Signal Processing 216 (2024) 109308

Contents lists available at ScienceDirect

Signal Processing

journal homepage: www.elsevier.com/locate/sigpro

# Graph signal interpolation and extrapolation over manifold of Gaussian mixture

Itay Zach [a,*], Tsvi G. Dvorkind [b], Ronen Talmon [a]

[a] Viterbi Faculty of Electrical and Computer Engineering, Technion - Israel Institute of Technology, Haifa, 3200003, Israel
[b] RAFAEL Advanced Defense Systems LTD, Haifa, 31021, Israel

ARTICLE INFO

Dataset link: github.com/itayzach/RoMix

*Keywords:*
Graph signal processing
Graph signal interpolation
Spectral graph theory
Reproducing kernel Hilbert space
Gaussian mixture model
Data manifolds

ABSTRACT

Signals with an underlying irregular geometric structure are prevalent in modern applications and are often well-represented using graphs. The field of graph signal processing has emerged to accommodate such signals' analysis, processing, and interpolation. In this paper, we address the latter, where signal samples are given only on a subset of graph nodes, and the goal is to estimate the graph signal on the remaining nodes. In addition, we consider a related extrapolation task in which new graph nodes that were not part of the original graph are added, and the goal is to estimate the graph signal on those nodes as well. We present a new approach for both the interpolation and extrapolation tasks, which is based on modeling the graph nodes as samples from a continuous manifold with a Gaussian mixture distribution and the graph signal as samples of a continuous function in a reproducing kernel Hilbert space. This model allows us to propose an interpolation and extrapolation algorithm that utilizes the closed-form expressions of the Gaussian kernel eigenfunctions. We test our algorithm on synthetic and real-world signals and compare it to existing methods. We demonstrate superior or on-par accuracy results achieved in significantly shorter run times.

## 1. Introduction

### 1.1. Graph signal processing overview

Classical Digital Signal Processing (DSP) based on Fourier analysis has been the pillar framework for processing digital signals in numerous fields for the last decades. Communication systems, RADAR applications, and medical imaging are just a handful of examples of DSP's vast influence. The signals in these fields are prototypically sampled from uniform, regular grids, e.g., linear or spherical sensor arrays, where the classical DSP tools naturally apply. However, signals in various other fields describing biological connections, transportation systems, and sensor networks often do not follow a similar grid structure but rather have an irregular network structure. Therefore, classical DSP tools cannot be applied to these signals as is. Recently, the emerging field of Graph Signal Processing (GSP) [1,2] promoted the representation of irregular structures of networks using graphs [3], where the network elements are defined as the graph nodes and their mutual relationships by the graph edges. By associating the signal coordinates to the graph's nodes, an object termed *graph signal* is defined. To process and analyze these graph signals, classical DSP tools, such as filtering, denoising, and the Fourier and Wavelet transforms, have been extended to signals defined over graphs. These extensions have opened the door to various transformative applications, such as the prediction of protein-to-protein interactions in biological networks [4], analysis of traffic congestion in transportation systems [5], and power grid load monitoring [6], to name just a few. Furthermore, these GSP developments facilitated the emergence of geometric deep learning [7].

### 1.2. Graph signal interpolation and extrapolation

In this work, we focus on two of the most prominent tasks in the field of GSP: graph signal interpolation and graph signal extrapolation. In the first task, only a subset of the graph signal values is given, and the goal is to estimate the graph signal on the remaining set of nodes. For instance, consider the problem of estimating the received power of a Radio Frequency (RF) signal in a vast variable topographic terrain based on measurements from only a handful of scattered sensors. The geographic landscape and the sensor network constellation can be described using a graph as follows. Each graph node is attributed to a set of features of a point in the terrain, e.g., its $x, y, z$ coordinates. The graph edges represent some notion of affinity between the terrain points' features. Accordingly, the graph signal is defined on the graph nodes as the received signal power in each point, which is known only on the nodes corresponding to the sensor network. Estimating the

* Corresponding author.
*E-mail address:* itay.zach@campus.technion.ac.il (I. Zach).

https://doi.org/10.1016/j.sigpro.2023.109308
Received 1 November 2022; Received in revised form 20 October 2023; Accepted 26 October 2023
Available online 30 October 2023

received signal power over the entire terrain can then be recast as a task of graph signal interpolation. We will address this application in our work, yet, there exist various other applications of graph signal interpolation in the literature. For instance, matrix completion for recommendation systems [8,9], temperature recovery [10] and power restoration in energy harvesting sensor networks [11].

In the second task, graph signal extrapolation, the graph structure is expanded beyond the given set of nodes and edges, and the goal is to estimate the graph signal values of those newly introduced nodes. Unlike the interpolation task, the nodal information and connectivity of the joining nodes are not given in advance. The extrapolating function should infer the graph signal values of the joining nodes based on the original graph signal and graph structure alone, in an online fashion, without a change of parameters. For instance, following the RF signal power estimation example, expanding the terrain with new points and estimating the signal power on them can be recast as a graph signal extrapolation problem.

### 1.3. Literature review

Previous methods that address graph signal interpolation can be roughly divided into two categories: methods that learn the graph topology jointly with the interpolation procedure (e.g., [12,13]) and methods that assume some a-priori known graph topology. We focus on the latter. In some works (e.g., [14,15]), the interpolation procedure highly depends on the sampling process of the graph signal, namely, the subset of nodes on which the graph signal is known. In these works, the proposed interpolating algorithms might be irrelevant if the subset is given at random. Conversely, other works consider the case of an arbitrary subset of available values from which the graph signal is interpolated and can therefore be more robust. For instance, [16,17] utilize Diffusion maps [18] for the interpolation procedure, where [16] also combines the Nyström method [19] in the interpolation process. The work in [9] relies on the eigenvectors of the graph Laplacian matrix to impose smoothness on the interpolated graph signal. The work in [20] employs a variational spline in Paley–Wiener space as the model for the interpolating function [21], while utilizing Green's function of the regularized Laplacian. In a recent line of work, e.g., [22–24], the interpolation is performed in a Reproducing Kernel Hilbert Space (RKHS) [25–28]. In [22,23], the Representer theorem [29] is used, and in [24], a basis for the RKHS termed "graph basis functions" is defined. For a comprehensive review of graph signal interpolation methods, see [30].

The extrapolation problem is also discussed in the literature, however not always addressed together with an accompanying interpolation procedure. In [31,32], for instance, the authors suggest schemes where the connectivity of the newly introduced nodes is learned from data. In [31], the authors apply kernel regression over graphs to obtain the extrapolated graph signal. The method accounts for the possibility of the extrapolated graph signal being generated by a different physical phenomenon than the given measurements. In [33], the authors use a directed graph from which Diffusion maps embeddings are obtained. Then, the authors apply the Nyström method to achieve the extrapolated graph signal. In [34] the authors follow the work of [31], but approximate the kernel using Random Fourier Features (RFF) [35]. While all the aforementioned works present extrapolation schemes, the designed models are discrete. Moreover, neither provides an interpolation procedure to accommodate missing graph signal values.

Nevertheless, various other works suggest an extrapolation procedure accompanied by an interpolation method that indeed accommodates such missing graph signal values. For example, in [36], the authors perform matrix completion (i.e., interpolation) and extrapolation in an RKHS composed from the Kronecker product of the matrix rows RKHS and matrix columns RKHS. For instance, in a user-movie rating matrix completion and extrapolation task, the function space over the users would be the rows RKHS, and the function space over the movies would be along the columns. The work in [37] also utilizes an RKHS model to interpolate and extrapolate graph signals. The authors suggest a privacy-aware method that relies on RFF with various kernels while utilizing an online Multi Kernel Learning (MKL) framework to obtain the interpolated and extrapolated graph signals. While these works provide both interpolation and extrapolation procedures, they assume the graph affinities are known in advance. Moreover, neither suggests a continuous domain approach nor a parametric model for the graph nodes' distribution.

### 1.4. Motivation

In order to capture the underlying geometric structure of data and incorporate it into interpolation and extrapolation schemes, various works in the GSP literature utilize a kernel in an RKHS. The construction of the kernel varies across methods and applications and depends on whether or not the affinities in the data are naturally defined.

When the affinities of the graph are given, or naturally arise from the problem at hand, the kernel can be constructed as a function of the known affinities (e.g., [22–24,31,34,36,37]). For instance, in [22,23] the kernel is constructed as a function of the Laplacian, and in [37] the kernel is constructed as a function of the adjacency matrix. In [24], the connectivity is captured by translations of a graph basis function, where the translations are inferred from the Laplacian eigenvectors. For translations on graphs see e.g. [1].

Conversely, in many applications, geometric structure exists in the data but is not immediately apparent. A common assumption in these cases, termed the *manifold assumption*, is to assume that the data lies on a low dimensional manifold [38–40]. A natural way to approximate the manifold is by constructing a graph using a kernel function [1]. For instance, in the sensors network presented in Section 1.2, it is possible to reveal the underlying geometric structure by constructing a graph where each node describes a sensor. Some properties of the sensors can be aggregated into a feature vector for each sensor (such as its $x, y, z$ coordinates or line of sight existence), and then incorporate the feature vectors into a kernel function. The adjacency matrix of the graph can then be constructed from the kernel function.

All the aforementioned works that utilize an RKHS for interpolation and extrapolation (cf. [22–24,31,34,36,37]) assume the affinities between the graph nodes are known, or naturally defined. Furthermore, the previous works obtain a discrete perspective for their models and neither provides a model for the nodal information probability distribution. In this work, we present a continuous domain approach, in which we capture the underlying structure and nodal information by the eigenfunctions of a kernel, and by a parametric model for the nodal distribution. As the structural information is captured in traditional GSP by the adjacency matrix and its eigenvectors, or the Laplacian matrix with its eigenvectors, here however, we capture it by the continuous kernel and its eigenfunctions. In fact, we interpret the graph adjacency matrix with its eigenvectors as the discrete counterpart of a continuous kernel with its eigenfunctions. By utilizing the kernel eigenfunctions together with the graph nodal distribution, we establish the continuous model and corroborate it by performing interpolation and extrapolation of graph signals.

### 1.5. Main contributions

In analogy with digital (discrete) signals and analog (continuous) signals in classical DSP, we adopt the *manifold assumption* [38–40] and view the graph nodes as samples from a continuous manifold. We assume that these nodes admit a Gaussian Mixture Model (GMM) [41] on the manifold and define a corresponding RKHS with a Gaussian kernel. If the approximation of the manifold with a GMM in the ambient space is poor, we propose to embed the manifold into a latent space. The graph signals are then described as samples of continuous functions

defined on the manifold. Under this model, closed-form expressions of the eigenfunctions of the Gaussian kernel exist (see Section 3). Based on these closed-form expressions, we propose an algorithm for graph signal interpolation and extrapolation and provide some theoretical justifications. We test our algorithm on several problems and demonstrate superior or on-par interpolation and extrapolation results compared to existing baselines. Furthermore, we show that our approach is computationally more efficient and requires shorter run times.

In this work, we focus on the cases where the geometric structure of the data is not immediately apparent, i.e., the graph adjacency matrix is not given nor is it naturally defined. If the geometric structure is readily available, it is possible to incorporate it into the kernel as additional features, either directly in the ambient space or in a latent space. These scenarios were not explored in this study.

We summarize the novelties and contributions as follows.

(C1) We propose a novel continuous model for discrete graph signals when the graph connectivity is not immediately apparent.
(C2) We propose a nodal distribution approximation based on a Gaussian mixture model on a manifold, either in ambient space or in latent space.
(C3) Based on the continuous model and the nodal distribution approximation, we propose a low computational complexity algorithm for both interpolation and extrapolation of graph signals.

We mention a related line of work that utilizes the Kriging approach [42] to extrapolate functions using various methods of Operational Kriging in a Hilbert space [43–45]. While this model resembles ours, in [45], the function to be extrapolated is modeled by a Gaussian random field in a Hilbert space, whereas in our setup, the Bayesian model is for the distribution of the nodes and the graph signal is deterministic.

### 1.6. Outline

The remainder of the paper is organized as follows. In Section 2, we introduce some preliminaries on GSP, the manifold assumption, and RKHS theory. In Section 3, we present our model that we term **R**KHS **o**ver Manifold of Gaussian **Mix**ture (RoMix), where we establish links to the Gaussian kernel and to the manifold of Gaussian mixture. In Section 4, we present the algorithm for graph signal interpolation and extrapolation based on the RoMix model. Section 5 shows the experimental results, and finally, in Section 6, we conclude our work.

## 2. Preliminaries

In this section, we present preliminaries on graphs and graph signals (Section 2.1), the manifold assumption (Section 2.2), and RKHS theory (Section 2.3).

### 2.1. Graphs and graph signals

An undirected[1] weighted graph $G = (\mathcal{V}, \mathcal{E}, \mathbf{A})$ is specified by its set of nodes (also termed "vertex set") $\mathcal{V}$ of size $|\mathcal{V}| = n$, and its edge set $\mathcal{E}$. Graph-signal is defined as a map $s : \mathcal{V} \to \mathbb{R}$, and it is commonly represented as a vector $\boldsymbol{s} \in \mathbb{R}^n$. The graph-adjacency matrix, $\mathbf{A} \in \mathbb{R}^{n\times n}$, depicts the weights on each edge and describes the pairwise relations between the graph vertices. For undirected graphs, $\mathbf{A}$ is symmetric. The graph adjacency, alongside its eigenvectors, enables numerous GSP tasks [46–49]. For instance, in [46], the eigenvectors of the adjacency matrix are promoted to define the Graph Fourier Transform (GFT) [1, 2], rather than the eigenvectors of the graph Laplacian (e.g., [1]) and in [48], graph signals are filtered by manipulating the spectrum of the graph adjacency. In many applications, the edge weights are not naturally defined. Therefore, the graph adjacency matrix $\mathbf{A}$ is generated from an application-dependent kernel function $K : \mathcal{V} \times \mathcal{V} \to \mathbb{R}$ which captures the pairwise similarities between the nodes [1]. For instance, in a social network, where each node $v_i$ represents a user, $K(v_i, v_j)$ can return the number of mutual social connections as the weight of the connected edge. In other cases, $v_i$ and $v_j$ can be associated with some feature vectors $\boldsymbol{v}_i \in \mathbb{R}^d$, and $\boldsymbol{v}_j \in \mathbb{R}^d$ while the kernel function is some known positive semi-definite expression $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$. We focus on this latter setup.

For every pair of nodes $v_i, v_j \in \mathcal{V}$ connected by the edge $(i, j) \in \mathcal{E}$, the $(i, j)$-th entry of $\mathbf{A}$ is given by:

$$\mathbf{A}_{i,j} = K(v_i, v_j). \tag{1}$$

An important interpretation of the adjacency matrix is that it can serve as a graph shift operator. This interpretation allows defining a Fourier transform for graphs via its set of eigenvalues and eigenvectors [46].

The degree matrix is defined by $\mathbf{D} = \mathrm{diag}(\mathbf{A}\mathbf{1})$, where $\mathbf{1} \in \mathbb{R}^n$ is the all-ones vector, and diag() is the operation of constructing a diagonal matrix.

One of the most important matrices in graph theory, and particularly in GSP, is the (graph) Laplacian [3]. There are many utilities for the Laplacian in the literature. Specifically, its eigenvectors and eigenvalues can also be used in the definition of the GFT [1,2], similarly to the graph adjacency's [46]. There are several commonly-used variants of the Laplacian. We focus on the normalized graph Laplacian, which is defined as follows:

$$\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}, \tag{2}$$

where $\mathbf{I}$ is the $n \times n$ identity matrix. The GFT relies on the eigendecomposition of the Laplacian, which is generally computed in $\mathcal{O}(n^3)$. Hence, the ability to calculate the GFT when $n$ is large is computationally demanding. Several approaches were suggested in order to mitigate this shortcoming. For instance, [43] suggests a fast algorithm that approximates the GFT, and [50] presents an exact fast algorithm for bipartite graphs or graphs with symmetries. In Section 5, we utilize our algorithm in order to approximate the eigenvectors of the Laplacian in an efficient manner.

### 2.2. The manifold assumption

Our model, RoMix, relies on the commonly-used *manifold assumption* [38–40]. This assumption states that real-world data in some high-dimensional ambient space $\mathcal{X}$ lie on, or close to, a low-dimensional manifold $\mathcal{M} \subset \mathcal{X}$. Suppose the ambient space $\mathcal{X}$ is equipped with a probability distribution $P_{\mathcal{X}}$, and the manifold is the support of this probability distribution. Formally,

$$\begin{aligned} &\mathcal{M} = \mathrm{supp}(P_{\mathcal{X}}) = \{x \in \mathcal{X} \mid P_{\mathcal{X}}(x) \neq 0\} \subset \mathcal{X}, \\ &P_{\mathcal{X}}(\mathcal{M}) = \int_{\mathcal{M}} p(x)dx = 1, \end{aligned} \tag{3}$$

where $p$ is the probability density, i.e., $dP_{\mathcal{X}}(x) = p(x)dx$. Naturally, the data distribution $P_{\mathcal{X}}$ is unknown and is usually modeled or estimated.

In Section 3 we present the usage of this assumption and our model for the probability distribution $P_{\mathcal{X}}$.

### 2.3. Reproducing kernel Hilbert space

A kernel function is a symmetric, positive semi-definite function defined over a set $\mathcal{X}$, denoted by $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. In our work, $\mathcal{X}$ is the ambient space in which the manifold $\mathcal{M}$ lies. An RKHS is a Hilbert space of functions over $\mathcal{X}$ which is associated with a kernel $K$. We skip the formal definition for RKHS as it is not directly used in our work and present it in Appendix A. Instead, we present an alternative definition

[1] Directed graph can generally be considered, however, will not be addressed in this work.

that relies on the fact that every RKHS has a unique kernel and vice-versa (see Appendix A). To emphasize the one-to-one correspondence between the kernel function $K$ and its RKHS, we denote the RKHS by $\mathcal{H}_K$. The alternative definition we use in our work is derived from the space $\mathcal{L}^2_p(\mathcal{X})$ and an integral operator that is associated with the kernel $K$. $\mathcal{L}^2_p(\mathcal{X})$ is the space of square integrable real functions over $\mathcal{X}$ with respect to the probability measure $P_{\mathcal{X}}$ presented in (3). The inner product in $\mathcal{L}^2_p(\mathcal{X})$ between $f, g \in \mathcal{L}^2_p(\mathcal{X})$ is given by $\langle f, g\rangle_{\mathcal{L}^2_p} = \int_{\mathcal{X}} f(x)g(x)p(x)dx$.

