q}_m$ replacing Q_m.

2.6.2. Polynomial Quasi-Interpolation

A standard method for polynomial approximation is to consider a filtered projection defined in (2.38) below.

The Chebyshev polynomials (of first kind) are defined recursively for $t \in \mathbb{R}$ and integer $m \geq 0$ by

$$T_0(t) = 1, \quad T_1(t) = t, \quad T_m(t) = 2tT_{m-1}(t) - T_{m-2}(t). \quad (2.34)$$

In terms of monomials, the Chebyshev polynomials are given by

$$T_{2n}(t) = \sum_{j=0}^{n} \frac{(-4)^j}{(2j)!} \prod_{\ell=1}^{j} (n^2 - (j - \ell)^2))t^{2j},$$

$$T_{2n+1}(t) = \sum_{j=0}^{n} \frac{(-1)^j(2n+1)^2}{(2j+1)!} \prod_{\ell=1}^{j} ((2n+1)^2 - (2j - 2\ell + 1)^2)t^{2j+1}.$$

(2.35)

For $\mathbf{y} \in \mathbb{R}^d$ and multi-integer $\mathbf{m} = (m^1, \cdots, m^d) \geq 0$, the tensor product Chebyshev polynomial is defined by

$$T_{\mathbf{m}}(\mathbf{y}) = \prod_{j=1}^{d} T_{m^j}(y^j). \qquad (2.36)$$

We choose a smooth low pass filter h; i.e., an even function $h : \mathbb{R} \to [0,1]$ such that $h(u) = 1$ if $|u| \leq 1/2$, and $h(u) = 0$ if $|u| \geq 1$, and abuse the notation as usual to define

$$h(\mathbf{u}) = \prod_{j=1}^{d} h(u^j).$$

With this filter, we define the kernel

$$\Phi_n(\mathbf{y}, \mathbf{t}) = \sum_{\mathbf{k} \in \mathbb{Z}^d} h(\mathbf{k}/n) T_{\mathbf{k}}(\mathbf{y}) T_{\mathbf{k}}(\mathbf{t}), \qquad n > 0, \; \mathbf{y}, \mathbf{t} \in [-1,1]^d.$$

(2.37)

Then the filtered projection operator is defined by

$$V_n(f)(\mathbf{y}) = \int_{[-1,1]^d} f(\mathbf{t})\Phi_n(\mathbf{y}, \mathbf{t}) \frac{d\mathbf{t}}{\sqrt{(1 - (t^1)^2) \cdots (1 - (t^d)^2)}},$$
$$\mathbf{y} \in [-1,1]^d, \qquad (2.38)$$

It is well known that if f is any continuous function on $[-1,1]^d$, then $\{V_n(f)\}$ converges uniformly to f at the near optimal rate of approximation. For example, if f has partial derivatives up to

order r in each variable, then analogously to (2.30), but for large n rather than small h,

$$\max_{\mathbf{y}\in[-1,1]^d} |V_n(f)(\mathbf{y}) - f(\mathbf{y})| = \mathcal{O}(n^{-r}). \qquad (2.39)$$

Theoretically, the question then is to compute $V_n(f)$ using the data $\mathcal{C}$ as in section 2.6.1. The procedure we describe below from Mhaskar [55, 56] and Mhaskar et al. [5] also describes the choice of the parameter n depending upon the data.

Given: A set $\mathcal{C} = \{\boldsymbol{\xi}_j\}$ of points in $[-1,1]^d$. Let

$$\delta^\circ(\mathcal{C}) = \max_{\mathbf{x}\in[-1,1]^d} \min_{\boldsymbol{\xi}\in\mathcal{C}} \max_{1\le j\le d} |\arccos(x^j) - \arccos(\xi^j)|.$$

and $\delta^\circ(\mathcal{C})$ be sufficiently small. We also fix an integer $n > 0$.

