Sparse Solution Technique in Semi-supervised Local Clustering and High Dimensional Function Approximation

by

Zhaiming Shen

(Under the direction of Ming-Jun Lai)

Abstract

The sparse solution technique, stemming from the principles of compressive sensing, holds myriad applications in both applied mathematics and data science. This dissertation studies its applications in two directions: local clustering and function approximation. Local clustering aims at extracting a local structure inside a graph without the necessity of knowing the entire graph structure. As the local structure is usually small in size compared to the entire graph, one can think of it as a compressive sensing problem where the indices of target cluster can be thought as a sparse solution to a linear system associated with the graph Laplacian. Inspired by this idea, we developed two algorithms which can find the cluster of interest efficiently and effectively. For function approximation in $C([0,1]^d)$, we propose a new approach via Kolmogorov superposition theorem (KST) based on the linear spline approximation of the K-outer function in Kolmogorov superposition representation. We achieve the optimal approximation rate as $O(\frac{1}{n^2})$, with n being the number of knots of the linear spline functions over $[0,1]$, and the approximation constant increases linearly in the dimension d. We show that there is a dense subclass in $C([0,1]^d)$ whose approximation can achieve such optimal rate, and the number of parameters needed in such approximation is at most $O(nd)$. This approach can also be extended to the numerical solution of partial differential equation such as the Poisson equation.

INDEX WORDS: Spectral Graph Theory, Compressive Sensing, Sparse Solution, Semi-supervised Learning, Data Clustering, Approximation Theory, Kolmogorov Superposition Theorem, Spline Approximation, Pivotal Location, Curse of Dimensionality

Sparse Solution Technique in Semi-supervised Local Clustering and High Dimensional Function Approximation

by

Zhaiming Shen

B.Sc, Xi'an Jiaotong-Liverpool University, 2014

B.Sc (hons), University of Liverpool, 2014

M.Phil, University of Pennsylvania, 2017

A Dissertation Submitted to the Graduate Faculty

of The University of Georgia in Partial Fulfillment

of the

Requirements for the Degree

Doctor of Philosophy

Athens, Georgia

2024

Sparse Solution Technique in Semi-supervised Local Clustering and High Dimensional Function Approximation

by

Zhaiming Shen

Approved:

Major Professor: Ming-Jun Lai

Committee: Lin Mu
Jingzhi Tie
Qing Zhang

Electronic Version Approved:

Ron Walcott
Dean of the Graduate School
The University of Georgia
May 2024

Acknowledgments

I am immensely grateful to my advisor, Ming-Jun Lai, for the unwavering guidance and support extended to me throughout the past six years. I appreciate his time spent on me through classes, research meetings, conferences, etc,. Without him, I would not have the privilege of writing this dissertation, and would not become the person I am now. I also want to thank my committee members Lin Mu, Jingzhi Tie, and Qing Zhang. Their commitment to educating me through various classes in different branches of mathematics has been instrumental in broadening my knowledge. Additionally, I am thankful for their valuable suggestions and guidance, which have significantly contributed to the completion of this dissertation.

Thanks to the Math Department at University of Georgia for continuously providing me with the support for multiple years. I would like to thank Neil Lyall, David Gay, Laura Ackerley, Lucy Barrera, Christy McDonald, all the other faculty members, staff, colleagues, and friends for helping me succeed in general as a math graduate student. Their continuous assistance has been instrumental in my academic journey, contributing to my growth and achievements. I am thankful for the enriching environment they have fostered.

Furthermore, I would like to extend my gratitude to Ming-Jun Lai, Lin Mu, Daniel Mckenzie, Weiwei Hu, Sheng Li, David Gay, Lisa Townsley, and Moyi Tian for generously dedicating their time to assist me in research discussions and job applications. Their expertise and dedication have been invaluable assets as I embark on my next academic and professional endeavors.