The integral operator $L_K : \mathcal{L}^2_p(\mathcal{X}) \to \mathcal{L}^2_p(\mathcal{X})$ is associated to the kernel $K$ and is given by:

$$(L_K f)(x) = \int_{\mathcal{X}} K(x, y)f(y)p(y)dy. \tag{4}$$

It can be shown that the eigenfunctions of (4) form an orthonormal basis for the RKHS with a discrete spectrum which consists of positive eigenvalues that decay to zero (the spectral theorem, Ch. 3 of [26]). This yields an alternative definition for the RKHS:

$$\mathcal{H}_K = \left\{ f \in \mathcal{L}^2_p(\mathcal{X}) \;\middle|\; f = \sum_{m=0}^{\infty} c_m \phi_m \text{ with } \left(\frac{c_m}{\sqrt{\lambda_m}}\right) \in \ell^2 \right\},$$

where $\{\lambda_m\}_{m=0}^{\infty}, \{\phi_m\}_{m=0}^{\infty}$ are the eigenvalues and eigenfunctions, respectively.

Therefore, for any function $f \in \mathcal{H}_K$, there exists a unique set of expansion coefficients $\{c_m\}_{m=0}^{\infty} \in \mathbb{R}$ such that:

$$f(x) = \sum_{m=0}^{\infty} c_m \phi_m(x), \quad c_m = \langle f, \phi_m\rangle_{\mathcal{L}^2_p}, \; \forall x \in \mathcal{X}. \tag{5}$$

As a Hilbert space, $\mathcal{H}_K$ is endowed with an inner product $\langle\cdot,\cdot\rangle_{\mathcal{H}_K}$. Let $f, g \in \mathcal{H}_K$ where $f = \sum_m c_m\phi_m$ and $g = \sum_m c'_m\phi_m$ with $c_m = \langle f, \phi_m\rangle_{\mathcal{L}^2_p}$ and $c'_m = \langle g, \phi_m\rangle_{\mathcal{L}^2_p}$. The inner product and the induced norm are respectively given by:

$$\langle f, g\rangle_{\mathcal{H}_K} = \sum_{m=0}^{\infty} \frac{c_m c'_m}{\lambda_m}, \tag{6}$$

$$\|f\|_{\mathcal{H}_K} = \left(\sum_{m=0}^{\infty} \frac{c_m^2}{\lambda_m}\right)^{1/2}. \tag{7}$$

Since $\{\lambda_m\}$ are all positive and decay to zero, in order for the induced norm (7) to be convergent, the function expansion coefficients must decay faster than the eigenvalues, i.e., $(c_m/\sqrt{\lambda_m}) \in \ell^2$. This constraint on the norm (7) can provide a measure of the "smoothness" of a function in the RKHS over its domain.

## 3. Graph-signals as functions in RoMix

In this section, we present our model, RoMix. Consider a graph $G = (\mathcal{V}, \mathcal{E}, \mathbf{A})$ with $n$ nodes, $\mathcal{V} = \{v_i\}_{i=1}^n$, as in Section 2.1, with the graph adjacency matrix $\mathbf{A}$ generated from a kernel function $K : \mathcal{V} \times \mathcal{V} \to \mathbb{R}$ as in (1). We assume the following:

**Assumption 3.1** ($\mathcal{V}$ *are Samples of* $\mathcal{M}$)**.** We adopt the manifold assumption (Section 2.2) and view the set of nodes $\mathcal{V}$ as samples of a continuous manifold $\mathcal{M}$ embedded in an ambient space $\mathcal{X}$, i.e., $\mathcal{V} \subset \mathcal{M} \subset \mathcal{X}$.

**Assumption 3.2** (*The Kernel Imposes an RKHS Over* $\mathcal{V}$)**.** Following Section 2.3, once a kernel is defined, it imposes a unique RKHS. If the graph adjacency matrix is constructed using a kernel function, an underlying implicit RKHS is imposed over the vertex set $\mathcal{V}$ with $K : \mathcal{V} \times \mathcal{V} \to \mathbb{R}$ being its reproducing kernel (Section 2.3). We denote this RKHS by $\mathcal{H}_K$.

Following Assumptions 3.1 and 3.2, we extend the kernel function from the discrete set of nodes, $\mathcal{V}$, to the continuous manifold of nodes, $\mathcal{M}$, i.e., $K : \mathcal{M} \times \mathcal{M} \to \mathbb{R}$. Since the graph was constructed with the kernel $K$, it imposes a unique space of continuous functions with $\mathcal{M}$ being their continuous domain. Therefore, as $\mathcal{V}$ are samples from $\mathcal{M}$, we view a discrete graph-signal $s : \mathcal{V} \to \mathbb{R}$ as $n$ samples from a continuous function $f : \mathcal{M} \to \mathbb{R}$ that lives in $\mathcal{H}_K$. Fig. 1 shows an illustration for this model.

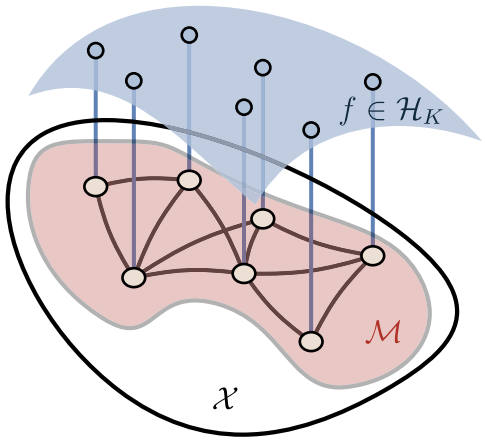


**Fig. 1.** RoMix model illustration. The graph is illustrated by nodes and edges lying in a manifold $\mathcal{M}$ that is embedded in the ambient space $\mathcal{X}$. The blue sticks illustrate the graph signal, which is a sampled version of a continuous function $f \in \mathcal{H}_K$.

Furthermore, we can express each function $f \in \mathcal{H}_K$ by the eigenbasis of the RKHS, $\{\phi_m\}_{m=0}^{\infty} \in \mathcal{H}_K$, as in (5). The graph-signal, being the discrete counterpart of $f$, can therefore be expressed in the same form, while being evaluated on the nodes, namely:

$$s(v) = \sum_{m=0}^{\infty} c_m \phi_m(v), \quad c_m = \langle f, \phi_m\rangle_{\mathcal{L}^2_p}, \quad v \in \mathcal{V}. \tag{8}$$

The expansion in (8) implies we can evaluate $f(x)$, the continuous counterpart of $s$, at any point on the manifold $x \in \mathcal{M}$, given the kernel eigenfunctions $\phi_m(x)$ and the expansion coefficients $c_m$, facilitating both interpolation and extrapolation of the graph signal $s$.

In the remainder of this section, we show how we exploit the existence of closed-form expressions for $\{\phi_m\}$ when $\mathbf{A}$ is generated by the Gaussian kernel. Later, in Section 4, we present an algorithm that finds the proper set of coefficients $\{c_m\}$. We remark that even though we hereafter focus on the Gaussian kernel, any symmetric positive semi-definite function can be employed as the kernel of the RKHS [27]. For instance, the sinc kernel, $K(x, y) = \text{sinc}(x - y)$, also has closed-form expressions for its associated eigenfunctions [51] under a uniform distribution, and can thus be utilized instead of the Gaussian kernel under the same approach.

### 3.1. The Gaussian Kernel and its associated eigenfunctions

To generate the adjacency matrix $\mathbf{A}$ we use the popular Gaussian kernel, i.e., for any $v_i, v_j \in \mathcal{V}$, (1) takes the form:

$$\mathbf{A}_{i,j} = K(v_i, v_j) = \exp\left(-\frac{\|v_i - v_j\|^2_{\mathcal{X}}}{2\omega^2}\right), \tag{9}$$

where $\omega \in (0, \infty)$ is the Gaussian width and $\|\cdot\|_{\mathcal{X}}$ is the norm in the ambient space. Taking the broader view, this is just an evaluation of the following kernel function $K : \mathcal{M} \times \mathcal{M} \to \mathbb{R}$ over two points in the manifold,

$$K(x, y) = \exp\left(-\frac{\|x - y\|^2_{\mathcal{X}}}{2\omega^2}\right), \tag{10}$$

with $x = v_i$ and $y = v_j$.

If the ambient space is $\mathcal{X} = \mathbb{R}^d$ and the data distribution $P_{\mathcal{X}}$ (3) follows a Gaussian distribution, the Gaussian kernel eigenfunctions that

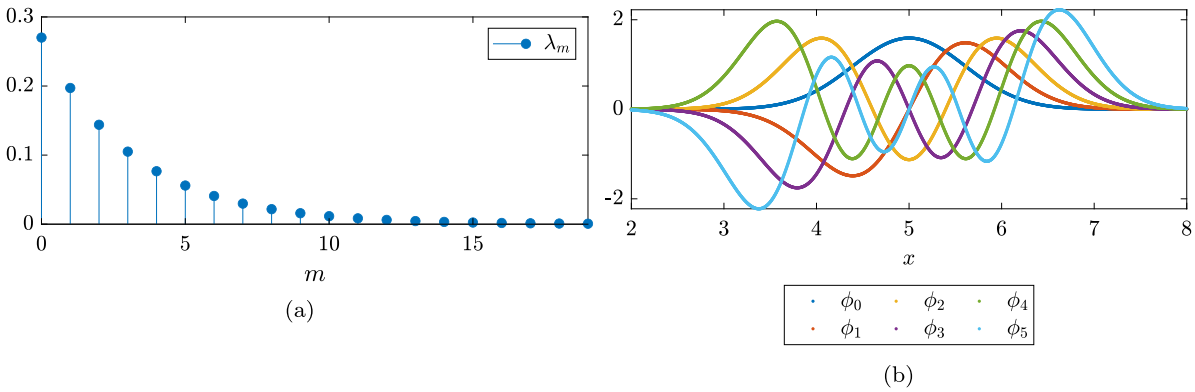


**Fig. 2.** Top eigenvalues and eigenfunctions of the Gaussian kernel integral operator on $\mathcal{X} = \mathbb{R}$ with $p = \mathcal{N}(\mu = 5, \sigma^2 = 1)$. The Gaussian kernel width is set to $\omega = 0.3$. (a) Top 20 eigenvalues (11). (b) Top 6 eigenfunctions (12).

span the associated RKHS have closed-form expressions [52,53]. We describe them here for completeness.

First, let us consider the one-dimensional case of $\mathcal{X} = \mathbb{R}$ with $p = \mathcal{N}(\mu, \sigma^2)$ being a Gaussian density with mean $\mu \in \mathbb{R}$ and variance $\sigma^2 > 0$. In this setting, the eigenvalues $\{\lambda_m\}_{m=0}^{\infty}$ and eigenfunctions $\{\phi_m\}_{m=0}^{\infty}$ are given by:

$$\lambda_m = \frac{\sqrt{2}}{(1+\beta+\sqrt{1+2\beta})^{1/2}} \left( \frac{\beta}{(1+\beta+\sqrt{1+2\beta})} \right)^m, \tag{11}$$

$$\phi_m(x) = \frac{(1+2\beta)^{1/8}}{\sqrt{2^m m!}} \exp\left( -\frac{(x-\mu)^2}{2\sigma^2} \frac{\sqrt{1+2\beta}-1}{2} \right) \times H_m\left( \left( \frac{1+2\beta}{4} \right)^{1/4} \frac{x-\mu}{\sigma} \right), \tag{12}$$

where $\beta = 2\sigma^2/\omega^2$ and $H_m(x)$ is the $m$th Hermite polynomial. For illustration, a few of the top eigenvalues and eigenfunctions are shown in Fig. 2.

Now we turn to the extension of (11) and (12) to the $d$-dimensional case, i.e, $\mathcal{X} = \mathbb{R}^d$. Let $p$ be a multivariate Gaussian, $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ with mean vector $\boldsymbol{\mu} \in \mathbb{R}^d$ and covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{d \times d}$. Let $\sum_{i=1}^{d} \sigma_i^2 \boldsymbol{u}_i \boldsymbol{u}_i^T$ be the eigen-decomposition of $\boldsymbol{\Sigma}$. The eigenvalues and eigenfunctions of the Gaussian kernel in $\mathbb{R}^d$ are given by a product of $d$ univariate elements, where each element is due to the density $\mathcal{N}(\mu_i, \sigma_i^2)$ with $\mu_i = \langle \boldsymbol{\mu}, \boldsymbol{u}_i \rangle_{\mathbb{R}^d}$, i.e.:

$$\lambda_m = \lambda_{[m_1,\dots,m_d]} = \prod_{i=1}^{d} \lambda_{m_i}, \tag{13}$$

$$\phi_m(\boldsymbol{x}) = \phi_{[m_1,\dots,m_d]}(\boldsymbol{x}) = \prod_{i=1}^{d} \phi_{m_i}(\langle \boldsymbol{x}, \boldsymbol{u}_i \rangle_{\mathbb{R}^d}), \tag{14}$$

where $[m_1, \dots, m_d]$ is a multi-index vector with $m_i = 0, 1, 2, \dots$ over all components. The multi-index vectors are sorted according to the magnitude of the eigenvalues (13). For simplicity, we use a uni-index $m = 0, 1, 2, \dots$ for the sorted multi-index vectors.

It is possible to further extend the closed-form expressions to the setting of $k$ Gaussian mixture components, i.e., $p = \sum_{i=1}^{k} \pi_i \mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$ with some $\{\pi_i\}_{i=1}^{k} \in (0, 1)$ such that $\sum_{i=1}^{k} \pi_i = 1$. As in [53], we propose to utilize the closed-form expressions of (13), (14) and perform a superposition of $k$ individual multivariate components, assuming the components are well separated. See [53] for more details.

### 3.2. Manifold of a Gaussian mixture

In order to utilize the closed-form expressions in (13), (14), $P_{\mathcal{X}}$ in (3) must follow a Gaussian mixture. Since this is not the general case, we propose two approaches.

**Approach 3.1** (*Approximate $P_{\mathcal{X}}$ as GMM in Ambient Space*)**.** When the manifold allows, we approximate $P_{\mathcal{X}}$ directly in the ambient space as a Gaussian mixture. We follow several prior works [54–56] that utilize this approximation using variants of a GMM.

**Approach 3.2** (*Approximate $P_{\mathcal{Z}}$ as GMM in Latent Space*)**.** When the approximation in the ambient space with a GMM is poor, we propose to embed the manifold to a latent space $\mathcal{Z}$. The embedding is performed such that the probability measure in the embedded space, $P_{\mathcal{Z}}$, can be well approximated using a GMM. One such embedding algorithm is Variational Autoencoder (VAE) [57] in which the latent space distribution follows a GMM by construction. This distribution allows us to exploit the closed-form expressions in the latent space.

As in various other works in the GSP domain which employ a kernel-based method for interpolation and extrapolation, (e.g., [22–24,31,34,36,37]), the underlying geometric structure of the data, i.e., the structural information, is captured by the kernel. In the proposed method, the eigenfunctions of the kernel play the same role and capture this geometry. In either Approach 3.1 or Approach 3.2, the GMM parameters (i.e., the means, the covariance matrices, and the component proportions) are estimated from the nodal information using the Expectation Maximization (EM) algorithm [41]. Then, the estimated parameters are fed into the closed-form expressions of the eigenfunctions (14). As the connectivity is captured by the kernel, it is also captured by its eigenfunctions. This can be seen algebraically from Mercer's theorem [58]:

$$K(x, y) = \sum_{m=0}^{\infty} \lambda_m \phi_m(x) \phi_m(y).$$

We further emphasize that the kernel operator $L_K$ in (4) and therefore its eigenbasis depend on the distribution parameters, and for the same kernel function $K$, different sets of eigenbasis will result from different sets of estimated parameters. An extensive analysis of the effect of the data density on kernel methods is done in [59].

By applying either of the two approaches, we obtain closed-form expressions for the eigenfunctions at any point on the manifold. Combining it with the expansion in (8) allows us to present graph signal interpolation and extrapolation in the following Section 4. For illustration, we conclude this section with two examples demonstrating the two approaches.

#### 3.2.1. Example for Approach 3.1: Modeling the swiss roll with a GMM in ambient space

We present a simple illustration of a case where the manifold allows approximating $P_{\mathcal{X}}$ in the ambient space as a Gaussian mixture. The Swiss roll is a 2D manifold, $\mathcal{M}$, embedded in $\mathcal{X} = \mathbb{R}^3$.

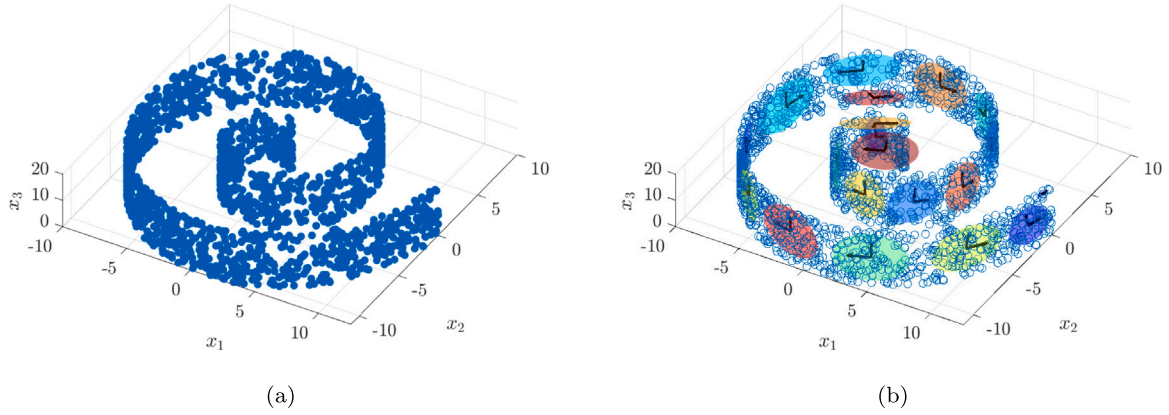


**Fig. 3.** Manifold of Gaussian mixture example: Swiss roll in ambient space. (a) $n = 1000$ points sampled from the Swiss roll dataset. (b) GMM parameters visualization. Blue hollow circles: original $n = 1000$ samples. Colored filled circles: contours of the covariance matrices corresponding to 1.5 standard deviations. Black arrows: 3 eigenvectors of each component, centered around the corresponding component mean.

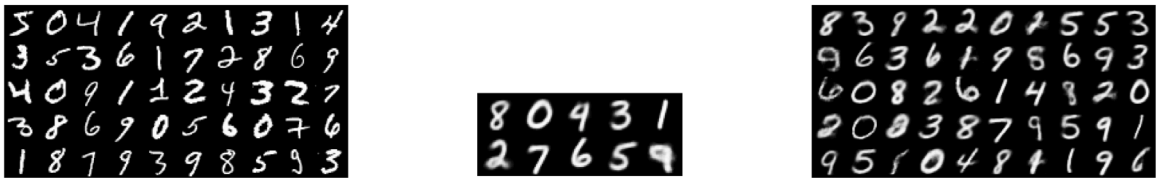


**Fig. 4.** Manifold of Gaussian mixture example: MNIST in latent space. (a) 50 random samples from the MNIST dataset. (b) GMM mean vectors in latent space after applying VAE decoding. (c) 50 random draws from obtained GMM in latent space after VAE decoding.