Objective: To find real numbers $\{w_{\boldsymbol{\xi}}\}_{\boldsymbol{\xi}\in\mathcal{C}}$ such that the functional

$$\gamma^\circ(f) := \sum_{\boldsymbol{\xi}\in\mathcal{C}} w_{\boldsymbol{\xi}} f(\boldsymbol{\xi}) \qquad (2.40)$$

satisfies

$$\gamma^\circ(P) = \int_{[-1,1]^d} \frac{P(\mathbf{t})d\mathbf{t}}{\sqrt{(1-(t^1)^2)\cdots(1-(t^d)^2)}},$$

for all d-variate polynomials P of coordinatewise degree $\le n - 1$.

Steps:

1. Divide $[-1,1]^d$ into congruent subcubes of side not exceeding $2\delta^\circ(\mathcal{C})$.
2. Choose $\mathcal{C}_0 \subseteq \mathcal{C}$, so that each subcube has exactly one point of $\mathcal{C}_0$.
3. Solve the following (underdetermined) system of equations for the unknowns $w_{\boldsymbol{\xi}}$, $\boldsymbol{\xi} \in \mathcal{C}_0$.

$$\sum_{\boldsymbol{\xi}\in\mathcal{C}} w_{\boldsymbol{\xi}} T_{\mathbf{k}}(\boldsymbol{\xi}) = \int_{[-1,1]^d} \frac{T_{\mathbf{k}}(\mathbf{t})d\mathbf{t}}{\sqrt{(1-(t^1)^2)\cdots(1-(t^d)^2)}}$$

$$= \delta_{\mathbf{k},0}, |\mathbf{k}|_\infty \le n-1. \qquad (2.41)$$

4. Set $w_{\boldsymbol{\xi}} := 0$ if $\boldsymbol{\xi} \in \mathcal{C} \setminus \mathcal{C}_0$.
5. Output $(w_{\boldsymbol{\xi}})_{\boldsymbol{\xi}\in\mathcal{C}}$.

It is proved in the papers cited above that with the discretized operator

$$V_{n,\mathcal{C}}(f)(\mathbf{y}) = \sum_{\boldsymbol{\xi}\in\mathcal{C}} w_{\boldsymbol{\xi}} f(\boldsymbol{\xi}) \Phi_n(\mathbf{y}, \boldsymbol{\xi}), \qquad (2.42)$$

one obtains the near best rates of approximation. In particular, if f has partial derivatives up to order r in each variable, then (2.39) holds with $V_{n,\mathcal{C}}(f)$ replacing $V_n(f)$. In practice, we choose n to be the largest integer such that either the condition number of the

system of equations in (2.41) is "reasonable" or else by checking the resulting errors in (2.41) [57].

The formula (2.42) can be re-written in the form

$$V_{n,\mathcal{C}}(f)(\mathbf{y}) = \sum_{|\mathbf{k}|_\infty \le n-1} h(\mathbf{k}/n)\hat{f}(\mathcal{C}, \mathbf{k}) T_{\mathbf{k}}(\mathbf{y}), \qquad (2.43)$$

where

$$\hat{f}(\mathcal{C}, \mathbf{k}) = \sum_{\boldsymbol{\xi}\in\mathcal{C}} w_{\boldsymbol{\xi}} f(\boldsymbol{\xi}) T_{\mathbf{k}}(\boldsymbol{\xi}). \qquad (2.44)$$

Therefore, rather than evaluating the Chebyshev polynomials as defined in (2.36), (2.35), one can use a multi-layered network to evaluate $V_{n,\mathcal{C}}(f)$ in a more stable manner as follows. The first layer computes the coefficients $h(\mathbf{k}/n)\hat{f}(\mathcal{C}, \mathbf{k})$ using the available data. The output of this layer is input to a recurrent network to execute a multi-variate version of the well known Clenshaw algorithm [58, pp. 78–80].