Finally, a special thank to my parents for consistently being by my side, offering unwavering support through both ups and downs. Their enduring confidence in me and their encouragement to believe in my capabilities beyond my own perception have been a source of strength and motivation. Their supportive presence and encouraging words have not only created a profound sense of community but also served as a wellspring of inspiration, guiding me through the challenges of my academic pursuit.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

The structure of this dissertation is as follows. In Chapter 1, we give an introduction of the preliminaries and background which are necessary for the development of this dissertation. More specifically, we introduce compressive sensing, spectral graph theory, and Kolmogorov superposition theorem (KST). In Chapter 2 and 3, we introduce two approaches based on least squares and compressive sensing for finding the local cluster respectively. In Chapter 4 and 5, we present a new function approximation scheme based on the representation of KST, and we also discuss its application to numerically solving partial differential equations.

1.1 Compressive Sensing and Sparse Solution

We call a vector *sparse* if it has few non-zero entries in comparison to its overall length. The idea of compressive sensing (also called compressed sensing/sampling), which is motivated by problems arose in signal acquisition and compression for storage, comes from solving the noisy recovery of the sparse solution with small tolerance $\epsilon > 0$:

$$\min_{\mathbf{x}\in\mathbb{R}^n} \|\mathbf{x}\|_0 \quad s.t. \quad \|\Phi\mathbf{x} - \mathbf{y}\|_2 \leq \epsilon, \tag{1.1}$$

where $\Phi \in \mathbb{R}^{m\times n}$ is called sensing matrix (usually underdetermined), $\mathbf{y} \in \mathbb{R}^n$ is called measurement vector, and the "zero quasi-norm" $\|\cdot\|_0$ counts the number of nonzero components in a vector. The goal of (1.1) is to find a s-sparse solution $\mathbf{x} \in \mathbb{R}^n$ under some constraint. In the case of $\epsilon = 0$, we have the exact (noiseless) sparse recovery problem

$$\min_{\mathbf{x}\in\mathbb{R}^n} \|\mathbf{x}\|_0 \quad s.t. \quad \Phi\mathbf{x} = \mathbf{y}. \tag{1.2}$$

A related problem is

$$\arg\min_{\mathbf{x}\in\mathbb{R}^n} \|\Phi\mathbf{x}-\mathbf{y}\|_2 \quad s.t. \quad \|\mathbf{x}\|_0 \leq s. \tag{1.3}$$

However, these problems involved in the $\|\cdot\|_0$ is NP-hard. To solve it practically, people often consider their convex relaxations:

$$\min_{\mathbf{x}\in\mathbb{R}^n} \|\mathbf{x}\|_1 \quad s.t. \quad \|\Phi\mathbf{x}-\mathbf{y}\|_2 \leq \epsilon. \tag{1.4}$$

In the noiseless case, we have

$$\min_{\mathbf{x}\in\mathbb{R}^n} \|\mathbf{x}\|_1 \quad s.t. \quad \Phi\mathbf{x}=\mathbf{y} \tag{1.5}$$

and

$$\arg\min_{\mathbf{x}\in\mathbb{R}^n} \|\Phi\mathbf{x}-\mathbf{y}\|_2 \quad s.t. \quad \|\mathbf{x}\|_1 \leq s. \tag{1.6}$$

As in the literature, problem (1.5) is often referred to as basis pursuit [17]. The origin of compressive sensing can be traced back to the realization that in the noiseless case, if Φ is a random matrix such that $\mathbf{y}=\Phi\mathbf{x}^*$ with $\|\mathbf{x}^*\|_0 = s$ such that $m = O(s\log(n/s))$, then (1.5) and (1.6) has a unique solution, and this solution coincides with $\mathbf{x}^*$, which is the solution to (1.2) and (1.3). Among all the contributors, Donoho [29] and Candès, Romberg, Tao [16] are widely recognized as the first to explicitly establish this connection.

1.1.1 Restricted Isometry Property

Now let us present two crucial concepts which are often employed in the analysis of sensing matrices and compressive sensing algorithms. The first one is called Restricted Isometry Property (RIP).