The Swiss roll is parameterized by $(\theta, t)$, with $h$ being a constant, such that:

$$(x_1, x_2, x_3) = (\theta \cos\theta, \theta \sin\theta, ht). \tag{15}$$

We generate $n = 1000$ samples of $(\theta, t)$ pairs such that $\theta$ is sampled uniformly on the arclength of a spiral, and $t$ is sampled uniformly in $[0, 1]$. We set $h = 20$. We then estimate a GMM using the Expectation Maximization (EM) algorithm [41] with $k = 20$ components. We plot the original dataset and the retrieved parameters in Fig. 3.

#### 3.2.2. *Example for Approach 3.2: Modeling MNIST with a GMM in a latent space*

We now present an illustration for the case where we embed data into a latent space, $\mathcal{Z}$, and estimate a GMM in the latent space. We train a VAE on the MNIST dataset [60] with a latent space dimension of $d = 20$. Fig. 4(a) shows 50 samples from the MNIST dataset. Once the VAE training is done, we randomly sample $n = 24{,}000$ images from MNIST and apply the trained VAE encoder to retrieve the corresponding representations in the latent space. Then, we estimate a GMM on the retrieved representations with $k = 10$ components. To visualize the GMM fit on the data in the latent space, we present the 10 mean vectors, one for each component, after applying the trained VAE decoder for visualization in Fig. 4(b). We observe that the obtained means assume the shapes of the digits. This implies that the GMM in the latent space retrieved reasonable parameters. After obtaining the GMM parameters, we generate 50 new points from the estimated GMM and run them through the decoder to visualize the results. Fig. 4(c) shows the generated images, which indeed resemble images from MNIST.

## 4. Graph-signals interpolation and extrapolation using RoMix

In this section, we present an algorithm for graph signal interpolation and extrapolation based on our model RoMix (Section 3). Consider a graph-signal $s_\ell$, whose values are known only on a subset of $\ell$ labeled nodes $\mathcal{V}_\ell \subset \mathcal{V}, \ell < n$. The task of graph-signal interpolation is to determine the graph-signal values on the remaining $n-\ell$ nodes, denoted $\mathcal{V}_\ell^c$, based on $s_\ell$, while the task of graph-signal extrapolation is to determine the values on any unseen node added to the graph, which was not in the original set $\mathcal{V}$.

To interpolate $s_\ell : \mathcal{V}_\ell \to \mathbb{R}$ to $s : \mathcal{V} \to \mathbb{R}$ and extrapolate $s : \mathcal{V} \to \mathbb{R}$ to $f : \mathcal{M} \to \mathbb{R}$, we model $f$ as a function in the RKHS as in (5) and require the following: First, we require consistency by minimizing the mean squared error between $f(v_i)$ and $s_\ell(v_i)$. Second, we impose regularity by penalizing the RKHS norm (7) of $f$ over the manifold, with some parameter $\gamma > 0$. Penalizing high norm values imposes a "smoother" function in the RKHS over its domain (see Section 2.3).

The two requirements lead to the following optimization problem:

$$f = \arg\min_{\tilde{f} \in \mathcal{H}_K} \sum_{i=1}^{\ell} \left(\tilde{f}(v_i) - s_\ell(v_i)\right)^2 + \gamma \|\tilde{f}\|_{\mathcal{H}_K}^2. \tag{16}$$

### 4.1. *From infinite to finite set of coefficients*

Finding $f \in \mathcal{H}_K$ is equivalent to finding the (possibly infinite number of) representation coefficients $\{c_m\}$ in the expansion (5). However, finding infinitely many coefficients is unnecessary. By penalizing high norm values in (16), the higher order eigenfunctions are de-emphasized in the expansion (Section 2.3). Therefore, rather than taking infinitely many terms, we approximate $f$ by the $M$ principal eigenfunctions corresponding to the largest $M$ eigenvalues of the kernel operator, i.e.:

$$f(x) = \sum_{m=0}^{\infty} c_m \phi_m(x) \approx f_M(x) = \sum_{m=0}^{M-1} c_m \phi_m(x). \tag{17}$$

The selection of $M$ can be performed via cross-validation. This restriction to an $M$ dimensional subspace can be interpreted in several ways. First, from the GSP perspective, it is possible to relate the eigenvalues of the graph adjacency matrix to the notion of frequencies [46]. This notion remains valid while extending the set $\mathcal{V}$ to the entire manifold $\mathcal{M}$; as the index $m$ increases, the eigenfunctions of the

kernel operator become less "smooth" over the manifold. Therefore, from a frequency-domain perspective, setting the coefficients $c_m$ for $m \geq M$ to zero can be considered as low-pass filtering that imposes a smooth function over the manifold. Another interpretation is that by restricting the eigen-expansion to depend only on a small subset of eigenfunctions, we follow Occam's Razor principle, roughly stating that "the simplest solution is most likely the right one". In fact, by restricting the expansion only to depend on the top $M$ eigenfunctions, we look for a simpler solution that does not scale up with the number of data points, and we solve the problem in a well-posed fashion. Following (17), the RKHS norm (7) can also be approximated using the first $M$ terms:

$$\|f\|_{\mathcal{H}_K}^2 = \sum_{m=0}^{\infty} \frac{c_m^2}{\lambda_m} \approx \sum_{m=0}^{M-1} \frac{c_m^2}{\lambda_m} = \boldsymbol{c}^T \boldsymbol{\Lambda}_M^{-1} \boldsymbol{c}, \tag{18}$$

where $\boldsymbol{c}_m = c_m$ and $\boldsymbol{\Lambda}_M \in \mathbb{R}^{M\times M}$ is a diagonal matrix with $(\boldsymbol{\Lambda}_M)_{m,m} = \lambda_m$.

In Section 3.2, we proposed two approaches. In the first (Approach 3.1), we approximate $P_{\mathcal{X}}$ with a GMM in the ambient space $\mathcal{X} = \mathbb{R}^d$. In the second (Approach 3.2), we embed the manifold in a latent space $\mathcal{Z} = \mathbb{R}^d$ and then approximate $P_{\mathcal{Z}}$ with a GMM in the latent space. Either way, we consider the embedding of the nodes in $\mathbb{R}^d$. Let the vector $\boldsymbol{v}_i \in \mathbb{R}^d$ represent the embedding of node $v_i$. We define the nodes embedding coordinates matrix $\mathbf{V}_\ell = [\boldsymbol{v}_1, \dots, \boldsymbol{v}_\ell]^T \in \mathbb{R}^{\ell\times d}$, the graph signal represented by a vector $\boldsymbol{s}_\ell = [s(\boldsymbol{v}_1), \dots, s(\boldsymbol{v}_\ell)]^T \in \mathbb{R}^\ell$, and the eigenfunctions matrix evaluated at $\mathbf{V}_\ell$ by $\boldsymbol{\Phi}_M(\mathbf{V}_\ell) = [\boldsymbol{\phi}_0(\mathbf{V}_\ell), \dots, \boldsymbol{\phi}_{M-1}(\mathbf{V}_\ell)] \in \mathbb{R}^{\ell\times M}$ with $\boldsymbol{\phi}_m(\mathbf{V}_\ell) = [\phi_m(\boldsymbol{v}_1), \dots, \phi_m(\boldsymbol{v}_\ell)]^T \in \mathbb{R}^\ell$ following (14) for $m \in \{0, \dots, M-1\}$. Substituting (17) and (18) into (16) leads to the following finite dimensional optimization problem:

$$\boldsymbol{c} = \arg\min_{\tilde{\boldsymbol{c}}\in\mathbb{R}^M} \|\boldsymbol{\Phi}_M \tilde{\boldsymbol{c}} - \boldsymbol{s}_\ell\|_{\mathbb{R}^M}^2 + \gamma \tilde{\boldsymbol{c}}^T \boldsymbol{\Lambda}_M^{-1} \tilde{\boldsymbol{c}}, \tag{19}$$

whose solution is given by

$$\boldsymbol{c} = \left(\boldsymbol{\Phi}_M^T \boldsymbol{\Phi}_M + \gamma \boldsymbol{\Lambda}_M^{-1}\right)^{-1} \boldsymbol{\Phi}_M^T \boldsymbol{s}_\ell. \tag{20}$$

Note that $\{\lambda_m\}_{m=0}^{M-1}$ are strictly positive and bounded away from zero. Therefore the diagonal matrix $\boldsymbol{\Lambda}_M$ is invertible.

### 4.2. The proposed interpolation and extrapolation algorithm

Following the above, we now present the interpolation and extrapolation algorithm using RoMix in Algorithm 1.

### 4.3. Parameters selection

Algorithm 1 has four parameters: $k$, the number of GMM components, $\omega$, the Gaussian kernel width, $M$, the number of eigenfunctions, and $\gamma$, the RKHS norm penalty term. The four parameters can be determined by cross-validation, as they are application-specific. However, there exist methods to select some of the parameters analytically. The number of GMM components can be selected by minimizing the Akaike information criterion (AIC) or the Bayesian information criterion (BIC) [41]. The selection of the Gaussian kernel width parameter, $\omega$, is greatly studied in the literature. For instance, in [61], it is selected according to $\varepsilon = 2\omega^2$, which is determined by the expression

$$\varepsilon = \frac{1}{n} \sum_{i=1}^{n} \min_{j:\boldsymbol{v}_i \neq \boldsymbol{v}_j} \|\boldsymbol{v}_i - \boldsymbol{v}_j\|_{\mathbb{R}^d}^2. \tag{21}$$

Another approach is using

$$\varepsilon = \max_{i=1,\dots,n} \min_{j:\boldsymbol{v}_i \neq \boldsymbol{v}_j} \|\boldsymbol{v}_i - \boldsymbol{v}_j\|_{\mathbb{R}^d}^2. \tag{22}$$

With regards to the number of eigenfunctions $M$, we note that [9] obtains the number of eigenvectors of the graph Laplacian in an analytical procedure. In order to obtain a similar analytical procedure for selecting $M$ in our context, further work is required, which is beyond the scope of this paper.

**Algorithm 1** Interpolation and Extrapolation Using RoMix

**Parameters:**

$k$ : Number of GMM components.
$M$ : Number of eigenfunctions.
$\omega$ : Gaussian kernel width.
$\gamma$ : RKHS norm penalty term.

**Input:**

$\mathbf{V} \in \mathbb{R}^{n\times d}$ : Given nodes embedding matrix.
$\boldsymbol{s}_\ell \in \mathbb{R}^\ell$ : Graph signal on $\mathcal{V}_\ell \subset \mathcal{V}$.

**Output:**

$f_M : \mathcal{M} \to \mathbb{R}$ : The interpolating and extrapolating function.

**Procedure:**

1: Model the data distribution $P_{\mathcal{X}}$ (or $P_{\mathcal{Z}}$) supported by the manifold $\mathcal{M} \subset \mathcal{X}$ (or $\mathcal{Z}$) as a GMM (using the Expectation Maximization algorithm [41]) given the nodes coordinates matrix $\mathbf{V}$:

$$[\{\boldsymbol{\mu}_i\}_{i=1}^k, \{\boldsymbol{\Sigma}_i\}_{i=1}^k, \boldsymbol{\pi}] = \text{GMM}(\mathbf{V}, k)$$

where $\boldsymbol{\mu}_i \in \mathbb{R}^d, \boldsymbol{\Sigma}_i \in \mathbb{R}^{d\times d}, \boldsymbol{\pi} \in \mathbb{R}^k$ are the means vectors, the covariance matrices and the proportions of the components, respectively.

2: With the GMM outputs $[\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}]$, calculate the matrix $\boldsymbol{\Phi}_M$ containing the first $M$ analytic eigenfunctions (14) of the kernel operator (4) at the nodes $\mathbf{V}_\ell$:

$$\boldsymbol{\Phi}_M(\mathbf{V}_\ell) = \begin{bmatrix} | & & | \\ \boldsymbol{\phi}_0(\mathbf{V}_\ell) & \cdots & \boldsymbol{\phi}_{M-1}(\mathbf{V}_\ell) \\ | & & | \end{bmatrix} \in \mathbb{R}^{\ell\times M}$$

3: Find $\boldsymbol{c}$ using eq. (20).

4: For every missing graph signal value at node $x \in \mathcal{V}$ and for any new node $x \in \mathcal{M}$:

$$f_M(x) = \sum_{m=0}^{M-1} c_m \phi_m(x)$$

### 4.4. Computational complexity analysis

We now turn to analyze the computational complexity of Algorithm 1. In the case of Approach 3.1, there is no need to perform a pre-processing embedding step of the $n$ points. In the case of Approach 3.2 we apply a pre-processing step and embed the $n$ points into a latent space before running Algorithm 1. Since this optional pre-processing step depends on the application at hand, its computational complexity varies. For instance, we analyze a simple implementation of VAE as the pre-processing step for the case of images of size $d \times d$ in Appendix B. In that case, the computational complexity of training a VAE before applying Algorithm 1 can be bounded by $\mathcal{O}(tbd^2)$ where $t$ is the number of training epochs and $b$ is the batch size. We note that in some cases a pre-trained VAE might exist or can be easily obtained by fine-tuning an already-trained VAE from a similar domain. Hence, the computational complexity of training a VAE might be avoided.

In the first step of Algorithm 1, we compute the GMM parameters using the EM algorithm [41]. In each iteration, the E step costs $\mathcal{O}(nkd + nk)$ and the M step costs $\mathcal{O}(nkd)$ (see, e.g., [62]). In the typical case where the number of EM iterations is much smaller than the order of $n$, we have a total computational complexity $\mathcal{O}(nkd)$. In the second step, we evaluate the $M$ eigenfunctions at the $\ell$ nodes and therefore this step has a computational complexity of $\mathcal{O}(M\ell)$. In the third step, we compute the coefficients (20) and invert an $M \times M$ matrix which is generally $\mathcal{O}(M^3)$. In the last step we compute the $M$ eigenfunctions in $\mathcal{O}(M)$. Summing all steps results in the computational complexity for interpolation:

$$T = \mathcal{O}(nkd + M\ell + M^3 + M) = \mathcal{O}(nkd + M\ell + M^3).$$

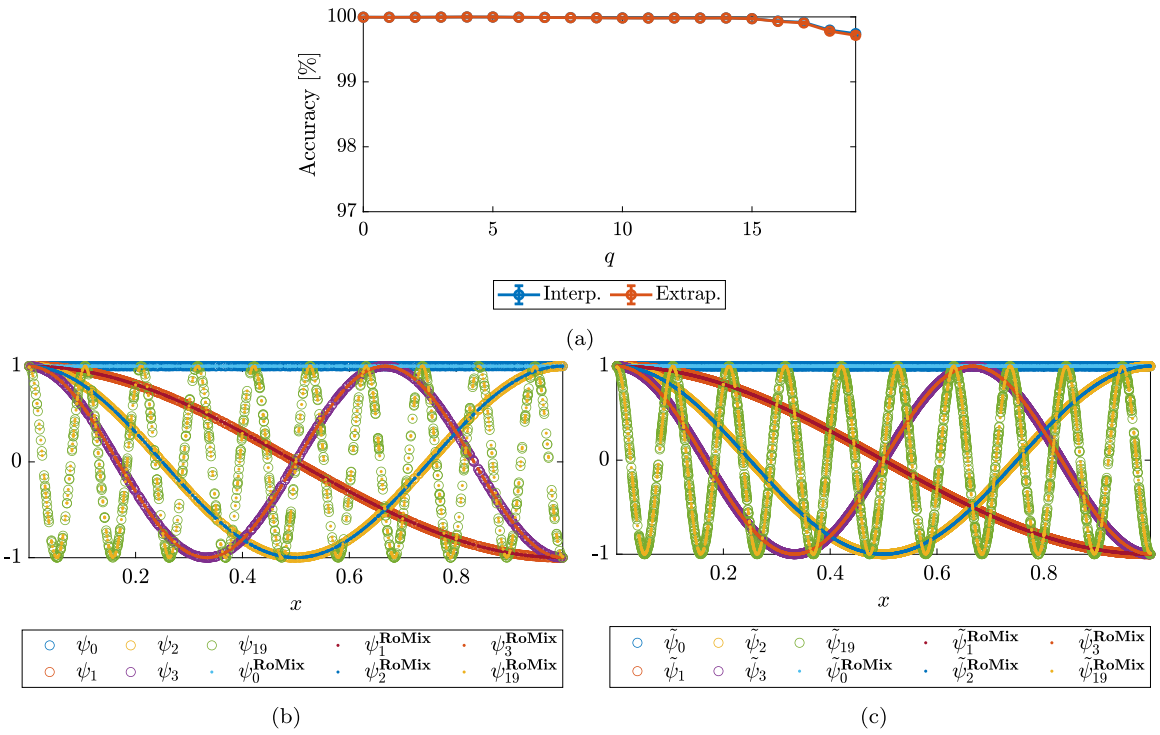


**Fig. 5.** Toy example 1: Laplacian eigenfunctions interpolation and extrapolation results with a uniform sample of the 1D grid. (a) Accuracy (23). (b) Eigenfunctions $q \in \{0, 1, 2, 3, Q-1\}$ interpolation. (c) Eigenfunctions $q \in \{0, 1, 2, 3, Q-1\}$ extrapolation.

Typically, the number of GMM components is selected such that $k \ll n$. For the case of $d \ll n$, we have

$$T = \mathcal{O}(n + M\ell + M^3).$$

The last step in Algorithm 1 is used both for interpolation and extrapolation and its computational complexity is $\mathcal{O}(M)$. Therefore, the total computational complexity is

$$T = \mathcal{O}(n + M\ell + M^3).$$

## 5. Experimental results

In this section, we showcase the performance of the RoMix algorithm (Algorithm 1) in several experiments and compare it to other baseline algorithms. In each experiment, we are given a vertex set of $n$ nodes, $\mathcal{V}$, and a graph-signal $\boldsymbol{s}_\ell \in \mathbb{R}^\ell$ with given values over $\ell < n$ nodes. We attempt to interpolate from $\boldsymbol{s}_\ell$ to the ground-truth graph-signal, $\boldsymbol{s} \in \mathbb{R}^n$, which is defined over the set $\mathcal{V}$, and further extrapolate to $\tilde{\boldsymbol{s}} \in \mathbb{R}^N$, the ground-truth graph-signal defined over an extended set of $N > n$ nodes. The set of $N - n$ nodes, alongside their nodal information, edge weights, and graph signal values are not revealed to any of the algorithms and are used only for evaluation of the extrapolation accuracy. The origin of the ground-truth graph signals is described in each of the following experiments.