3. EXTENSIONS

Since the starting point of diffusion geometry is to consider eigen-decomposition of a diffusion matrix, which is constructed using the available data, the entire computation needs to be redone if a new data becomes available. Since the manifold $\mathbb{X}$ is only an abstract model, it is not even clear that the new data will belong to this manifold. This gives rise to two related questions. One is to find new points on the manifold; the so called pre-image problem [41] and the other is the out-of-sample extension problem; i.e., extend the target function to points not necessarily on $\mathbb{X}$. In this section, we make some comments on how to use the theory described in the previous sections for solving these problems.

3.1. Pre-image Problem

In Remark 2.1, we have given an onto diffeomorphism $\Phi : \mathbb{U} \to [-1,1]^d$ where $\mathbb{U} \subset \mathbb{X}$. The pre-image problem is the following. Given a point $\mathbf{y} \in [-1,1]^d$, find $\mathbf{x} \in \mathbb{X}$ such that $\Phi(\mathbf{x}) = \mathbf{y}$. This amounts to approximating the D-output function Φ^{-1} on $[-1,1]^d$, given its values at the known points $\{\Phi(\boldsymbol{\xi}) : \boldsymbol{\xi} \in \mathcal{C}\}$, and can therefore be solved using any of the techniques described in the previous sections.

3.2. Out of Sample Extension

One well known strategy for function extension outside the manifold is the following. One starts with a compact, positive-semi-definite symmetric kernel $K : \mathbb{R}^D \times \mathbb{R}^D$ and considers the eigen-decomposition of K restricted to $\mathbb{X} \times \mathbb{X}$; thus, for example, if μ^* is the volume measure on $\mathbb{X}$, one finds numbers $\lambda_k \ge 0$ and orthonormal functions ϕ_k on $\mathbb{X}$ such that

$$\int_{\mathbb{X}} K(\mathbf{x}, \mathbf{y})\phi_k(\mathbf{y})d\mu^*(\mathbf{y}) = \lambda_k \phi_k(\mathbf{x}), \qquad k = 0, 1, \cdots, \mathbf{x} \in \mathbb{X}.$$
$$(3.1)$$

A function on $\mathbb{X}$ can then be expanded in terms of the orthonormal system of functions $\{\phi_k\}$. The extension to $\mathbb{R}^D$ is achieved by treating (3.1) as a definition of ϕ_k on $\mathbb{R}^D$ (which can be done since K is defined on $\mathbb{R}^D \times \mathbb{R}^D$), and the expansion

of the original function f where the basis functions are now interpreted as extended by (3.1) as the desired extension. This leads to a variety of theoretical problems related to the judicious construction of kernels defined on the whole space whose eigenfunctions are meaningful as functions on $\mathbb{X}$ (e.g., kernels that commute with the Laplace-Beltrami operator) so as to allow such a construction. In the end, it is not clear how well this extension will behave outside of $\mathbb{X}$.

Our construction gives an alternative method for extending a function on $\mathbb{X}$ to a tubular neighborhood of $\mathbb{X}$, which we feel is more appropriate for most applications rather than trying to extend the function to the entire ambient space. Toward this goal, we first explain the local coordinate learning phase for tubular neighborhood of $\mathbb{X}$.

Let $s \geq d$ be an integer, $s \leq D$. For $\mathbf{q} \in \mathbb{R}^s$ (or $\mathbf{q} \in \mathbb{R}^D$), we write $\pi_c(\mathbf{q}) = (q^1, \cdots, q^d)$, and

$$v(\mathbf{q}) = u(\pi_c(\mathbf{q})) + (\underbrace{0, \cdots, 0}_{d \text{ times}}, q^{d+1}, \cdots, q^s, \underbrace{0, \cdots, 0}_{D-s \text{ times}}) \in \mathbb{R}^D.$$