Definition 1.1.1 (Restricted Isometry Property)**.** *Let $0 < s < m$ be an integer, and sensing matrix $\Phi \in \mathbb{R}^{m\times n}$. Suppose there exists a constant $\delta_s > 0$ such that*

$$(1-\delta_s)\|\mathbf{x}\|_2^2 \leq \|\Phi\mathbf{x}\|_2^2 \leq (1+\delta_s)\|\mathbf{x}\|_2^2 \tag{1.7}$$

for all $\mathbf{x} \in \mathbb{R}^n$ *with* $\|\mathbf{x}\|_0 \leq s$. *Then the matrix* Φ *is said to have the Restricted Isometry Property (RIP) of order* s. *The smallest constant* $\delta_s(\Phi)$ *which makes (1.7) hold is called the Restricted Isometry Constant (RIC) of* Φ.

With this, it is easy to see that if $\delta_{2s} < 1$, then the solution of (1.2) is unique. Indeed, if there are two s-sparse solutions $\mathbf{x}_1$ and $\mathbf{x}_2$ such that $\Phi\mathbf{x}_1 = \mathbf{y}$ and $\Phi\mathbf{x}_2 = \mathbf{y}$, i.e., $\|\mathbf{x}_1 - \mathbf{x}_2\|_0 \leq 2s$ such that $\Phi(\mathbf{x}_1 - \mathbf{x}_2) = 0$, then

$$(1 - \delta_{2s})\|\mathbf{x}_1 - \mathbf{x}_2\|_2^2 \leq \|\Phi(\mathbf{x}_1 - \mathbf{x}_2)\|_2^2 = 0.$$

It follows that $\|\mathbf{x}_1 - \mathbf{x}_2\|_2 = 0$ when $\delta_{2s} < 1$. That is, the solution is unique.

The RIC can also defined via the following lemma. Its proof is straightforward, which is included in [80] and [40].

Lemma 1.1.1. *Let* $0 < s < m$ *be an integer,* Φ_T *be a submatrix of* $\Phi \in \mathbb{R}^{m\times n}$ *which consists of columns of* Φ *whose columns indices are in* $T \subset \{1, 2, \cdots, n\}$. *Then*

$$\delta_s = \max_{\#(T)\leq s} \|I_m - \Phi_T^\top \Phi_T\|_2, \tag{1.8}$$

where I_m *is the identity matrix of size* $m \times m$.

The following lemma gives a simple but useful bound for the spectrum of $\Phi_T^\top \Phi_T$. We leave its proof to interested readers.

Lemma 1.1.2. *Suppose that the sensing matrix* Φ *satisfies the RIP with RIC* $\delta_s < 1$. *Then for any index set* T *with* $\#(T) \leq s$, *the symmetric matrix* $\Phi_T^\top \Phi_T$ *is positive definite with the largest eigenvalue* $\lambda_1 \leq 1 + \delta_s$ *and smallest eigenvalue* $\lambda_s \geq 1 - \delta_s$.

Practically, one wants to establish the uniqueness of solution to the ℓ_1 noiseless sparse vector recovery problem (1.5), so that any solution found via algorithmic approaches can be guaranteed to be the desired solution. If the sensing matrix Φ has a good structure, i.e., the RIC of δ_s is small, one can establish the following uniqueness result.

Theorem 1.1.1 (Cai and Zhang, 2013 [14]). *Suppose* $\Phi \in \mathbb{R}^{m\times n}$ *and* $\mathbf{y} = \Phi\mathbf{x}^*$ *such that* $\|\mathbf{x}^*\|_0 = s$. *Then* $\mathbf{x}^*$ *is the unique solution to (1.5) provided* $\delta_s(\Phi) = 1/3$. *Moreover, this bound is sharp in the sense that there exists* Φ *such that* $\delta_s(\Phi) \geq 1/3$ *and s-sparse vector* $\mathbf{x}^* \in \mathbb{R}^n$ *such that* $\mathbf{x}^*$ *is not the unique solution to (1.5).*

Moreover, there are conditions imposed on δ_{2s}, δ_{3s}, and δ_{4s} to guarantee to uniqueness of solution to (1.5). For example:

Theorem 1.1.2 (Candes and Tao, 2005 [15]). *Let* $\mathbf{x}^* \in \mathbb{R}^n$ *with sparsity* $\|\mathbf{x}^*\|_0 = s$ *such that* $\Phi\mathbf{x}^* = \mathbf{y}$. *Suppose* $\Phi \in \mathbb{R}^{m\times n}$ *satisfies*

$$\delta_{3s} + 3\delta_{4s} < 2.$$

Then $\mathbf{x}^*$ *is the unique solution to (1.5).*

Theorem 1.1.3 (Foucart and Lai, 2009 [41]). *Suppose that* $s \geq 1$ *such that*

$$\delta_{2s} < \frac{2}{3+\sqrt{2}} \approx 0.4531,$$

and let $\mathbf{x}^* \in \mathbb{R}^n$ *be vector with* $\|\mathbf{x}^*\|_0 \leq s$ *and* $\Phi\mathbf{x}^* = \mathbf{y}$. *Then* $\mathbf{x}^*$ *is the unique solution to (1.5).*

One notable feature of the RIP condition is that a random matrix is likely to satisfy RIP. In fact, it has been demonstrated that matrices whose entries are drawn from Gaussian, sub-Gaussian, uniform, or Bernoulli distributions satisfy the RIP condition with overwhelming probability. We leave the discussion of what kind of matrices will satisfy the RIP condition to Chapter 8 in the book [80].

1.1.2 Mutual Coherence

The second crucial concept which characterizes the correlation between pairs of columns in the sensing matrix is called mutual coherence.

Definition 1.1.2 (Mutual Coherence)**.** *Let $\Phi = [\phi_1, \cdots, \phi_n] \in \mathbb{R}^{m\times n}$, where ϕ_i is the i-th column of Φ. The mutual coherence is defined as*

$$\mu(\Phi) := \max_{\substack{i,j=1,\cdots,n \\ i\neq j}} \frac{|\phi_i^\top \phi_j|}{\|\phi_i\|_2 \|\phi_j\|_2}. \tag{1.9}$$

Note that $\mu(\Phi) \in [0, 1]$. In many occasions, the mutual coherence is also defined as

$$\mu(\Phi) := \max_{\substack{i,j=1,\cdots,n \\ i\neq j}} |\phi_i^\top \phi_j| \tag{1.10}$$

if assuming all the columns ϕ_i are normalized. One can show that the mutual coherence satisfies the following lower bound if the matrix is of full rank.

Lemma 1.1.3. *Assume that $\Phi \in \mathbb{R}^{m\times n}$ is of full rank. Then $\mu(\Phi) \geq \sqrt{\frac{n-m}{m(n-1)}}$. In particular, if $n \geq 2m$, then $\mu(\Phi) \geq (2m)^{-1/2}$.*

Similar to Lemma 1.1.2, it is straightforward to show that the singular values of Φ_T are bounded below and above based on the value of mutual coherence. We omit its proof and refer interested readers to [80].

Lemma 1.1.4. *Let $\mu = \mu(\Phi)$ and $s < 1/\mu + 1$. For any $T \subset \{1, \cdots, n\}$ with $\#(T) \leq s$ and Φ_T the matrix consisting of the s columns of Φ with column indices in T, the singular values of Φ_T are bounded above by $(1 + \mu(s-1))^{1/2}$ and below by $(1 - \mu(s-1))^{1/2}$.*

Given a sensing matrix Φ satisfying RIP, its mutual coherence and RIC δ_s have the following relation:

Lemma 1.1.5 (Rauhut, 2010 [112])**.** *For any sensing matrix Φ,*

$$\delta_s \leq (s-1)\mu(\Phi),$$

where δ_s is the RIC of Φ.

1.1.3 Greedy Algorithms and Perturbation Analysis

Before the idea of compressive sensing was introduced, (1.6) is often solved via techniques such as matching pursuit [94], LASSO [132], and basis pursuit [17]. There were theoretical results describing when these algorithms recovered sparse solutions, but the required type and number of measurements were sub-optimal and subsequently greatly improved by compressive sensing. More recently, the algorithmic approaches based on the idea of compressive sensing were introduced. These mainly include two families of approaches: thresholding algorithms (for example, iterative hard thresholding [10]) and greedy algorithms (for example, orthogonal matching pursuit [133]). We present the Orthogonal Matching Pursuit (OMP) algorithm as Algorithm 1, it is sometimes also referred to as Orthogonal Greedy Algorithm (OGA) in the literature.