Each experiment is composed of $R$ Monte-Carlo iterations. In each iteration $r \in \{1, \ldots, R\}$, we are given a different set $\mathcal{V}_\ell$ from which Algorithm 1 and the competing algorithms attempt to interpolate and extrapolate. In each iteration $r$ and for each algorithm **algo**, we measure the accuracy between the ground truth graph-signal $\boldsymbol{s}$ and the specific algorithm resultant graph-signal, $\boldsymbol{s}_r^{\text{algo}}$. The accuracy metric is then defined by:

$$\text{Acc}(\boldsymbol{s}^{\text{algo}}, \boldsymbol{s}) = 100\left(1 - \frac{1}{R}\sum_{r=1}^{R}\frac{\left\|\boldsymbol{s}_r^{\text{algo}} - \boldsymbol{s}\right\|}{\left\|\boldsymbol{s}\right\|}\right), \tag{23}$$

where the norm $\|\cdot\|$ in (23) is the Euclidean norm ($\mathbb{R}^n$ or $\mathbb{R}^N$, as needed).

We refer to the published code [63] to reproduce the results presented in this section.

### 5.1. Toy examples: Laplacian eigenfunctions

In this section, we utilize Algorithm 1 in order to approximate the eigenvectors of the normalized Laplacian (2) in $\mathcal{X} = \mathbb{R}^d$ on the $d$-dimensional hyperrectangle being the manifold $\mathcal{M} = [0, L_1] \times \cdots \times [0, L_d]$. In this case, the eigenfunctions and eigenvalues of the continuous Laplace–Beltrami operator on the manifold $\mathcal{M}$ have closed-form expressions of the form [64]:

$$\lambda_q = \lambda_{[q_1,\ldots,q_d]} = \sum_{i=1}^{d}\left(\frac{q_i\pi}{L_i}\right)^2, \tag{24}$$

$$\psi_q(\boldsymbol{x}) = \psi_{[q_1,\ldots,q_d]}(x_1, \ldots, x_d) = \prod_{i=1}^{d}\cos\left(\frac{q_i\pi x_i}{L_i}\right), \tag{25}$$

where $[q_1, \ldots, q_d]$ is a multi-index vector with $q_i = 0, 1, 2, \ldots$ over all components. The multi-index vectors are sorted according to the magnitude of the eigenvalues (24). For simplicity, we use a uni-index $q = 0, 1, 2, \ldots$ for the sorted multi-index vectors.

When the normalized graph Laplacian (2) is constructed using an adjacency matrix generated from the Gaussian kernel (10), the graph Laplacian eigenvectors and eigenvalues converge to (24) and (25), respectively, as $n \longrightarrow \infty$ and $\omega \longrightarrow 0$ (e.g. [40] and reference therein). This allows us to verify our algorithm in a controlled fashion by comparing the resulting estimations of Algorithm 1 with the closed-form expressions in (25). We note that the general task of learning the graph Laplacian from data is covered in the literature (e.g. [12,13]). In the general case, the entire graph structure, i.e., the graph Laplacian or its eigendecomposition, is learned from data. However, in our context, we recast a particular case of graph Laplacian eigenvectors estimation as graph signal interpolation and extrapolation tasks to validate Algorithm 1 results against ground-truth expressions.

In each example, we run the following $R = 10$ Monte-Carlo trials: we uniformly sample $n$ points from $\mathcal{M}$, which form the set $\mathcal{V}$, and randomly

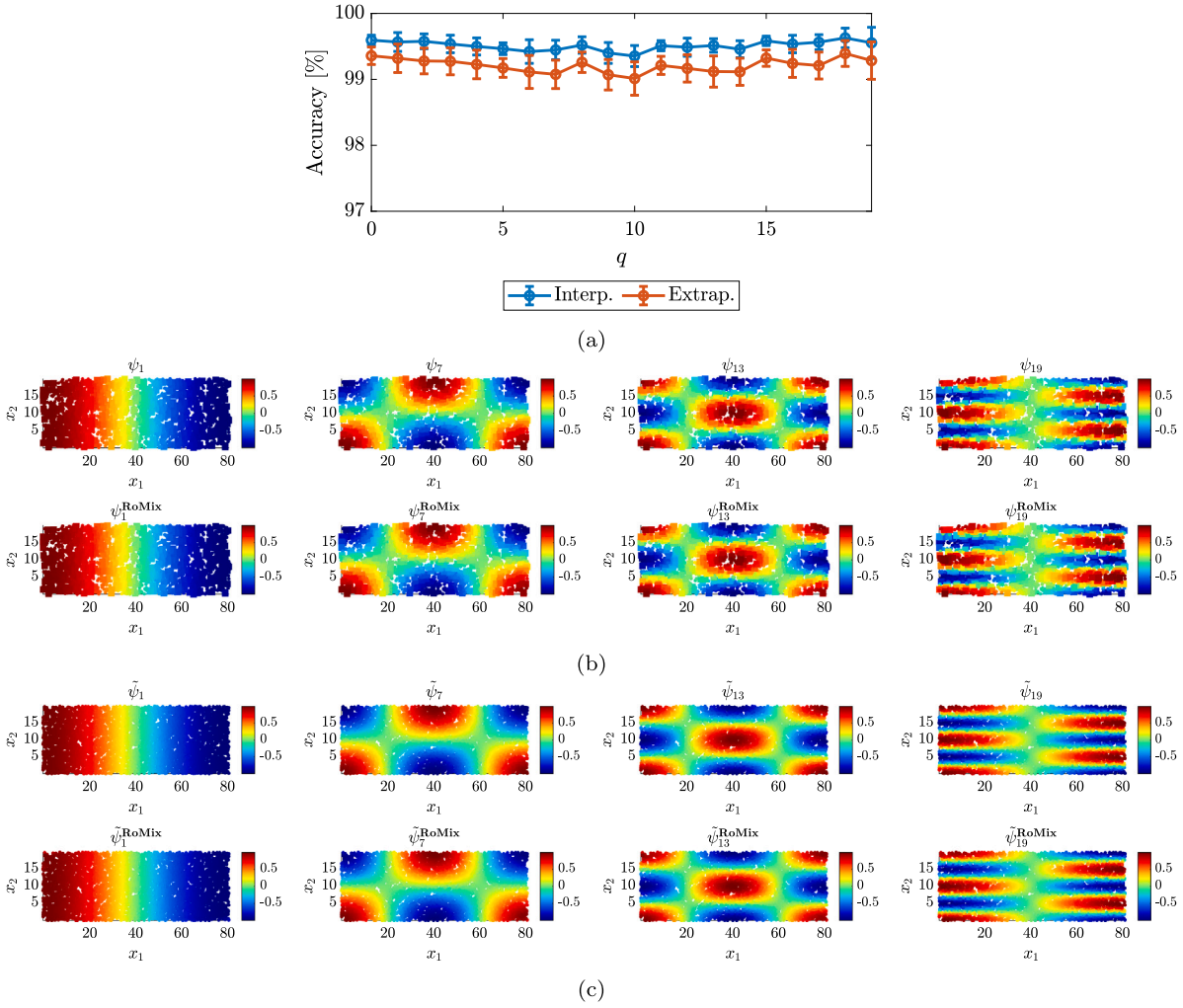


**Fig. 6.** Toy example 2: Laplacian eigenfunctions interpolation and extrapolation results with a uniform sample of the 2D grid. (a) Accuracy (23). (b) Eigenfunctions $q \in \{1, 7, 13, Q-1\}$ interpolation. (c) Eigenfunctions $q \in \{1, 7, 13, Q-1\}$ extrapolation.

sample $\ell < n$ points from $\mathcal{V}$ which form the set $\mathcal{V}_\ell$. We then evaluate the top $Q$ Laplacian eigenfunctions of the Laplacian, in $\mathcal{V}_\ell$, using (25) as the graph signals we wish to interpolate and extrapolate.

For $q \in \{0, \dots, Q-1\}$, we denote the $q$th ground-truth interpolated graph-signal (eigenfunction) by $\boldsymbol{\psi}_q \in \mathbb{R}^n$ and the ground-truth extrapolated graph-signal (eigenfunction) by $\tilde{\boldsymbol{\psi}}_q \in \mathbb{R}^N$.

The parameters for Algorithm 1 in each example are presented in Table 1. We remark that in all simulations, we chose the number of Gaussian kernel eigenfunctions, $M$, in Algorithm 1 to be $M > Q$. In fact, if $Q$ increases up to $n$, it is always possible to respectively increase $M$ due to the infinite number of the Gaussian kernel eigenfunctions, which have closed-form expressions.

### 5.1.1. Example 1: Uniform samples over a 1D grid

In this example we set $\mathcal{X} = \mathbb{R}$ and $\mathcal{M} = [0, L]$ where $L = 1$. The number of points is set to $\ell = 500$, $n = 1000$ and $N = 3000$, i.e., we have $n - \ell = 500$ missing values, and $N - n = 2000$ newly introduced nodes for the extrapolation. In this case, the eigenfunctions in (25) have the form:

$$\psi_q(x) = \cos(q\pi x). \tag{26}$$

We present the interpolation and extrapolation for eigenfunctions $q \in \{0, 1, 2, 3, Q-1\}$ of the normalized Laplacian in Figs. 5(b) and 5(c), respectively. In Fig. 5(a) we present the accuracy (23) results. Note that the obtained accuracy always exceeds $99.7 \pm 0.05\%$ for all $q = 0, \dots, Q-1$.

### 5.1.2. Example 2: Uniform samples over a 2D grid

In this example, we set $\mathcal{X} = \mathbb{R}^2$ and $\mathcal{M} = [0, L_1] \times [0, L_2]$. The values for $L_1$ and $L_2$ are chosen such that $\mathcal{M}$ is later transformed to a Swiss roll (see (15)) in the following Example 3. Therefore, we select $L_1 = S(4\pi) \approx 80$ where $S(\theta)$ is the arclength given in the following Eq. (28) and $L_2 = 20$. The number of points is set to $\ell = 1500$, $n = 2000$ and $N = 5000$, i.e., we have $n - \ell = 500$ missing values, and $N - n = 3000$ newly introduced nodes for extrapolation. In this case, the eigenfunctions in (25) are given by:

$$\psi_q(\boldsymbol{x}) = \psi_{[q_1,q_2]}(x_1, x_2) = \cos\left(\frac{q_1 \pi x_1}{L_1}\right) \cos\left(\frac{q_2 \pi x_2}{L_2}\right). \tag{27}$$

Similarly to Example 1, we present the results in Fig. 6, demonstrating the high accuracy obtained by our algorithm in the two-dimensional case as well.

### 5.1.3. Example 3: Uniform samples over a Swiss roll

In this example, we demonstrate our algorithm on the Swiss roll (see (15)). Same as in Example 2, we have two uniformly distributed random variables, $(t, \theta)$, where $\theta \in [0, 4\pi]$ is uniformly sampled along the arclength $S(\theta) \in [0, L_1]$ and $t$ is uniformly sampled in $[0, L_2]$. The arclength is given by

$$S(\theta) = \frac{1}{2}\left(\theta\sqrt{1+\theta^2} + \log\left(\theta + \sqrt{1+\theta^2}\right)\right). \tag{28}$$

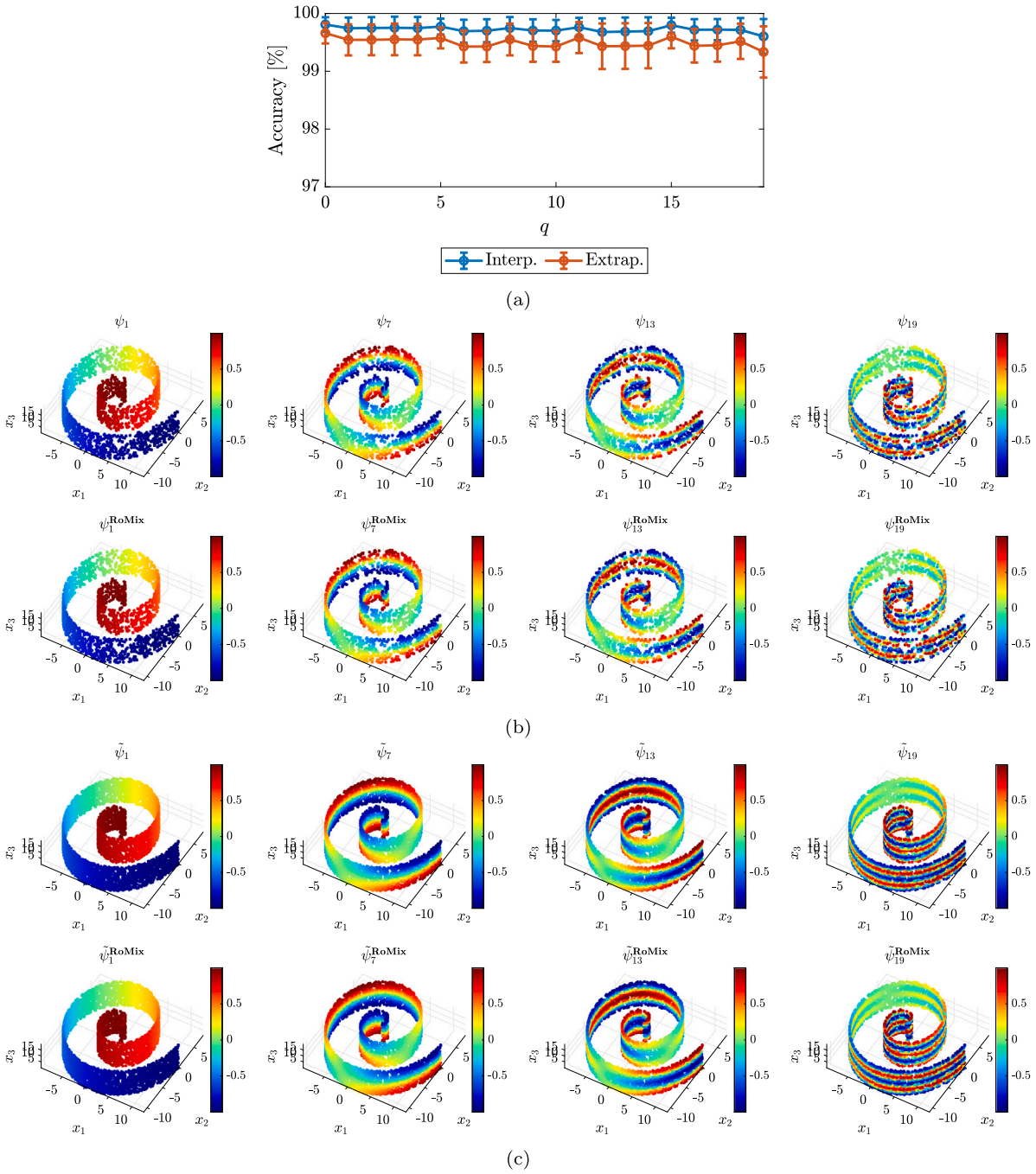


**Fig. 7.** Toy example 3: Laplacian eigenfunctions interpolation and extrapolation results with a uniform sample of the 3D Swiss roll. (a) Accuracy (23). (b) Eigenfunctions $q \in \{1, 7, 13, Q-1\}$ interpolation. (c) Eigenfunctions $q \in \{1, 7, 13, Q-1\}$ extrapolation.

**Table 1**
Toy examples Algorithm 1 parameters.

| Parameter | 1D Uniform | 2D Uniform | 3D Swiss roll |
|---|---|---|---|
| $\omega$, Kernel width | $2\sqrt{L/n}$ | $\sqrt{5/2}$ | $\sqrt{5/2}$ |
| $k$, # GMM comp. | 1 | 20 | 20 |
| $M$, # eigs. | 50 | 50 | 50 |
| $\gamma$, $\|\cdot\|_{\mathcal{H}_K}$ penalty | $1e^{-5}$ | $1e^{-5}$ | $1e^{-5}$ |

We present the results in Fig. 7 and observe similar trends as in Example 2. We note that we use the same number of points $n, \ell, N$ and the same parameters in Algorithm 1 as in Example 2.

### 5.2. Toy example: Two-Moons semi-supervised classification

In this section, we utilize Algorithm 1 for a semi-supervised classification task of the Two-Moons dataset. The Two-Moons dataset is a 2D manifold embedded in $\mathbb{R}^2$ denoted by $\mathcal{M}$. The parameterization of the upper and lower moons is given, respectively, by

$$\begin{aligned} \boldsymbol{x} &= (x_1, x_2) = (\cos(\pi t),\ \sin(\pi t)) + \boldsymbol{\eta_1}, \\ \boldsymbol{x} &= (x_1, x_2) = (1 - \cos(\pi t),\ -\sin(\pi t)) + \boldsymbol{\eta_2}, \end{aligned} \tag{29}$$

where $t$ is a parameter in the range $[0, 1]$ and $\boldsymbol{\eta_i}$ are i.i.d according to $\mathcal{N}(\mathbf{0}, \sigma_\eta^2 \boldsymbol{I})$. We generate $n = 200$ samples from (29), divided equally between the two moons and present them in Fig. 8(a). We then run the first step of Algorithm 1, i.e., the GMM approximation step, with $k = 8$ components and present the result in Fig. 8(b).

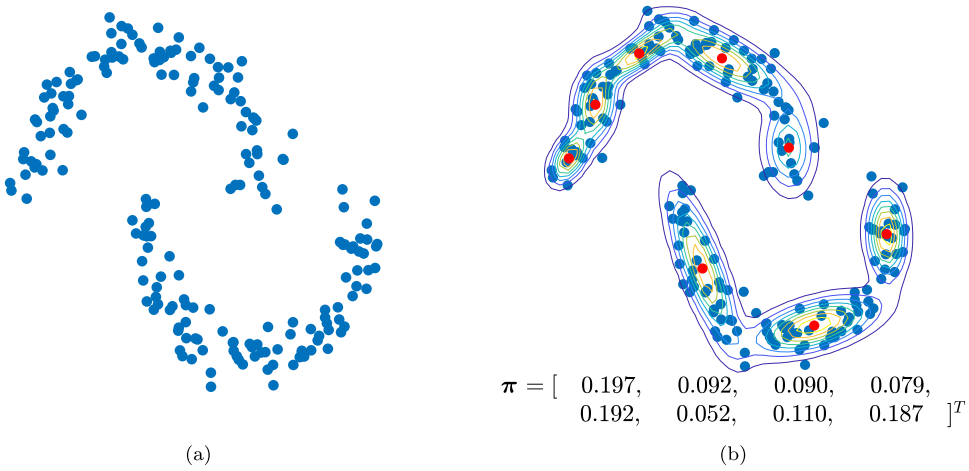


**Fig. 8.** Two-Moons manifold of Gaussian mixture. (a) $n = 200$ points of the Two Moons dataset (29). (b) GMM probability density function contour lines with $k = 8$ components. Blue circles: original $n = 200$ points. Red dots: mean vectors. Colored circles: contours of the probability density function. $\boldsymbol{\pi}$ denotes the proportions of each component in the GMM.