If J_s is the Jacobian matrix for v, i.e., $(J_s)_{i,j}(\mathbf{q}) = D_i v^j(\mathbf{q})$, then it is easy to check that

$$J_s(0) = [I_s | 0_{s,D-s}]. \tag{3.2}$$

Moreover, if $\mathbf{p}, \mathbf{q} \in B_d(0, \delta^*)$, then $\mathbf{p} = \pi_c(\mathbf{p})$, $\mathbf{q} = \pi_c(\mathbf{q})$. Therefore,

$$(J_s(\mathbf{q}) - J_s(\mathbf{p}))(z) = (J_d(\pi_c(\mathbf{q})) - J_d(\pi_c(\mathbf{p})))(z),$$
$$\mathbf{p}, \mathbf{q} \in B_d(0, \delta^*), \ z \in \mathbb{R}^D. \tag{3.3}$$

Consequently,

$$\|J_s(\mathbf{q}) - J_s(0)\| \leq \kappa \|\pi_c(\mathbf{q})\|_d \leq \kappa \|\mathbf{q}\|_s. \tag{3.4}$$

If we now define for $\mathbf{p}, \mathbf{q} \in B_d(0, \delta^*)$

$$\rho_1(v(\mathbf{p}), v(\mathbf{q}))^2 := \rho(u(\pi_c(\mathbf{p})), u(\pi_c(\mathbf{q})))^2 + \sum_{k=d+1}^{s} |p^k - q^k|^2,$$

then following the same argument as the one leading to (2.7) leads to

$$(1/2)\rho_1(v(\mathbf{p}), v(\mathbf{q})) \leq \|\mathbf{p} - \mathbf{q}\|_s \leq 2\rho_1(v(\mathbf{p}), v(\mathbf{q})), \quad \|\mathbf{p}\|_s, \|\mathbf{q}\|_s \leq \eta^*.$$

Thus, there is no loss of generality in assuming that $\mathbb{X}$ is already a s dimensional submanifold of $\mathbb{R}^D$, defined by v, and with geodesic distance ρ_1. This has several consequences. Even if one overestimates the dimension of the original manifold to be s rather than d, the resulting "distance respecting" coordinate system will also be "distance respecting" for the original manifold, except for the presumably small error resulting from the overestimate. If we have no information about d (or s), we may take $s = D$. This would answer the question regarding points off the manifold, as well as take noise into account. However, then the advantage of dimension reduction is lost. Also, all the constants will depend upon D rather than s (or d).

Having defined a local coordinate system for the tubular neighborhood of $\mathbb{X}$ in this way, we can then construct the local basis functions on this neighborhood as in section 2.3. However, since the original data $\mathcal{C}$ is not dense on the local coordinate patch in the tubular neighborhood, one cannot use the ideas in section 2.6. We would like instead to keep some control on the growth of the extension operator. For this purpose, we propose to use the minimal Sobolev norm (MSN) interpolant introduced in Chandrasekaran et al. [59] and used very fruitfully in solutions of partial differential equations [60] and image segmentation [61].

Thus, using the procedures explained in these papers, we consider a differential operator Δ depending upon the application, and find an integer N and coefficients $c_{\mathbf{k}}^*$, $|\mathbf{k}|_\infty \leq N$, so as the s-variate polynomial $P^* = \sum_{\mathbf{k}: |\mathbf{k}|_\infty \leq N} c_{\mathbf{k}}^* T_{\mathbf{k}}$ minimizes

$$\int_{[-1,1]^s} |\Delta P(\mathbf{t})|^2 \frac{d\mathbf{t}}{\sqrt{(1 - (t^1)^2) \cdots (1 - (t^s)^2)}} \tag{3.5}$$

over all s-variate polynomials P of coordinatewise degree $\leq N$, subject to the conditions

$$P(\Phi(\mathbf{x}_j)) = f(\mathbf{x}_j), \qquad \mathbf{x}_j \in \mathcal{C}. \tag{3.6}$$

The polynomial P^* then defines an extension of f to the tubular neighborhood of the local coordinate patch of $\mathbb{X}$.

AUTHOR CONTRIBUTIONS

All authors listed have made a substantial, direct and intellectual contribution to the work, and approved it for publication.