Algorithm 1: Orthogonal Matching Pursuit (OMP)

Data: Sensing matrix $\Phi = [\phi_1, \cdots, \phi_n]$, measurement vector $\mathbf{y}$, sparsity parameter L, tolerance ϵ.

Result: The estimated signal $\mathbf{x}^{\#}$.

Initialization: $S^{(0)} = \emptyset$, $\mathbf{r}^{(0)} = \mathbf{y}$;

for $k = 1, \cdots, L$ **do**

  $i_k = \arg\max_{1 \le i \le n} \{|\phi_i^\top \mathbf{r}^{(k-1)}|\}$;

  $S^{(k)} = S^{(k-1)} \cup \{i_k\}$;

  $\mathbf{x}^{(k)} = \arg\min_{\mathbf{z} \in \mathbb{R}^n} \{\|\mathbf{y} - \Phi\mathbf{z}\|_2 : supp(\mathbf{z}) \subset S^{(k)}\}$;

  $\mathbf{r}^{(k)} = \mathbf{y} - \Phi\mathbf{x}^{(k)}$ and $\mathbf{x}^{\#} = \mathbf{x}^{(k)}$;

  if $\|\mathbf{r}^{(k)}\|_2 < \epsilon$ **then**

    break

  end

end

There are also other variants of OMP, such as Generalized Orthogonal Matching Pursuit [140], Regularized Orthogonal Matching Pursuit [105], Iterative Least Squares Orthogonal Matching Pursuit [109], Quasi-Orthogonal Matching Pursuit [75, 37]. Later on, we will be applying a more sophisticated algorithm, named *Subspace Pursuit* [21], in Chapter 2 and 3 for local clustering task. The notable aspect of the Subspace Pursuit lies in its ability to dynamically update the selection of column indices after each iteration, whereas Orthogonal Matching Pursuit (OMP) maintains the selection of indices from the previous iteration without alteration. We present the algorithm of Subspace Pursuit as Algorithm 2.

Algorithm 2: Subspace Pursuit

Data: Sensing matrix Φ, measurement vector $\mathbf{y}$, sparsity parameter s, number of iterations K.

Result: The estimated signal $\mathbf{x}^{\#}$.

Initialization: $S^{(0)} = \mathcal{L}_s(\Phi^\top \mathbf{y})$, $\mathbf{x}^{(0)} = \arg\min_{\mathbf{z}\in\mathbb{R}^n}\{\|\mathbf{y}-\Phi\mathbf{z}\|_2 : supp(\mathbf{z}) \subset S^{(0)}\}$, $\mathbf{r}^{(0)} = \mathbf{y} - \Phi\mathbf{x}^{(0)}$;

for $k = 1, \cdots, K$ **do**

  $\hat{S}^{(k)} = S^{(k-1)} \cup \mathcal{L}_s(\Phi^\top \mathbf{r}^{(k-1)})$;

  $\mathbf{u} = \arg\min_{\mathbf{z}\in\mathbb{R}^n}\{\|\mathbf{y}-\Phi\mathbf{z}\|_2 : supp(\mathbf{z}) \subset \hat{S}^{(k)}\}$;

  $S^{(k)} = \mathcal{L}_s(\mathbf{u})$ and $\mathbf{x}^{(k)} = \mathcal{H}_s(\mathbf{u})$;

  $\mathbf{r}^{(k)} = \mathbf{y} - \Phi\mathbf{x}^{(k)}$;

end

Let $\mathbf{x}^{\#}_{\mathcal{S}^{(K)}} = (\Phi^\top_{S^{(K)}}\Phi_{S^{(K)}})^{-1}\Phi^\top_{S^{(K)}}\mathbf{y}$ and $\mathbf{x}^{\#}_{(\mathcal{S}^{(K)})^c} = \mathbf{0}$.