The semi-supervised classification task aims to classify a set of $n$ points given a small subset of $\ell \ll n$ labeled points. In this Two-Moons dataset, there are two labels; all points in the upper moon are given the label $y = +1$, and all points in the lower moon are given the label $y = -1$. This classification task can be recast as a graph signal interpolation problem by defining the set of $n$ points as the vertex set $\mathcal{V}$ with $\boldsymbol{v_i} = \boldsymbol{x_i}$ and the set of labeled points as the set $\mathcal{V}_\ell$. The graph signal can then be used as a classifier between the two classes by taking its sign, i.e., $y_i = \text{sign}\big(s(\boldsymbol{v}_i)\big)$. Points not part of the original dataset should also be classified correctly. This classification can be recast as a graph signal extrapolation problem.

A common regularizer applied to various GSP tasks is the manifold regularization [38], given by $\boldsymbol{s}^T\mathbf{L}\boldsymbol{s} = \boldsymbol{c}^T\boldsymbol{\Phi}_M^T\mathbf{L}\boldsymbol{\Phi}_M\boldsymbol{c}$. We apply this regularizer to (19) which yields

$$\boldsymbol{c} = \arg\min_{\tilde{\boldsymbol{c}}\in\mathbb{R}^M} \|\boldsymbol{\Phi}_M\tilde{\boldsymbol{c}} - \boldsymbol{s}_\ell\|^2_{\mathbb{R}^M} + \gamma_A\tilde{\boldsymbol{c}}^T\boldsymbol{\Lambda}_M^{-1}\tilde{\boldsymbol{c}} + \gamma_I\tilde{\boldsymbol{c}}^T\boldsymbol{\Phi}_M^T\mathbf{L}\boldsymbol{\Phi}_M\tilde{\boldsymbol{c}}, \tag{30}$$

where $\gamma_A$ is the ambient penalty term and $\gamma_I$ is the intrinsic penalty term. A discussion on these penalty terms can be found in [38]. The solution for (30) is given by

$$\boldsymbol{c} = \left(\boldsymbol{\Phi}_M^T\boldsymbol{\Phi}_M + \gamma_A\boldsymbol{\Lambda}_M^{-1} + \gamma_I\boldsymbol{\Phi}_M^T\mathbf{L}\boldsymbol{\Phi}_M\right)^{-1}\boldsymbol{\Phi}_M^T\boldsymbol{s}_\ell. \tag{31}$$

We note that adding the additional manifold regularization term does not affect the computational complexity of Algorithm 1 as the matrix to be inverted is still $M \times M$.

In this example, we run the following $R = 10$ Monte-Carlo trials: we randomly sample $n = 200$ points from $\mathcal{M}$ given by (29) with $\sigma_\eta = 0.1$, which forms the set $\mathcal{V}$. We randomly sample only $\ell = 2$ points from $\mathcal{V}$, one from each class, which forms the set $\mathcal{V}_\ell$. Furthermore, we sample a test set of $N - n = 200$ points to evaluate the extrapolation performance. We show the resultant graph signal in Fig. 9(a) as a continuous function over $\mathcal{M}$, and the classifier with its decision boundary in Fig. 9(b). We get 100% classification accuracy in all trials for both the interpolation and extrapolation results.

### 5.3. Real world examples

In this section, we demonstrate Algorithm 1 on two real-world datasets and compare the performance to four baseline methods described in the following Section 5.3.1 and summarized in Table 2.

#### 5.3.1. Baseline methods

The first baseline method is the Representer Theorem (abbreviated Rep. Thm.) [29] applied in an RKHS. The representer theorem is used for graph signal interpolation, for instance, in [22,23]. In this method, the interpolating and extrapolating functions are given, respectively, by

$$\boldsymbol{s}^{\text{Rep.Thm.}} = \mathbf{K}\boldsymbol{\alpha} = \mathbf{K}(\mathbf{J}\mathbf{K} + \gamma\mathbf{I})^{-1}\boldsymbol{s}_\ell, \tag{32}$$

$$\tilde{s}^{\text{Rep.Thm.}}(x) = \sum_{i=1}^{n}\alpha_i K(\upsilon_i, x). \tag{33}$$

where $\bar{\boldsymbol{s}}_\ell = [\boldsymbol{s}_\ell^T, \boldsymbol{0}_{n-\ell}^T]^T$, $\mathbf{J} = \text{diag}(\mathbf{1}_\ell, \mathbf{0}_{n-\ell})$ with $\mathbf{0}_k$ being a $k$-zeros vector, $\mathbf{1}_k$ being a $k$-ones vector, $\mathbf{K}_{i,j} = K(\upsilon_i, \upsilon_j)$ and $\gamma > 0$ is some penalty term. The computational complexity of the interpolation is $\mathcal{O}(n^3)$ due to the $n \times n$ matrix inversion. The computational complexity of the extrapolation is $\mathcal{O}(n)$ due to the $n$ evaluations $\{K(\upsilon_i, x)\}_{i=1}^n$. We remark that the representer theorem can be related to our expansion (8) using Mercer's Theorem [58]. However, our expansion does not scale up with the number of data points $n$.

The second baseline method is Variational Splines interpolation in Paley-Wiener space (abbreviated VSPW) [21]. We use the MATLAB function `gsp_interpolate()` provided in the toolbox GSPBox [65]. It implements an interpolation method based on Pesenson's model [21] and described in [20]. In this method, the graph signal is assumed to be smooth over the graph $G$ (i.e., it is a function in a Paley-Wiener space of the graph $G$ with some frequency $\omega$), and the interpolating graph signal, referred to as a variational spline, is given by

$$\boldsymbol{s}^{\text{VSPW}} = \boldsymbol{\Phi}_{n\times\ell}\boldsymbol{\alpha} = \boldsymbol{\Phi}_{n\times\ell}\boldsymbol{\Phi}_{\ell\times\ell}^{-1}\boldsymbol{s}_\ell, \tag{34}$$

where $\boldsymbol{\Phi}_{n\times\ell} \in \mathbb{R}^{n\times\ell}$ are Green's functions of the regularized Laplacian, $\bar{\mathbf{L}} = \mathbf{L} + \varepsilon\mathbf{I}$ with some $\varepsilon > 0$, and $\boldsymbol{\Phi}_{\ell\times\ell} \in \mathbb{R}^{\ell\times\ell}$ contains only the rows from $\boldsymbol{\Phi}_{n\times\ell}$ which correspond to $\mathcal{V}_\ell$. The computational complexity of the interpolation is $\mathcal{O}(n^3)$ due to the inevitable eigendecomposition of the normalized Laplacian while computing $\boldsymbol{\Phi}_{n\times\ell}$. We remark that this method does not support an extrapolation procedure and therefore needs to be run from scratch if a new node is added to the graph. As an aside, rather than taking Green's functions in (34), [9] takes the regularized Laplacian eigenvectors associated with its lowest eigenvalues.

The third baseline method is the Nyström method (abbreviated Nys) [19], which is used, for instance, in [16]. The Nyström method allows to numerically approximate the eigenfunctions of a kernel integral operator (4) [67]. In particular, it allows to extrapolate the eigenvectors of the $n \times n$ kernel matrix $\mathbf{K}_{i,j} = K(\upsilon_i, \upsilon_j)$:

$$\phi_m(x) = \frac{1}{\lambda_m}\sum_{i=1}^{n}\phi_m(\upsilon_i)K(\upsilon_i, x). \tag{35}$$

where $\{\lambda_m, \phi_m\}$ are the eigenvalues and eigenvectors of $\mathbf{K}$. Assuming the graph signal can be approximated using the top $M$ eigenfunctions

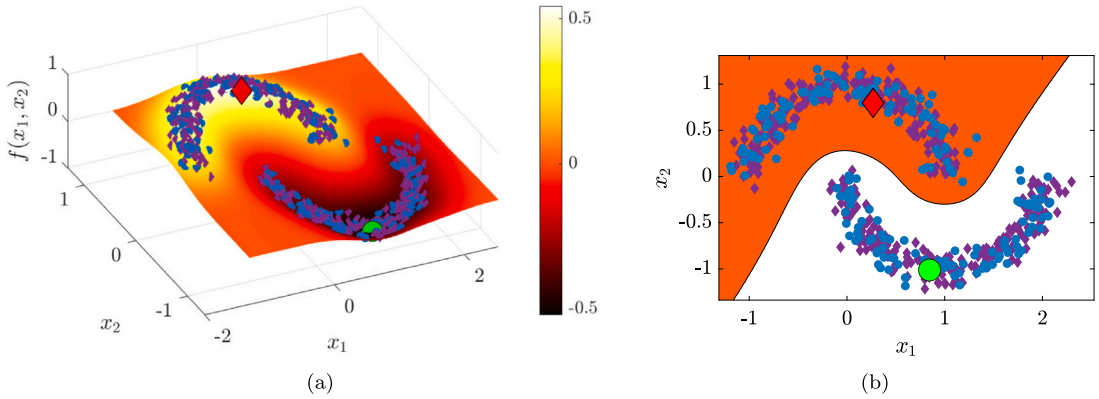


**Fig. 9.** Two-Moons RoMix continuous graph signal and classifier. Red diamond: first labeled point with label $y = +1$. Green dot: second labeled point with label $y = -1$. (a) Colored surface: Two-Moons RoMix continuous graph signal, $f_M(x_1, x_2)$. Blue dots: graph signal values of the interpolated $n = 200$ points. Purple diamonds: graph signal values of the extrapolated $N - n = 200$ points. (b) Two-Moons RoMix decision boundary where $f_M(x_1, x_2) = 0$. All points in the orange region will have the label $y = +1$ and all points in the white region will have the label $y = -1$. Blue dots: $n = 200$ given points. Purple diamonds: $N - n = 200$ extrapolation points.

**Table 2**
Baseline methods.

| Method | Description | Comments |
|---|---|---|
| Representer Theorem [29] | $\boldsymbol{s}^{\text{Rep.Thm.}} = \mathbf{K}\boldsymbol{\alpha} = \mathbf{K}(\mathbf{J}\mathbf{K} + \gamma\mathbf{I})^{-1}\boldsymbol{s}_{\ell}$<br>$\tilde{s}^{\text{Rep.Thm.}}(x) = \sum_{i=1}^{n} \alpha_i K(v_i, x)$ | • Used in [22,23]<br>• $\mathbf{K}_{i,j} = K(v_i, v_j)$<br>• $\mathbf{J} = \text{diag}(\mathbf{1}_{\ell}, \mathbf{0}_{n-\ell})$<br>• Interp. complexity: $\mathcal{O}(n^3)$<br>• Extrap. complexity: $\mathcal{O}(n)$ |
| Variational Splines and Paley-Wiener Spaces [21] | $\boldsymbol{s}^{\text{VSPW}} = \boldsymbol{\Phi}_{n\times\ell}\boldsymbol{\alpha} = \boldsymbol{\Phi}_{n\times\ell}\boldsymbol{\Phi}_{\ell\times\ell}^{-1}\boldsymbol{s}_{\ell}$ | • `gsp_interpolate()` from GSPBox [65]<br>• $\bar{\mathbf{L}} = \mathbf{L} + \varepsilon\mathbf{I}$<br>• $\boldsymbol{\Phi}_{n\times\ell}$ are Green's functions of $\bar{\mathbf{L}}$ at $\mathcal{V}$<br>• $\boldsymbol{\Phi}_{\ell\times\ell}$ are $\boldsymbol{\Phi}_{n\times\ell}$ at $\mathcal{V}_{\ell}$<br>• Interp. complexity: $\mathcal{O}(n^3)$ |
| Nyström method [19] | $\boldsymbol{s}^{\text{Nys}} = \boldsymbol{\Phi}_{n\times M}\boldsymbol{c} = \boldsymbol{\Phi}_{n\times M}\boldsymbol{\Phi}_{\ell\times M}^{\dagger}\boldsymbol{s}_{\ell}$<br>$\tilde{s}^{\text{Nys}}(x) = \sum_{m=1}^{M} c_m\phi_m(x)$<br>$= \sum_{m=1}^{M} \frac{c_m}{\lambda_m} \sum_{i=1}^{n} \phi_m(v_i)K(v_i, x)$ | • Used in [16]<br>• $\mathbf{K} = \boldsymbol{\Phi}\boldsymbol{\Lambda}\boldsymbol{\Phi}^T$<br>• Interp. complexity: $\mathcal{O}(M^2\ell)$<br>• Extrap. complexity: $\mathcal{O}(nM)$ |
| Weighted k-nearest neighbors [66] | For the vertex $v$:<br>1. $[u_1, \ldots, u_k, d_1, \ldots, d_k] = k\text{-NN}(v, \mathcal{V}_{\ell})$<br>2. $\forall i \in \{1, \ldots, k\}$ :<br>$a_i = 1/d_i, \quad w_i = a_i/\sum_{j=1}^{k} a_j$<br>3. $s^{\text{w-kNN}}(v) = \sum_{i=1}^{k} w_i s(u_i)$ | • Interp. complexity: $\mathcal{O}\big((n-\ell)\log_2(n-\ell)\big)$ |

of $K$, the interpolating and extrapolating functions using Nyström's method are given, respectively, by

$$\boldsymbol{s}^{\text{Nys}} = \boldsymbol{\Phi}_{n\times M}\boldsymbol{c} = \boldsymbol{\Phi}_{n\times M}\boldsymbol{\Phi}_{\ell\times M}^{\dagger}\boldsymbol{s}_{\ell}, \tag{36}$$

$$\tilde{s}^{\text{Nys}}(x) = \sum_{m=1}^{M} c_m\phi_m(x) = \sum_{m=1}^{M} \frac{c_m}{\lambda_m} \sum_{i=1}^{n} \phi_m(v_i)K(v_i, x), \tag{37}$$

where $\boldsymbol{\Phi}_{n\times M} \in \mathbb{R}^{n\times M}$ are the top $M$ eigenvectors of $\mathbf{K}$ at $\mathcal{V}$, $\boldsymbol{\Phi}_{\ell\times M}$ is the eigenvectors matrix $\boldsymbol{\Phi}_{n\times M}$ at the given $\mathcal{V}_{\ell}$ and $\dagger$ denotes the pseudo-inverse. The computational complexity of the interpolation is $\mathcal{O}(M^2\ell)$ due to the $\ell \times M$ matrix pseudo-inversion. The computational complexity of the extrapolation is $\mathcal{O}(nM)$ due to the $M$ evaluations of $\{c_m\phi_m(x)\}_{m=1}^{M}$ and $n$ evaluations according to (35).

The fourth baseline method is the weighted $k$ Nearest Neighbors (abbreviated w-kNN) [66]. In this method, for a node $v$ (either for interpolation, i.e., $v \in \mathcal{V}_{\ell}^{c}$, or for extrapolation, i.e., $v \in \mathcal{M} \setminus \mathcal{V}$) the following steps are performed. (1) Search for the $k$ nearest neighbors in terms of Euclidean distance. The $k$ closest neighboring vertices to $v$ are denoted $u_1, \ldots, u_k$ with distances $d_1, \ldots, d_k$. (2) Calculate the $k$ weights using the affinities defined by $a_i = 1/d_i$, having $w_i = a_i/\sum_{j=1}^{k} a_j$. (3) The interpolating graph signal is given by

$$s^{\text{w-kNN}}(v) = \sum_{i=1}^{k} w_i s(u_i) \tag{38}$$

The computational complexity of the interpolation is $\mathcal{O}\big((n-\ell)\log_2(n-\ell)\big)$ due to the use of MATLAB's `knnsearch()` which implements [68]. We remark that this method's interpolation and extrapolation processes are the same.

All four baseline methods utilize the graph structure, either through its adjacency matrix or its Laplacian, in order to infer the missing graph signal values. The Rep. Thm. method [29], the Nyström method [19], and w-kNN [66] leverage the graph adjacency matrix, which captures the connectivity of the nodes. The Rep. Thm. method and the Nyström method use a kernel function to generate the adjacency matrix, while w-kNN uses a $k$ nearest neighbors adjacency matrix. On the other hand, VSPW [21] captures the graph structure using the graph Laplacian obtained from its Green functions.

### 5.3.2. Handwritten digits classification

We examine the classification accuracy of handwritten digits from the $28 \times 28 = 784$-dimensional MNIST dataset [60] using Algorithm 1 and the baseline methods (see Table 2). We present 50 random samples

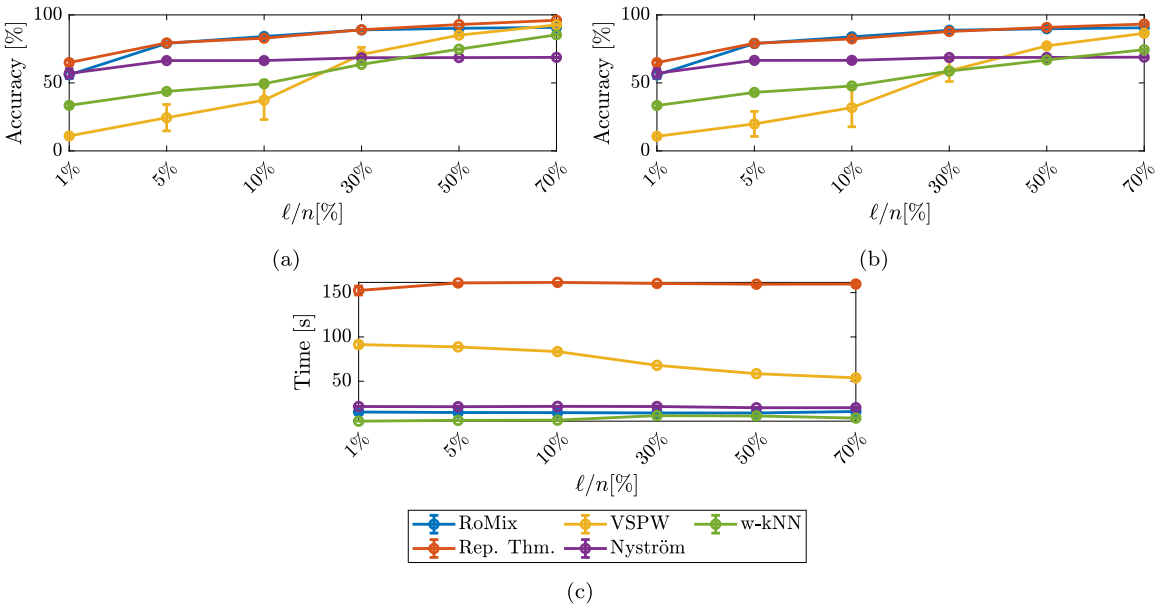


**Fig. 10.** MNIST (with VAE): digits classification accuracy and run times results. (a) Interpolation accuracy (23) of each method ($n$ images). (b) Extrapolation accuracy (23) of each method ($N$ images). (c) Interpolation time of each method.