ACKNOWLEDGMENTS

REFERENCES

1. Vapnik V. *The Nature of Statistical Learning Theory*. New York, NY: Springer Science & Business Media (2013).

2. Tiao GC, Tsay RS. Model specification in multivariate time series. *J Roy Statist Soc Ser B* (1989) **51**:157–213.

3. Chakraborty K, Mehrotra K, Mohan CK, Ranka S. Forecasting the behavior of multivariate time series using neural networks. *Neural Netw.* (1992) **5**:961–70. doi: 10.1016/S0893-6080(05)80092-9

4. Naumova V, Pereverzyev SV, Sivananthan S. Adaptive parameter choice for one-sided finite difference schemes and its application in diabetes technology. *J Complex.* (2012) **28**:524–38. doi: 10.1016/j.jco.2012.06.001

5. Mhaskar HN, Naumova V, Pereverzyev SV. Filtered Legendre expansion method for numerical differentiation at the boundary point with application to blood glucose predictions. *Appl Math Comput.* (2013) **224**:835–47. doi: 10.1016/j.amc.2013.09.015

6. Sharda R, Delen D. *Forecasting Box-Office Receipts of Motion Pictures Using Neural Networks.* Stillwater, OK: Citeseer (2002).

7. Mhaskar HN. Approximation properties of a multilayered feedforward artificial neural network. *Adv Comput Math.* (1993) **1**:61–80. doi: 10.1007/BF02070821

8. Maggioni M, Mhaskar HN. Diffusion polynomial frames on metric measure spaces. *Appl Comput Harm Anal.* (2008) **24**:329–53. doi: 10.1016/j.acha.2007.07.001

9. Ehler M, Filbir F, Mhaskar HN. Locally Learning Biomedical Data Using Diffusion Frames. *J Comput Biol.* (2012) **19**:1251–64. doi: 10.1089/cmb.2012.0187

10. Chui CK, Filbir F, Mhaskar HN. Representation of functions on big data: graphs and trees. *Appl Comput Harm Anal.* (2015) **38**:489–509. doi: 10.1016/j.acha.2014.06.006

11. Vapnik VN, Vapnik V. *Statistical Learning Theory.* New York, NY: Wiley (1998).

12. Girosi F, Poggio T. Networks and the best approximation property. *Biol Cybernet.* (1990) **63**:169–76. doi: 10.1007/BF00195855

13. Girosi F, Jones MB, Poggio T. Regularization theory and neural networks architectures. *Neural Comput.* (1995) **7**:219–69. doi: 10.1162/neco.1995.7.2.219

14. LeCun Y, Bengio Y, Hinton G. Deep learning. *Nature* (2015) **521**:436–44. doi: 10.1038/nature14539

15. Rosasco L, Shashua A, Cohen N, Poggio T. *Notes on Hierarchical Splines, DCLNs, Convolutional Kernels and i-Theory.* Boston, MA: Manuscript, Center for Brains, Minds, and Machines, MIT (2015).

16. Lorentz GG, von Golitschek M, Makovoz Y. *Constructive Approximation: Advanced Problems.* Berlin: Springer (1996).

17. Minsky M, Papert S. *Perceptrons: An Introduction to Computational Geometry.* Cambridge, MA: MIT Press (1988).

18. Funahashi KI. On the approximate realization of continuous mappings by neural networks. *Neural Netw.* (1989) **2**:183–92. doi: 10.1016/0893-6080(89)90003-8

19. Irie B, Miyake S. Capabilities of three-layered perceptrons. In: *Neural Networks, 1988., IEEE International Conference on. IEEE* (1988). p. 641–8.

20. Hornik K, Stinchcombe M, White H. Multilayer feedforward networks are universal approximators. *Neural Netw.* (1989) **2**:359–66. doi: 10.1016/0893-6080(89)90020-8

21. Cybenko G. Approximation by superposition of sigmoidal functions. *Math Contl Signals Syst.* (1989) **2**:303–14. doi: 10.1007/BF02551274

22. Chui CK, Li X. Approximation by ridge functions and neural networks with one hidden layer. *J Approx Theory* (1992) **70**:131–41. doi: 10.1016/0021-9045(92)90081-X

23. Chui CK, Li X. *Realization of Neural Networks with One Hidden Layer.* Singapore: Multivariate Approximation: From CAGD to Wavelets, World Scientific, (1993). p. 77–89.

24. Mhaskar HN, Micchelli CA. Approximation by superposition of sigmoidal and radial basis functions. *Adv Appl Math.* (1992) **13**:350–73. doi: 10.1016/0196-8858(92)90016-P

25. Leshno M, Lin VY, Pinkus A, Schocken S. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Netw.* (1993) **6**:861–7. doi: 10.1016/S0893-6080(05)80131-5

26. Barron AR. Universal approximation bounds for superpositions of a sigmoidal function. *Inf Theory IEEE Trans.* (1993) **39**:930–45. doi: 10.1109/18.256500

27. Kurková V, Sanguineti M. Bounds on rates of variable basis and neural network approximation. *IEEE Trans Inf Theory* (2001) **47**:2659–65. doi: 10.1109/18.945285

28. Kurková V, Sanguineti M. Comparison of worst case errors in linear and neural network approximation. *IEEE Trans Inf Theory* (2002) **48**:264–75. doi: 10.1109/18.971754

29. Mhaskar HN. On the tractability of multivariate integration and approximation by neural networks. *J Complex.* (2004) **20**:561–90. doi: 10.1016/j.jco.2003.11.004

30. Mhaskar HN. Neural networks for optimal approximation of smooth and analytic functions. *Neural Comput.* (1996) **8**:164–77. doi: 10.1162/neco.1996.8.1.164

31. Chui CK, Li X, Mhaskar HN. Neural networks for localized approximation. *Math Comput.* (1994) **63**:607–23. doi: 10.1090/S0025-5718-1994-1240656-2

32. Chui CK, Li X, Mhaskar HN. Limitations of the approximation capabilities of neural networks with one hidden layer. *Adv Comput Math.* (1996) **5**:233–43. doi: 10.1007/BF02124745

33. Mhaskar HN. Neural networks for localized approximation of real functions. In: *Neural Networks for Processing [1993] III. Proceedings of the 1993 IEEE-SP Workshop.* IEEE (1993). p. 190–96.

34. Mhaskar HN, Khachikyan L. Neural networks for function approximation. In: *Neural Networks for Signal Processing [1995] V. Proceedings of the 1995 IEEE Workshop.* IEEE (1995). p. 21–9.

35. Chui CK, Donoho DL. Special issue: diffusion maps and wavelets. *Appl. Comput. Harm. Anal.* (2006) **21**:1–2. doi: 10.1016/j.acha.2006.05.005

36. Belkin M, Niyogi P. Towards a theoretical foundation for Laplacian-based manifold methods. *J Comput Syst Sci.* (2008) **74**:1289–308. doi: 10.1016/j.jcss.2007.08.006

37. Lafon SS. *Diffusion Maps and Geometric Harmonics.* Yale, MI: Yale University (2004).

38. Singer A. From graph to manifold Laplacian: the convergence rate. *Appl Comput Harm Anal.* (2006) **21**:128–34. doi: 10.1016/j.acha.2006.03.004

39. Jones PW, Maggioni M, Schul R. Manifold parametrizations by eigenfunctions of the Laplacian and heat kernels. *Proc Natl Acad Sci USA.* (2008) **105**:1803–8. doi: 10.1073/pnas.0710175104

40. Jones PW, Maggioni M, Schul R. Universal local parametrizations via heat kernels and eigenfunctions of the Laplacian. *Ann Acad Sci Fenn Math.* (2010) **35**:131–74. doi: 10.5186/aasfm.2010.3508