**Table 3**
MNIST: Algorithm 1 parameters.

| Parameter | Value |
|---|---|
| Kernel width | $\omega = 2.28$ |
| # GMM comp. | $k = 10$ |
| # eigs. | $M = 1000$ |
| $\|\cdot\|_{\mathcal{H}_K}$ penalty | $\gamma = 0.1$ |

from the dataset in Fig. 4(a). We begin by building the latent space, $\mathcal{Z}$, in which we test all the algorithms. We train a VAE with a latent space of dimension $d = 20$ using the 60,000 training images from the dataset. In each experiment, we select a varying number of labeled images $\ell$ from which all methods perform the interpolation/extrapolation. We run $R = 5$ Monte-Carlo iterations in each experiment. In each iteration, we perform the following procedure. We randomly select a set of $N = 34{,}000$ images and their corresponding labels $\{y_i\}_{i=1}^{N}$ out of the 70,000 images of MNIST. From that set, we randomly select $n = 24{,}000$ ($n < N$) images. We keep the remaining $N - n = 10{,}000$ labels to evaluate the extrapolation performance. We view the corresponding latent representations denoted by $\{\boldsymbol{v}_i\}_{i=1}^{n}$ as the inputs of each of the tested methods and construct the input matrix $\mathbf{V} \in \mathbb{R}^{n\times d}$. From the $n$ latent representations, we randomly select the labeled subset $\{\boldsymbol{v}_i\}_{i=1}^{\ell}$ with the corresponding labels $\{y_i\}_{i=1}^{\ell}$. We keep the remaining $n - \ell$ labels to evaluate the interpolation performance. We then construct 10 indicator vectors $\{\boldsymbol{s}_\ell^c\}_{c=0}^{9}$ from the labels $\{y_i\}_{i=1}^{\ell}$, one for each digit, that act as 10 graph signals. For the digit $c \in \{0,\ldots,9\}$, $\boldsymbol{s}_\ell^c(i) = 1$ if $y_i = c$ and $\boldsymbol{s}_\ell^c(i) = 0$ if $y_i \neq c$. We present the parameters chosen for Algorithm 1 in Table 3. We show the obtained accuracy results and corresponding interpolation run times in Fig. 10. We can observe that our method, RoMix, achieves the best accuracy along with the Representer theorem. However, our approach requires much lower run times.

We now discuss the impact of applying the VAE pre-processing step. We conduct an ablation test where we run the same experiment described previously in this section using the same parameters in Table 3, without the VAE pre-processing step. We present the results in Fig. 11. We observe the following implications. The Rep. Thm. method accuracy and computation complexity were not significantly impacted. The accuracy of the Nyström method was improved but its computational complexity degraded. The accuracy of w-kNN was improved, but the computation complexity was significantly impaired. The VSPW method accuracy degraded, and its computation complexity was not significantly impacted. The accuracy of our method, RoMix, was not significantly impacted, while the computational complexity was significantly reduced. This corroborates the benefit of the VAE pre-processing step for Algorithm 1 (RoMix).

### 5.3.3. *Bulgaria beacons*

We now test our algorithm in an application to a sensor network. Given a Digital Elevation Model (DEM) of a certain terrain in Bulgaria [69], we wish to estimate a received Radio Frequency (RF) signal power in the entire terrain based on a small number of sensors measuring the received signal power at their positions. We present the DEM in Fig. 12(a). The terrain size is one by one degree in longitude and latitude (approximately 110 [km] on each axis). The topography of the terrain greatly affects the propagation of the RF signals. We begin by generating the ground-truth graph signal as the measurements of the sensor network by simulating the propagation using the commercial software EDX® SignalPro®.

To this end, we place two transmitters ($TX_1$ and $TX_2$) on the DEM at random locations (designated by blue diamonds in Fig. 12(a)), and generate measurements of their received powers on every point of the DEM. We follow the same configuration introduced in [70]. Specifically, we use the Anderson-2D for the path-loss model and include ground reflections and Fresnel zones in the computation. We set the transmitters to be omnidirectional, hand-held, at 2 [m] height with a transmission frequency of $f = 150$ [MHz]. The transmitters' power is 0 [dBm], and the processing bandwidth is assumed to be 100 [Hz]. We assume the thermal noise density is $-174$ [dBm/Hz], such that the noise floor is at $-154$ [dBm]. We present the simulated received signal power at each point of the DEM in Fig. 12(b). To avoid long simulation times, we sample the DEM along each axis and form a grid of $N = 100\times100$ nodes. The sampling intervals are approximately 1.1[km]. We then define $\bar{\boldsymbol{s}} \in \mathbb{R}^N$, the ground-truth graph signal, by sampling the simulated received signal power at the closest points on the sampled grid. $\bar{\boldsymbol{s}}$ will be used to evaluate the extrapolation performance and is presented in Fig. 12(c). From $\bar{\boldsymbol{s}}$ we randomly sample $n$ points and form the ground-truth graph signal $\boldsymbol{s} \in \mathbb{R}^n$ which will be used to evaluate the interpolation performance. We present it in Fig. 12(d).

In order to construct a graph, we extract four features for each point on the grid of $N$ nodes, having a latent space $\mathcal{Z}$. The first feature is the

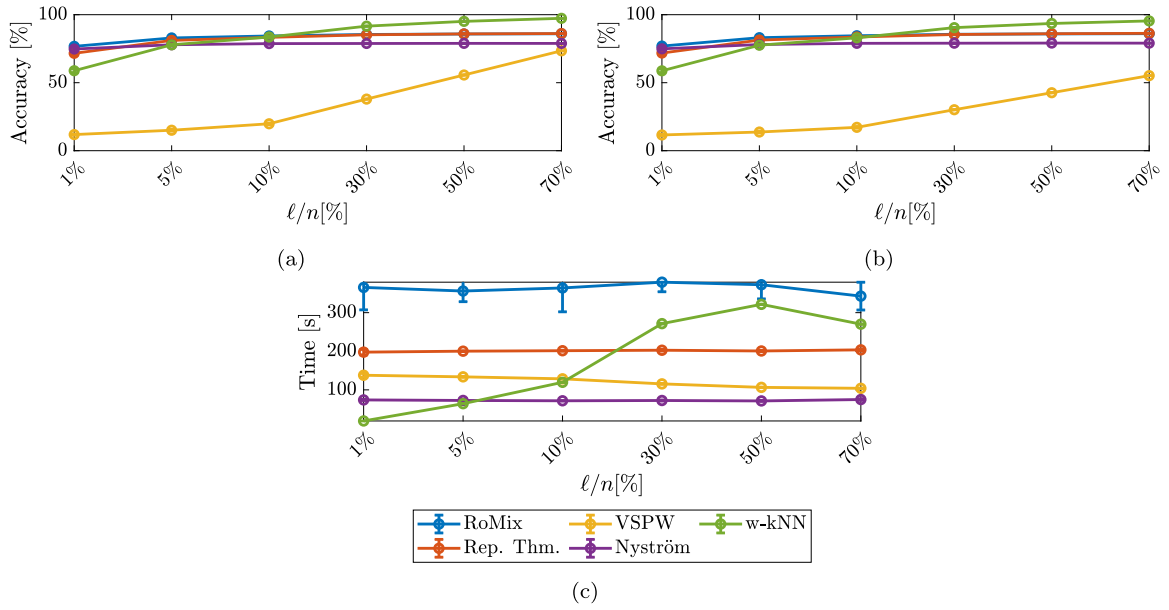


**Fig. 11.** MNIST (without VAE): digits classification accuracy and run times ablation test results. (a) Interpolation accuracy (23) of each method ($n$ images). (b) Extrapolation accuracy (23) of each method ($N$ images). (c) Interpolation time of each method.

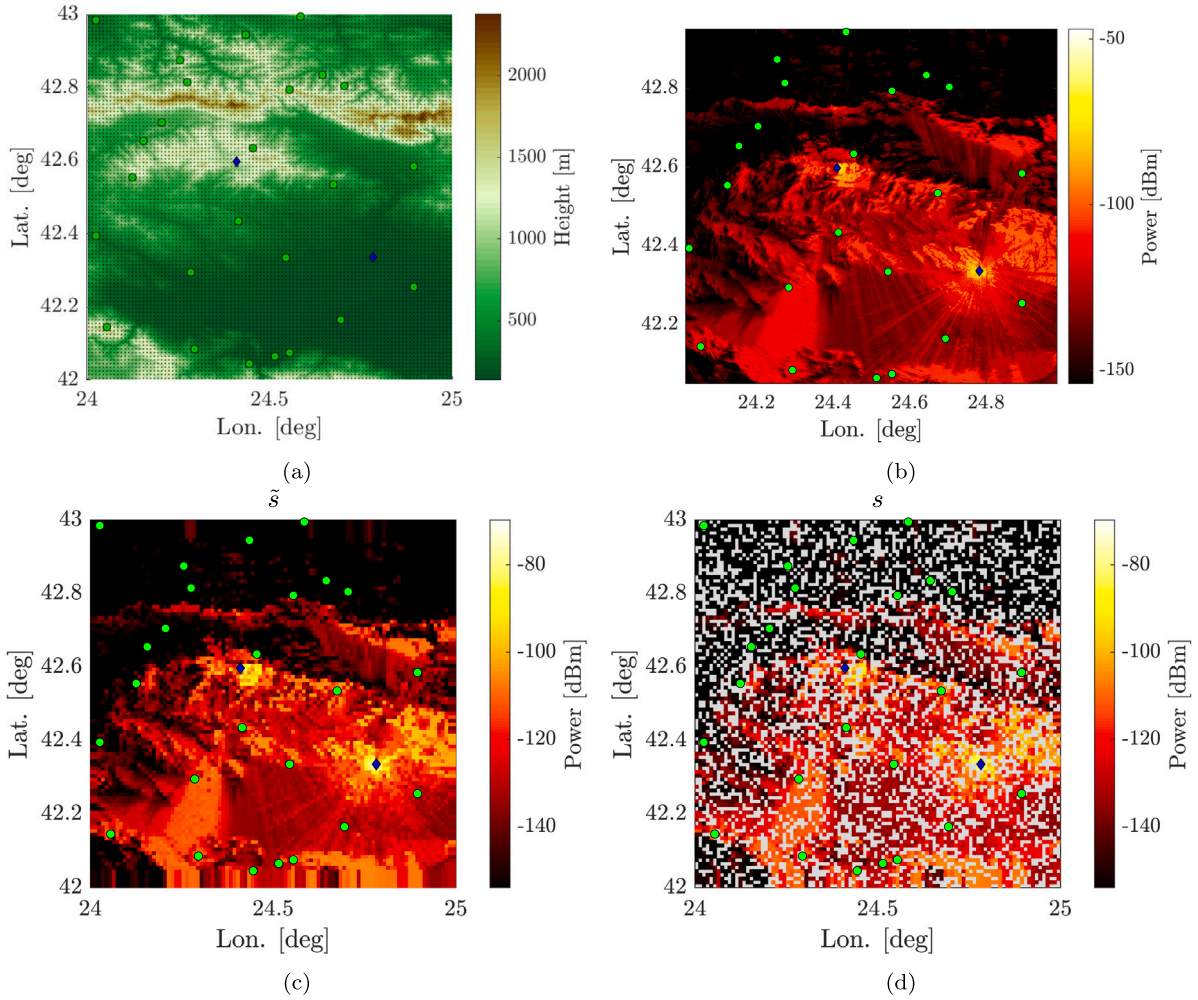


**Fig. 12.** Bulgaria beacons: DEM, received RF signal power simulated using EDX® SignalPro® and ground-truth graph signals. Blue diamonds: two transmitters. Green circles: $\ell = 25$ sensors. (a) Bulgaria DEM. Black dots: grid of $N = 100 \times 100$ nodes. (b) SignalPro® simulation result on the DEM points. (c) Ground-truth graph signal for extrapolation. (d) Ground-truth graph signal for interpolation.

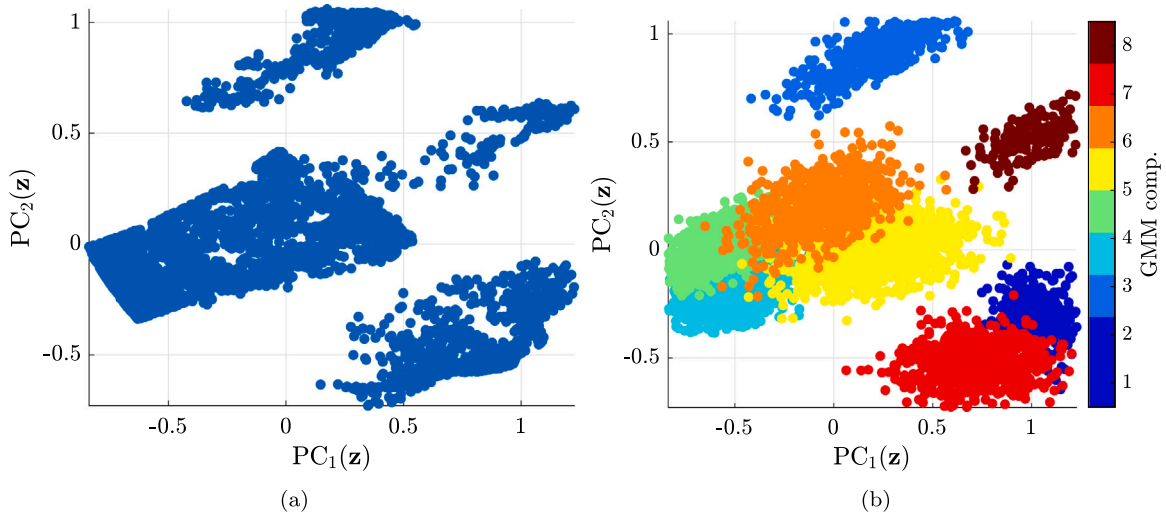


**Fig. 13.** Bulgaria beacons: manifold of Gaussian mixture demonstration. The first two principal components of PCA. (a) Given $n$ points. (b) Points generated from learned GMM parameters.

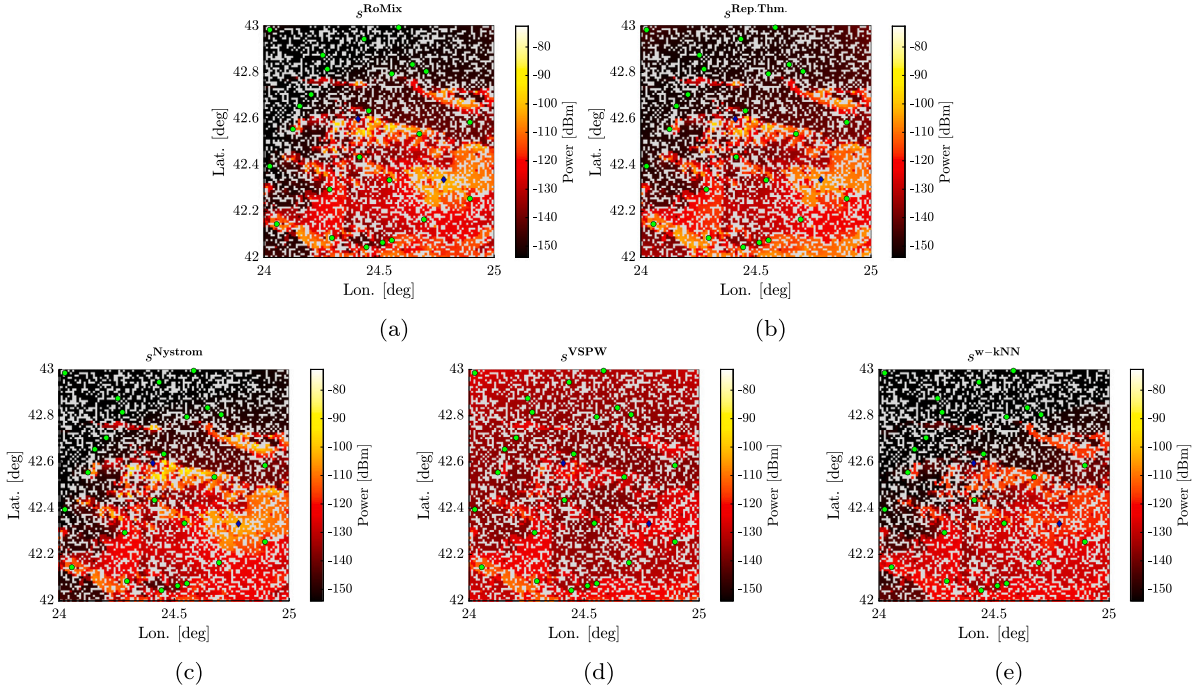


**Fig. 14.** Bulgaria beacons: interpolated graph signals on $n$ points sampled from the grid of $N$ nodes. Taken from the first iteration in the first experiment with $\ell = 25$ sensors. Blue diamonds: two transmitters. Green circles: $\ell = 25$ sensors. (a) RoMix. (b) Representer Theorem [29]. (c) Nyström method [19]. (d) Variational Spline in Paley–Weiner space [21]. (e) Weighted k-nearest neighbors [66].

position, $\boldsymbol{p}_i = [x_i, y_i, z_i]^T$. The second feature is the Free-Space Path-Loss (FSPL) from each of the two transmitters to $\boldsymbol{p}_i$, i.e., for $j = 1, 2$: $\text{FSPL}(\boldsymbol{p}_i, \boldsymbol{p}_{\text{TX}_j}) = \left(4\pi\|\boldsymbol{p}_i - \boldsymbol{p}_{\text{TX}_j}\|_{\mathbb{R}^3}/\lambda\right)^2$ where $\lambda = c/f$, and $c$ is the speed of light. The third feature is the indication $\{0, 1\}$ of Line of Sight (LoS) existence between each transmitter to the point $i$. The last feature is the LoS$^2$, which counts the number of LoS paths between a transmitter to the point $i$ through any other neighboring point in the grid. Each node $i$ is therefore associated with the 9-$d$ feature vector $\boldsymbol{z}_i \in \mathcal{Z}$:

$$\boldsymbol{z}_i = [\ \boldsymbol{p}_i^T,\ \{\text{FSPL}(\boldsymbol{p}_i, \boldsymbol{p}_{\text{TX}_j})\}_{j=1}^2,\ \{\text{LoS}(\boldsymbol{p}_i, \boldsymbol{p}_{\text{TX}_j})\}_{j=1}^2,\ \{\text{LoS}^2(\boldsymbol{p}_i, \boldsymbol{p}_{\text{TX}_j})\}_{j=1}^2\ ]^T.$$

Finally, for numerical stability, we normalize the features to have an even scale between $[0, 1]$. We summarize the features in Table 4.