41. Mishne G, Shaham U, Cloninger A, Cohen I. Diffusion nets. arXiv:150607840v1 (2015).

42. Filbir F, Mhaskar HN. A quadrature formula for diffusion polynomials corresponding to a generalized heat kernel. *J Fourier Anal Appl.* (2010) **16**:629–57. doi: 10.1007/s00041-010-9119-4

43. Filbir F, Mhaskar HN. Marcinkiewicz–Zygmund measures on manifolds. *J Complex.* (2011) **27**:568–96. doi: 10.1016/j.jco.2011.03.002

44. Mhaskar HN. Eignets for function approximation on manifolds. *Appl Comput Harm Anal.* (2010) **29**:63–87. doi: 10.1016/j.acha.2009.08.006

45. Mhaskar HN. A generalized diffusion frame for parsimonious representation of functions on data defined manifolds. *Neural Netw.* (2011) **24**:345–59. doi: 10.1016/j.neunet.2010.12.007

46. Chui CK, Mhaskar HN. Smooth function extension based on high dimensional unstructured data. *Math Comput.* (2014) **83**:2865–91. doi: 10.1090/S0025-5718-2014-02819-6

47. Mhaskar HN. A unified framework for harmonic analysis of functions on directed graphs and changing data. *Appl Comput Harm Anal.* (2018) **44**:611–644. doi: 10.1016/j.acha.2016.06.007

48. Coifman RR, Lafon S. Geometric harmonics: a novel tool for multiscale out-of-sample extension of empirical functions. *Appl Comput Harm Anal.* (2006) **21**:31–52. doi: 10.1016/j.acha.2005.07.005

49. do Carmo Valero MP. *Riemannian Geometry.* Birkhäuser (1992).

50. Rudin W. *Principles of Mathematical Analysis (International Series in Pure & Applied Mathematics).* Singapore: McGraw-Hill Publishing Co. (1976).

51. Gottlieb D, Orszag SA. *Numerical Analysis of Spectral Methods: Theory and Applications.* Siam (1977).

52. Chui CK. *An Introduction to Wavelets.* San Diego, CA:Academic Press (1992).

53. Chui CK, Diamond H. A characterization of multivariate quasi-interpolation formulas and its applications. *Numerische Mathematik.* (1990) **57**:105–21. doi: 10.1007/BF01386401

54. Mhaskar HN, Narcowich FJ, Ward JD. Quasi-interpolation in shift invariant spaces. *J Math Anal Appl.* (2000) **251**:356–63. doi: 10.1006/jmaa.2000.7051

55. Mhaskar HN. Approximation theory and neural networks. In: *Wavelet Analysis and Applications, Proceedings of the International Workshop*, Delhi (1999). p. 247–89.

56. Mhaskar HN. Polynomial operators and local smoothness classes on the unit interval, II. *Jaén J Approx.* (2009) **1**:1–25.

57. Le Gia QT, Mhaskar HN. Localized linear polynomial operators and quadrature formulas on the sphere. *SIAM J Numer Anal.* (2009) **47**:440–66. doi: 10.1137/060678555

58. Gautschi W. *Orthogonal Polynomials: Computation and Approximation.* Oxford University Press on Demand (2004).

59. Chandrasekaran S, Jayaraman KR, Mhaskar HN. Minimum Sobolev norm interpolation with trigonometric polynomials on the torus. *J Comput Phys.* (2013) **249**:96–112. doi: 10.1016/j.jcp.2013.03.041

60. Chandrasekaran S, Jayaraman KR, Gu M, Mhaskar HN, Mofftt J. Higher order numerical discretization methods with sobolev norm minimization. *Proc Comput Sci.* (2011) **4**:206–15. doi: 10.1016/j.procs.2011.04.022

61. Chandrasekaran S, Jayaraman KR, Moffitt J, Mhaskar HN, Pauli S. Minimum Sobolev Norm schemes and applications in image processing.In: *IS&T/SPIE Electronic Imaging. International Society for Optics and Photonics* (2010). p. 753507.

62. Chui CK, Mhaskar HN. Deep nets for local manifold learning. arXiv:160707110 (2016).

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.