We visualize the first step of Algorithm 1, the GMM fit, to $n = 6000$ randomly sampled points from the grid of the corresponding latent representations $\{\boldsymbol{z}_i\}_{i=1}^n$. For the visualization, we reduce the dimension of the sampled points to $\mathbb{R}^2$ using a Principal Component Analysis (PCA) [71]. The results are presented in Fig. 13(a). We then randomly generate new samples from the learned GMM parameters with $k = 8$ components and perform a similar visualization using PCA. The result is

**Table 4**
Bulgaria beacons: 9-$d$ latent space representation.

| Feature | Expression | Dimension |
|---|---|---|
| Position | $\boldsymbol{p}_i = [x_i, y_i, z_i]^T$ | 3 |
| Free Space Path Loss | $\left(4\pi\|\boldsymbol{p}_i - \boldsymbol{p}_{\text{TX}_j}\|_{\mathbb{R}^3}/\lambda\right)^2$ | 2 (one for each TX) |
| Line of Sight indication | $\text{LoS}(\boldsymbol{p}_i, \boldsymbol{p}_{\text{TX}_j}) \in \{0, 1\}$ | 2 (one for each TX) |
| No. of 2-hop Line of Sight paths | $\text{LoS}^2(\boldsymbol{p}_i, \boldsymbol{p}_{\text{TX}_j})$ | 2 (one for each TX) |

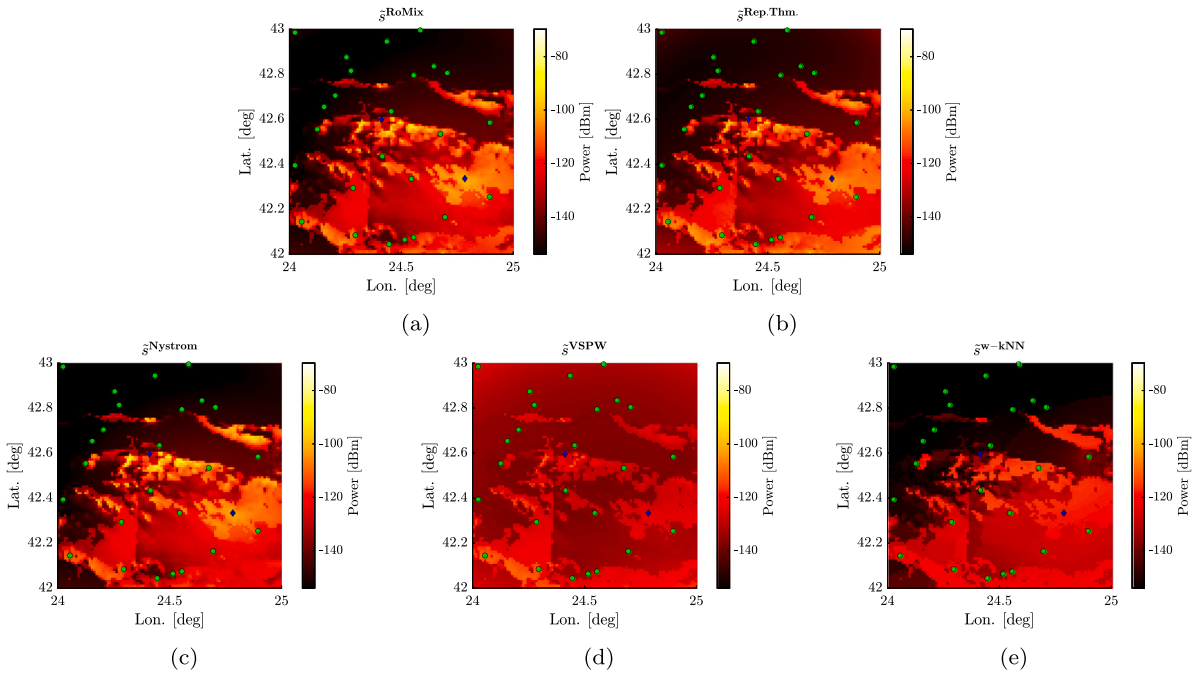


**Fig. 15.** Bulgaria beacons: extrapolated graph signals on the grid of $N$ nodes. Taken from the first iteration in the first experiment with $\ell = 25$ sensors. Blue diamonds: two transmitters. Green circles: $\ell = 25$ sensors. (a) RoMix. (b) Representer Theorem [29]. (c) Nyström method [19]. (d) Variational Spline in Paley-Weiner space [21]. (e) Weighted k-nearest neighbors [66].

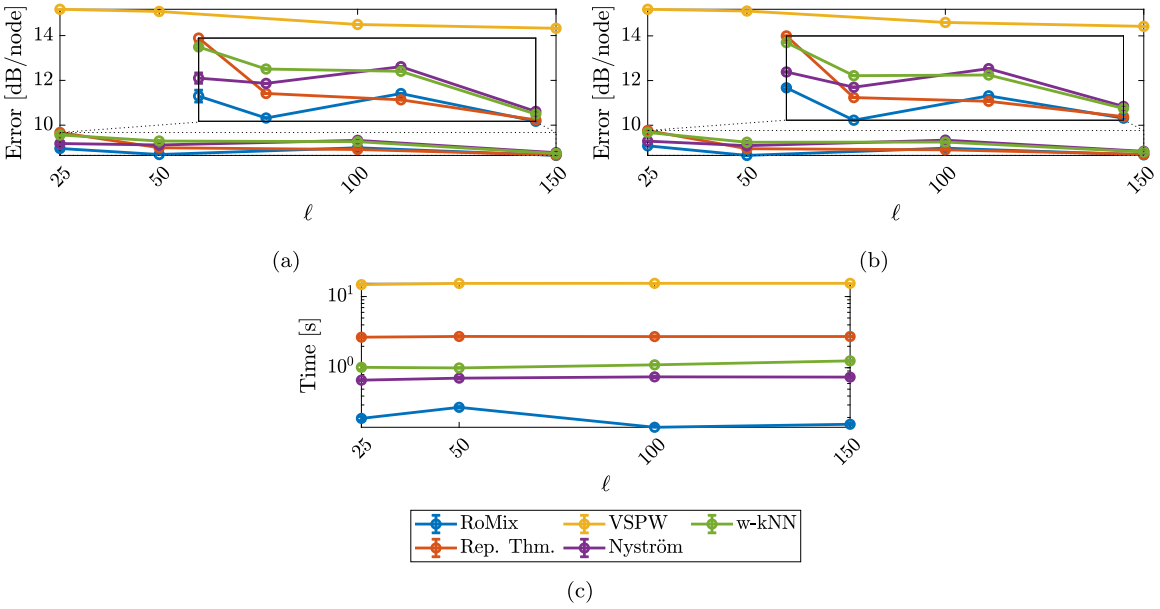


**Fig. 16.** Bulgaria beacons: errors and run times results. (a) Interpolation error (39) of each method ($n$ points). Black box: focus on the bottom of the figure. (b) Extrapolation error (39) of each method (entire grid of $N$ points). Black box: focus on the bottom of the figure. (c) Interpolation time of each method (log scale).

presented in Fig. 13(b), where each color represents the most probable component to which the generated point relates. We can observe four point clouds due to the LoS feature with two transmitters, i.e., there are $|\{0,1\}|^2 = 4$ options for LoS existence from two transmitters to each node. We observe that the obtained first two principal components of the generated points resemble the first two principal components of the sampled $n$ points. We note that the $k = 8$ components partially overlap when projecting the generated points on the first two principal components. However, we assume the separation in the higher 9-$d$ latent space is better.

To test the performance of Algorithm 1 and the competing methods presented in Table 2, we run four experiments with different values of $\ell$, the number of sensors spread on the grid of $N$ nodes. In each experiment, we run $R = 10$ Monte-Carlo iterations of the following procedure. We randomly sample $n = 6000$ points from the latent space of $N = 100 \times 100$ feature vectors and arrange them in the input matrix $\mathbf{V} \in \mathbb{R}^{n \times d}$. Then, we randomly select $\ell < n$ nodes for the $\ell$ sensors placements and build the graph signal on the $\ell$ subset of nodes from $\tilde{s} \in \mathbb{R}^N$ at the locations of the sensors, having $s_\ell$. Algorithm 1 parameters are presented in Table 5. Samples from the first iteration in the first experiment with $\ell = 25$ sensors are presented in Fig. 14. The title of each sub-figure states the name of the interpolation method. Similarly, we present in Fig. 15 the extrapolation results on the entire grid. Since the graph signal values are in units of [dBm], we set the metric in this

**Table 5**
Bulgaria beacons: Algorithm 1 parameters.

| Parameter | Value |
|---|---|
| Kernel width | $\omega = 1.5$ |
| # GMM comp. | $k = 8$ |
| # eigs. | $M = 100$ |
| $\|\cdot\|_{\mathcal{H}_K}$ penalty | $\gamma = 0.3$ |

simulation to be the error in dB per node:

$$\mathrm{Err}(\boldsymbol{s}^{\mathrm{algo}}, \boldsymbol{s}) = \frac{1}{R}\sum_{r=1}^{R}\frac{1}{\sqrt{n}}\left\|\boldsymbol{s}_r^{\mathrm{algo}} - \boldsymbol{s}\right\| \quad [\mathrm{dB/node}]. \tag{39}$$

The obtained results in terms of the error metric in (39) and the corresponding interpolation run times are presented in Fig. 16. We observe that the shortest run time and lowest error per node are achieved by Algorithm 1.

## 6. Conclusion

In this work, we propose a continuous domain approach for graph signal interpolation and extrapolation. We present a continuous domain model for graph signals, interpolation and extrapolation algorithm, and several synthetic and real-world test cases for performance evaluation. Our model relies on the typical construction of the graph adjacency matrix using a kernel function. In this case, a unique RKHS of functions is imposed. Therefore, we model a (discrete) graph signal over the (discrete) set of graph nodes as samples of a (continuous) function over a (continuous) manifold in that RKHS. We focus on the popular choice of a Gaussian kernel. This kernel provides us with closed-form expressions for the eigenbasis of the RKHS when the data distribution is Gaussian. Under a Gaussian mixture, the eigenbasis can be approximated using those closed-form expressions. Therefore, we model the manifold of nodes as a Gaussian mixture, either directly in the ambient space or indirectly in an embedded space. We then exploit the closed-form eigenbasis expressions in our interpolation and extrapolation efficient algorithm. We compare our algorithm performance to existing methods and show that it achieves on-par or better accuracy while being computationally more efficient and requiring shorter run times.

## CRediT authorship contribution statement

**Itay Zach:** Conceptualization, Methodology, Writing – original draft, Software. **Tsvi G. Dvorkind:** Conceptualization, Methodology, Supervision, Writing – review & editing. **Ronen Talmon:** Conceptualization, Methodology, Supervision, Writing – review & editing.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Data availability

Code and data at github.com/itayzach/RoMix.

## Acknowledgments

We wish to thank the Associate Editor and the reviewers for their useful comments, which helped us to improve the manuscript significantly.

## Appendix A. Elaborated reproducing kernel Hilbert space theory

All definitions were taken from [27,28] (based on [25]) with adjustments to our notations and to the field of real numbers, $\mathbb{R}$, rather than the complex numbers, $\mathbb{C}$. Refer to [26] for an excellent discussion and further references for this section.

**Definition A.1** (*Reproducing Kernel Hilbert Space (RKHS) (Ch. 1 of [27])*)**.** Let $\mathcal{X}$ be a set and denote by $\mathcal{F}(\mathcal{X}, \mathbb{R})$ the set of all functions from $\mathcal{X}$ to $\mathbb{R}$. We will call a subset $\mathcal{H} \subset \mathcal{F}(\mathcal{X}, \mathbb{R})$ a reproducing kernel Hilbert space on $\mathcal{X}$ over $\mathbb{R}$ if:

1. $\mathcal{H}$ is a vector subspace of $\mathcal{F}(\mathcal{X}, \mathbb{R})$.
2. $\mathcal{H}$ is a Hilbert space endowed with an inner product $\langle\cdot,\cdot\rangle_{\mathcal{H}}$.
3. For every $x \in \mathcal{X}$, the linear evaluation functional $\delta_x : \mathcal{H} \to \mathbb{R}$ defined by $\delta_x(f) = f(x)$ is bounded.

**Theorem A.1** (*Riesz Representation Theorem (Ch. 2 of [28])*)**.** *If $L$ is a bounded linear functional on a Hilbert space $\mathcal{H}$, then there exists a unique $g \in \mathcal{H}$ such that for all $f \in \mathcal{H}$, $L(f) = \langle f, g\rangle_{\mathcal{H}}$.*

By applying the Riesz representation theorem on an RKHS $\mathcal{H}$, and since a linear functional on a Hilbert space is bounded if and only if it is continuous, we can say that for every $x \in \mathcal{X}$ there exists a unique function we denote by $k_x \in \mathcal{H}$ such that for every $f \in \mathcal{H}$, $f(x) = \delta_x(f) = \langle f, k_x\rangle_{\mathcal{H}}$.

**Definition A.2** (*Reproducing Kernel (Ch. 1 of [27])*)**.** The function $k_x$ above is the reproducing kernel for the point $x$. The function $K : \mathcal{X}\times\mathcal{X} \to \mathbb{R}$ defined by $K(x, y) = k_y(x)$ is called the reproducing kernel for $\mathcal{H}$.

A direct corollary of Definition A.2 is that a reproducing kernel must be a symmetric function since $K(x, y) = k_y(x) = \langle k_y, k_x\rangle_{\mathcal{H}} = \langle k_x, k_y\rangle_{\mathcal{H}} = K(y, x)$ where the third equality is due to working with $\mathbb{R}$.

**Definition A.3** (*Kernel Function (Ch. 2 of [27])*)**.** Let $\mathcal{X}$ be a set and let $K : \mathcal{X}\times\mathcal{X} \to \mathbb{R}$ be a function of two variables. Then $K$ is called a kernel function provided that for every $n$ and for every choice of distinct points $\{x_1, \ldots, x_n\} \subseteq \mathcal{X}$ the matrix $\mathbf{K}$ defined by $\mathbf{K}_{i,j} = K(x_i, x_j)$ is positive semi-definite matrix. i.e., $\mathbf{x}^T\mathbf{K}\mathbf{x} \geq 0$, where $\mathbf{x} = [x_1, \ldots, x_n]^T$.

It is possible to show (Ch. 2 of [27]) that there is a unique reproducing kernel associated with a given RKHS. Conversely, as the Moore-Aronszajn theorem suggests (Ch. 2 of [27]), given a kernel function (Definition A.3) $K : \mathcal{X}\times\mathcal{X} \to \mathbb{R}$ there exists a unique RKHS $\mathcal{H}$ for which $K$ is a reproducing kernel. In order to emphasize this property, we denote the RKHS defined by the kernel $K$ as $\mathcal{H}_K$ and its inner product by $\langle\cdot,\cdot\rangle_{\mathcal{H}_K}$. In our context, this implies that whenever the affinity of graph nodes is defined by a kernel function $K$ we can associate an RKHS, $\mathcal{H}_K$, within which we derive our 'continuous-domain' models. The corresponding RKHS of $K$ has *the reproducing property*: any function in the space can be expressed in terms of its kernel function, i.e., $f(x) = \langle f, k_x\rangle_{\mathcal{H}_K}$.

### A.1. Spectral theorem for the kernel operator

Consider the space of square-integrable real functions over the ambient space $\mathcal{X}$, with respect to the probability measure $P_{\mathcal{X}}$. We denote the space by $\mathcal{L}^2_p(\mathcal{X})$ where $p$ is the probability density, i.e., $dP_{\mathcal{X}} = p(x)dx$, and denote the inner product in $\mathcal{L}^2_p(\mathcal{X})$ by $\langle\cdot,\cdot\rangle_{\mathcal{L}^2_p}$. A function $f$ belongs to $\mathcal{L}^2_p(\mathcal{X})$ if the following holds:

$$\int_{\mathcal{X}} f(x)^2 p(x)dx < \infty,$$

and the inner product for $f, g \in \mathcal{L}^2_p(\mathcal{X})$ is:

$$\langle f, g\rangle_{\mathcal{L}^2_p} = \int_{\mathcal{X}} f(x)g(x)p(x)dx. \tag{A.1}$$

The operator $L_K$ (4) is compact (Theorem 2.29 in [72]) and self-adjoint (Ch. 3 of [26]) and therefore satisfies the conditions for the Spectral Theorem (Ch. 2 of [26], Sec. 4.10 of [73]) stating that for every compact, self-adjoint operator on an infinite-dimensional Hilbert space, there exists an orthonormal system of eigenfunctions denoted by $\{\phi_m\}_{m=0}^{\infty}$ corresponding to real, non-negative eigenvalues denoted by $\{\lambda_m\}_{m=0}^{\infty}$ that are countable and can be sorted in decreasing order, i.e., $\lambda_0 \geq \lambda_1 \geq \cdots \geq 0$. Furthermore, the eigenvalues decay to zero, i.e., $\lambda_m \xrightarrow[m\to\infty]{} 0$. We, therefore, have the following:

$$(L_K\phi_m)(x) = \langle k_x, \phi_m\rangle_{\mathcal{L}_p^2} = \lambda_m\phi_m(x). \tag{A.2}$$

This allows us to express any function $f \in \mathcal{H}_K$ in terms of the kernel operator eigenfunctions. Namely, if $f \in \mathcal{H}_K$ there exist expansion coefficients $\{c_m\}_{m=0}^{\infty} \in \mathbb{R}$ such that:

$$f(x) = \sum_{m=0}^{\infty} c_m\phi_m(x), \quad c_m = \langle f, \phi_m\rangle_{\mathcal{L}_p^2}, \ \forall x \in \mathcal{X}. \tag{A.3}$$

The inner product (A.1) can be therefore expressed in terms of the expansion coefficients. For $f = \sum_m c_m\phi_m, g = \sum_\ell c'_\ell\phi_\ell$:

$$\langle f, g\rangle_{\mathcal{L}_p^2} = \sum_{m,\ell=0}^{\infty} c_m c'_\ell \langle\phi_m, \phi_\ell\rangle_{\mathcal{L}_p^2} = \sum_{m=0}^{\infty} c_m c'_m. \tag{A.4}$$

This construction of the RKHS is used by Mercer theorem [58], meaning the kernel function can be expressed using the eigenfunctions in (A.2):

$$K(x, y) = \sum_{m=0}^{\infty} \lambda_m\phi_m(x)\phi_m(y).$$

For the induced norm (7) to be convergent, the function expansion coefficients must decay faster than the kernel operator eigenvalues, i.e., $\left(\frac{c_m}{\sqrt{\lambda_m}}\right) \in \ell^2$. Intuitively, by interpreting the eigenvalues as frequencies, a function in the RKHS must be "smooth" over its domain.

## Appendix B. Example: VAE computational complexity analysis

Consider a conventional Keras implementation of a convolutional VAE [74]. A forward pass of a convolutional layer for a batch of $b$ inputs where each input dimension is $d_{\text{in}}^2$ of $c_{\text{in}}$ channels, a filter of size $f \times f$ and $c_{\text{out}}$ output channels, has a time complexity of $\mathcal{O}(bd_{\text{in}}^2 c_{\text{in}} f^2 c_{\text{out}})$. A forward pass of a fully connected layer for a batch of $b$ inputs and $w$ weights is $\mathcal{O}(bd_{\text{in}}^2 c_{\text{in}} w)$. In the encoder, there are two convolutional layers followed by a fully connected layer, and in the decoder, there is a fully connected layer followed by two convolutional layers. Since the filter size in a convolutional layer is much smaller than the batch size and the input dimensionality, i.e., $f \ll b$, and $f \ll d_{\text{in}}^2$, and the fact that the dimensionality in a VAE decreases with each layer, it is possible to bound the overall complexity by $\mathcal{O}(bd^2)$. Since the backward pass is less computationally demanding, the overall training of a VAE can be bounded by $\mathcal{O}(tbd^2)$ where $t$ is the number of epochs.

## References


[1] D.I. Shuman, S.K. Narang, P. Frossard, A. Ortega, P. Vandergheynst, The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains, IEEE Signal Process. Mag. 30 (3) (2013) 83–98, http://dx.doi.org/10.1109/MSP.2012.2235192.

[2] A. Ortega, P. Frossard, J. Kovacevic, J.M. Moura, P. Vandergheynst, Graph signal processing: Overview, challenges, and applications, Proc. IEEE (2018) http://dx.doi.org/10.1109/JPROC.2018.2820126.

[3] F.R.K. Chung, Spectral Graph Theory, Vol. 92, American Mathematical Soc., 1997.

[4] S. Colonnese, M. Petti, L. Farina, G. Scarano, F. Cuomo, Protein-protein interaction prediction via graph signal processing, IEEE Access 9 (2021) http://dx.doi.org/10.1109/ACCESS.2021.3119569.

[5] D.M. Mohan, M.T. Asif, N. Mitrovic, J. Dauwels, P. Jaillet, Wavelets on graphs with application to transportation networks, in: 2014 17th IEEE International Conference on Intelligent Transportation Systems, ITSC 2014, 2014, pp. 1707–1712, http://dx.doi.org/10.1109/ITSC.2014.6957939.

[6] K. He, L. Stankovic, J. Liao, V. Stankovic, Non-intrusive load disaggregation using graph signal processing, IEEE Trans. Smart Grid 9 (3) (2018) 1739–1747, http://dx.doi.org/10.1109/TSG.2016.2598872.

[7] M.M. Bronstein, J. Bruna, Y. Lecun, A. Szlam, P. Vandergheynst, Geometric deep learning: Going beyond Euclidean data, IEEE Signal Process. Mag. 34 (4) (2017) 18–42, http://dx.doi.org/10.1109/MSP.2017.2693418.

[8] Z. Huang, W. Chung, T.-H. Ong, H. Chen, A graph-based recommender system for digital library, in: Proceedings of the 2nd ACM/IEEE-CS Joint Conference on Digital Libraries, 2002, pp. 65–73, http://dx.doi.org/10.1145/544229.544231.

[9] S.K. Narang, A. Gadde, A. Ortega, Signal processing techniques for interpolation in graph structured data, in: ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings, 2013, pp. 5445–5449, http://dx.doi.org/10.1109/ICASSP.2013.6638704.

[10] S. Chen, A. Sandryhaila, J.M. Moura, J. Kovacevic, Signal recovery on graphs: Variation minimization, IEEE Trans. Signal Process. 63 (17) (2015) 4609–4624, http://dx.doi.org/10.1109/TSP.2015.2441042.

[11] M. Kaneko, G. Cheung, W.T. Su, C.W. Lin, Graph-based joint signal/power restoration for energy harvesting wireless sensor networks, in: 2017 IEEE Global Communications Conference, GLOBECOM 2017 - Proceedings, Vol. 2018-Janua, 2017, pp. 1–6, http://dx.doi.org/10.1109/GLOCOM.2017.8254798.

[12] V. Kalofolias, How to learn a graph from smooth signals, in: A. Gretton, C.C. Robert (Eds.), Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, in: Proceedings of Machine Learning Research, vol. 51, PMLR, Cadiz, Spain, 2016, pp. 920–929, URL https://proceedings.mlr.press/v51/kalofolias16.html.

[13] X. Dong, D. Thanou, P. Frossard, P. Vandergheynst, Learning Laplacian matrix in smooth graph signal representations, IEEE Trans. Signal Process. 64 (23) (2016) 6160–6173, http://dx.doi.org/10.1109/TSP.2016.2602809.

[14] A. Gadde, A. Anis, A. Ortega, Active semi-supervised learning using sampling theory for graph signals, in: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2014, pp. 492–501, http://dx.doi.org/10.1145/2623330.2623760.

[15] S. Chen, R. Varma, A. Sandryhaila, J. Kovačević, Discrete signal processing on graphs: Sampling theory, IEEE Trans. Signal Process. (2015) http://dx.doi.org/10.1109/TSP.2015.2469645.

[16] A. Heimowitz, Y.C. Eldar, The Nyström extension for signals defined on a graph, in: ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings, 2018, pp. 4199–4203, http://dx.doi.org/10.1109/ICASSP.2018.8462408.

[17] S. Mousazadeh, I. Cohen, Out-of-sample extension of band-limited functions on homogeneous manifolds using diffusion maps, Signal Process. 108 (2015) http://dx.doi.org/10.1016/j.sigpro.2014.10.024.

[18] R.R. Coifman, S. Lafon, Diffusion maps, Appl. Comput. Harmon. Anal. 21 (1) (2006) 5–30, http://dx.doi.org/10.1016/j.acha.2006.04.006.

[19] C. Fowlkes, S. Belongie, F. Chung, J. Malik, Spectral grouping using the Nyström method, IEEE Trans. Pattern Anal. Mach. Intell. (2004) http://dx.doi.org/10.1109/TPAMI.2004.1262185.

[20] D.I. Shuman, M.J. Faraji, P. Vandergheynst, A multiscale pyramid transform for graph signals, IEEE Trans. Signal Process. 64 (8) (2016) 2119–2134, http://dx.doi.org/10.1109/TSP.2015.2512529.

[21] I. Pesenson, Variational splines and Paley-Wiener spaces on combinatorial graphs, Constr. Approx. (2009) http://dx.doi.org/10.1007/s00365-007-9004-9.

[22] D. Romero, M. Ma, G.B. Giannakis, Kernel-based reconstruction of graph signals, IEEE Trans. Signal Process. 65 (3) (2017) 764–778, http://dx.doi.org/10.1109/TSP.2016.2620116.

[23] V.N. Ioannidis, M. Ma, A.N. Nikolakopoulos, G.B. Giannakis, D. Romero, Chapter 8 - Kernel-based inference of functions over graphs, in: D. Comminiello, J.C. Príncipe (Eds.), Adaptive Learning Methods for Nonlinear System Modeling, Butterworth-Heinemann, 2018, pp. 173–198, http://dx.doi.org/10.1016/B978-0-12-812976-0.00010-5, URL https://www.sciencedirect.com/science/article/pii/B9780128129760000105.

[24] W. Erb, Graph signal interpolation with positive definite graph basis functions, Appl. Comput. Harmon. Anal. 60 (2022) 368–395, http://dx.doi.org/10.1016/J.ACHA.2022.03.005.

[25] N. Aronszajn, Theory of reproducing kernels, Trans. Amer. Math. Soc. (1950) http://dx.doi.org/10.2307/1990404.

[26] F. Cucker, S. Smale, On the mathematical foundations of learning, Bull. Amer. Math. Soc. 39 (1) (2002) 1–49, http://dx.doi.org/10.1090/S0273-0979-01-00923-5.

[27] V.I. Paulsen, M. Raghupathi, An Introduction to the Theory of Reproducing Kernel Hilbert Spaces, Cambridge University Press, 2016, http://dx.doi.org/10.1017/CBO9781316219232.

[28] I. Gohberg, S. Goldberg, M.A. Kaashoek, Spectral theory of integral operators, in: Basic Classes of Linear Operators, Birkhäuser Basel, Basel, 2003, pp. 193–202, http://dx.doi.org/10.1007/978-3-0348-7980-4_5.

[29] B. Scholkopf, A.J. Smola, Learning with kernels: Support vector machines, regularization, Optim. Beyond 10 (2001).

[30] A. Mazarguil, L. Oudre, N. Vayatis, Non-smooth interpolation of graph signals, Signal Process. 196 (2022) http://dx.doi.org/10.1016/j.sigpro.2022.108480.

[31] A. Venkitaraman, S. Chatterjee, P. Handel, Predicting graph signals using kernel regression where the input signal is agnostic to a graph, IEEE Trans. Signal Inf. Process. Netw. 5 (4) (2019) http://dx.doi.org/10.1109/TSIPN.2019.2936358.

[32] B. Das, E. Isufi, Learning expanding graphs for signal interpolation, in: ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP, 2022, pp. 5917–5921, http://dx.doi.org/10.1109/ICASSP43922.2022.9747156.

[33] S. Mousazadeh, I. Cohen, Embedding and function extension on directed graph, Signal Process. 111 (2015) http://dx.doi.org/10.1016/j.sigpro.2014.12.019.

[34] V.R.M. Elias, V.C. Gogineni, W.A. Martins, S. Werner, Kernel regression over graphs using random Fourier features, IEEE Trans. Signal Process. 70 (2022) http://dx.doi.org/10.1109/TSP.2022.3149134.

[35] A. Rahimi, B. Recht, Random features for large-scale kernel machines, Adv. Neural Inf. Process. Syst. 20 (2007).

[36] P. Giménez-Febrer, A. Pagès-Zamora, G.B. Giannakis, Matrix completion and extrapolation via kernel regression, IEEE Trans. Signal Process. 67 (19) (2019) http://dx.doi.org/10.1109/TSP.2019.2932875.

[37] Y. Shen, G. Leus, G.B. Giannakis, Online graph-adaptive learning with scalability and privacy, IEEE Trans. Signal Process. 67 (9) (2019) 2471–2483, http://dx.doi.org/10.1109/TSP.2019.2904922.

[38] M. Belkin, P. Niyogi, V. Sindhwani, Manifold regularization: A geometric framework for learning from labeled and unlabeled examples, J. Mach. Learn. Res. 7 (2006) 2399–2434, URL http://www.cse.msu.edu/.

[39] M. Belkin, P. Niyogi, Convergence of Laplacian eigenmaps, Adv. Neural Inf. Process. Syst. (2007) 129–136, http://dx.doi.org/10.7551/mitpress/7503.003.0021.

[40] A. Singer, From graph to manifold Laplacian: The convergence rate, Appl. Comput. Harmon. Anal. (2006) http://dx.doi.org/10.1016/j.acha.2006.03.004.

[41] C.M. Bishop, Pattern Recognition and Machine Learning (Information Science and Statistics), Springer-Verlag, Berlin, Heidelberg, 2006.

[42] N. Cressie, Statistics for Spatial Data, John Wiley & Sons, 2015.

[43] A. Menafoglio, P. Secchi, M. Dalla Rosa, A universal Kriging predictor for spatially dependent functional data of a Hilbert space, Electron. J. Stat. 7 (1) (2013) http://dx.doi.org/10.1214/13-EJS843.

[44] D. Nerini, P. Monestiez, C. Manté, Cokriging for spatial functional data, J. Multivariate Anal. 101 (2) (2010) 409–418, http://dx.doi.org/10.1016/J.JMVA.2009.03.005.

[45] A. Menafoglio, G. Petris, Kriging for Hilbert-space valued random fields: The operatorial point of view, J. Multivariate Anal. 146 (2016) 84–94, http://dx.doi.org/10.1016/J.JMVA.2015.06.012.

[46] A. Sandryhaila, J.M. Moura, Discrete signal processing on graphs, IEEE Trans. Signal Process. (2013) http://dx.doi.org/10.1109/TSP.2013.2238935.

[47] A. Sandryhaila, J.M. Moura, Discrete signal processing on graphs: Graph fourier transform, in: ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings, 2013, pp. 6167–6170, http://dx.doi.org/10.1109/ICASSP.2013.6638850.

[48] A. Sandryhaila, J.M. Moura, Discrete signal processing on graphs: Graph filters, in: ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings, 2013, pp. 6163–6166, http://dx.doi.org/10.1109/ICASSP.2013.6638849.

[49] A. Sandryhaila, J.M. Moura, Discrete signal processing on graphs: Frequency analysis, IEEE Trans. Signal Process. (2014) http://dx.doi.org/10.1109/TSP.2014.2321121.

[50] K.S. Lu, A. Ortega, Fast graph Fourier transforms based on graph symmetry and bipartition, IEEE Trans. Signal Process. 67 (18) (2019) http://dx.doi.org/10.1109/TSP.2019.2932882.

[51] V. Vaibhav, New method of bandlimited extrapolation, 2018, arXiv preprint arXiv:1804.04713.

[52] H. Zhu, C.K.I. Williams, R. Rohwer, M. Morciniec, Gaussian regression and optimal finite dimensional linear models, Neural Netw. Mach. Learn., C. Bishop (1998).

[53] T. Shi, M. Belkin, B. Yu, Data spectroscopy: Learning mixture models using eigenspaces of convolution operators, in: Proceedings of the 25th International Conference on Machine Learning, 2008, pp. 936–943.

[54] J. Liu, D. Cai, X. He, Gaussian mixture model with local consistency, in: Proceedings of the National Conference on Artificial Intelligence, Vol. 24, 2010, pp. 512–517.

[55] X. He, D. Cai, Y. Shao, H. Bao, J. Han, Laplacian regularized Gaussian mixture model for data clustering, IEEE Trans. Knowl. Data Eng. 23 (9) (2011) 1406–1418, http://dx.doi.org/10.1109/TKDE.2010.259.

[56] J. Shen, J. Bu, B. Ju, T. Jiang, H. Wu, L. Li, Refining Gaussian mixture model based on enhanced manifold learning, Neurocomputing 87 (2012) http://dx.doi.org/10.1016/j.neucom.2012.01.029.

[57] D.P. Kingma, M. Welling, Auto-encoding variational Bayes, in: Y. Bengio, Y. LeCun (Eds.), 2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14–16, 2014, Conference Track Proceedings, 2014, pp. 1–14, URL http://arxiv.org/abs/1312.6114.

[58] J. Mercer, Functions of positive and negative type, and their connection with the theory of integral equations, Proc. R. Soc. Lond. Ser. A 83 (559) (1909) 69–70, http://dx.doi.org/10.1098/rspa.1909.0075.

[59] C.K.I. Williams, M.M. Seeger, The effect of the input density distribution on kernel-based classifiers, in: Proceedings of the 17th International Conference on Machine Learning, 2000, pp. 1159–1166, http://dx.doi.org/10.1371/journal.pone.0189208, URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.18.6714.

[60] Y. LeCun, C. Cortes, MNIST handwritten digit database, 2010, AT&T Labs [Online]. Available: http://yann.lecun.com/exdb/mnist, 7.

[61] S. Lafon, Diffusion Maps and Geometric Harmonics, No. May (Ph.D. thesis), Yale University, 2004, pp. 1–87.

[62] V. Chandola, R.R. Vatsavai, D. Kumar, A. Ganguly, Analyzing big spatial and big spatiotemporal data: A case study of methods and applications, Handbook of Statist. 33 (2015) 239–258, http://dx.doi.org/10.1016/B978-0-444-63492-4.00010-1.

[63] I. Zach, RoMix github page, 2022, URL https://github.com/itayzach/RoMix/.

[64] C.J. Dsilva, R. Talmon, R.R. Coifman, I.G. Kevrekidis, Parsimonious representation of nonlinear dynamical systems through manifold learning: A chemotaxis case study, Appl. Comput. Harmon. Anal. 44 (3) (2018) http://dx.doi.org/10.1016/j.acha.2015.06.008.

[65] N. Perraudin, J. Paratte, D. Shuman, L. Martin, V. Kalofolias, P. Vandergheynst, D.K. Hammond, GSPBOX: A toolbox for signal processing on graphs, 2014, arXiv e-prints.

[66] O. Kramer, Dimensionality reduction with unsupervised nearest neighbors, Intell. Syst. Ref. Libr. 51 (2013) 21–22, http://dx.doi.org/10.1007/978-3-642-38652-7.

[67] Y. Bengio, J.F. Paiement, P. Vincent, O. Delalleau, N. Le Roux, M. Ouimet, Out-of-sample extensions for LLE, Isomap, MDS, Eigenmaps, and spectral clustering, in: Advances in Neural Information Processing Systems, 2004, pp. 1–8.

[68] J.H. Friedman, J.L. Bentley, R.A. Finkel, An algorithm for finding best matches in logarithmic expected time, ACM Trans. Math. Softw. 3 (3) (1977) 209–226, http://dx.doi.org/10.1145/355744.355745.

[69] Shuttle Radar Topography Mission (SRTM) Global, NASA shuttle radar topography mission (SRTM), in: Distributed By OpenTopography, 2013, http://dx.doi.org/10.5069/G9445JDF.

[70] T.G. Dvorkind, Y.C. Eldar, Geolocation with graph-based model fitting, in: 2019 IEEE 8th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing, CAMSAP 2019 - Proceedings, Institute of Electrical and Electronics Engineers Inc., 2019, pp. 356–360, http://dx.doi.org/10.1109/CAMSAP45676.2019.9022614.

[71] I. Jolliffe, Principal component analysis, in: M. Lovric (Ed.), International Encyclopedia of Statistical Science, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 1094–1096, http://dx.doi.org/10.1007/978-3-642-04898-2_455.

[72] S. Saitoh, Y. Sawano, Fundamental properties of RKHS, in: Theory of Reproducing Kernels and Applications, Springer Singapore, Singapore, 2016, pp. 65–160, http://dx.doi.org/10.1007/978-981-10-0530-5_2.

[73] L. Debnath, P. Mikusinski, et al., Introduction to Hilbert Spaces with Applications, Academic Press, 2005.

[74] F. Chollet, et al., Keras, 2015, URL https://keras.io.