The $\mathcal{L}(\cdot)$ and $\mathcal{H}(\cdot)$ are threholding operators defined as

$$\mathcal{L}_s(\mathbf{v}) := \{i \in [n] : v_i \text{ among } s \text{ largest-in-magnitude entries in } \mathbf{v}\},$$

$$\mathcal{H}_s(\mathbf{v})_i := \begin{cases} v_i & \text{if} \quad i \in \mathcal{L}_s(\mathbf{v}), \\ 0 & \text{otherwise.} \end{cases}$$

Many proofs of correctness for compressive sensing algorithms rely on the Restricted Isometry Property (RIP). Here we list a few of them. We omit their proofs and refer interested readers to [80].

Theorem 1.1.4 (Mo, 2015 [99]). *If $\delta_{s+1} < \frac{1}{\sqrt{s+1}}$, the exact recovery of the s-sparse signal can be guaranteed by using Algorithm 1 in s iterations.*

Theorem 1.1.5 (Wen, Zhou, Liu, Lai, and Tang, 2019 [142]). *Let $\mathbf{v} := \mathbf{y} - \Phi\mathbf{x}_b$. Suppose that $\mathbf{v}$ satisfies $\|\mathbf{v}\|_2 \leq \epsilon$ and Φ satisfies the RIP with δ_{s+1} satisfying*

$$\delta_{s+1} < \frac{1}{\sqrt{s+1}}. \tag{1.11}$$

Then Algorithm 1 with stopping criterion $\|\mathbf{r}^{(k)}\|_2 \leq \epsilon$ exactly recovers the support of the sparse signal $\mathbf{x}_b = [x_1, \cdots, x_n]^\top$ in s iterations provided

$$\min_{x_i \neq 0}\{|x_i|\} > \frac{2\epsilon}{1 - \sqrt{s+1}\delta_{s+1}}. \tag{1.12}$$

Moreover, the recovery error can be bounded by

$$\|\mathbf{x}_b - \mathbf{x}^{\#}\| \leq \epsilon, \tag{1.13}$$

where $\mathbf{x}^{\#}$ is the output from Algorithm 1.

One of the reasons behind the remarkable usefulness of compressive sensing lies in its robustness against errors, including both additive and multiplicative types. More precisely, suppose we know $\mathbf{y} = \Phi\mathbf{x}^*$ where $\mathbf{y}$ is the exact measurement of the acquired signal and Φ is the exact measurement of the sensing matrix. However, we may only be able to access to the noisy version $\tilde{\mathbf{y}} = \mathbf{y} + \Delta\mathbf{y}$ and $\tilde{\Phi} = \Phi + \Delta\Phi$. Can we expect to approximate the solution $\mathbf{x}^*$ well from the noisy measurements $\tilde{\mathbf{y}}$ and $\tilde{\Phi}$. This question is affirmatively addressed by several authors, starting with the work of [54]. For Subspace Pursuit algorithm, we have the following result:

Theorem 1.1.6 (Li, 2016 [82]). *Let $\mathbf{x}^*$, $\mathbf{y}$, $\tilde{y}$, Φ, $\tilde{\Phi}$ be as defined above, and for any $t \in [n]$, let $\delta_s := \delta_s(\tilde{\Phi})$. Suppose that $\|\mathbf{x}^*\|_0 \leq s$. Define the following constants:*

$$\epsilon_{\mathbf{y}} := \|\mathbf{\Delta y}\|_2 / \|\mathbf{y}\|_2 \quad \textit{and} \quad \epsilon_{\Phi}^s := \|\Delta\Phi\|_2^{(s)} / \|\Phi\|_2^{(s)}$$

where $\|M\|_2^{(s)} := \max\{\|M_S\|_2 : S \subset [n], \#(S) = s\}$ *for any matrix* M*. Define further:*

$$\rho := \frac{\sqrt{2\delta_{3s}^2(1+\delta_{3s}^2)}}{1-\delta_{3s}^2} \quad \textit{and} \quad \tau := \frac{(\sqrt{2}+2)\delta_{3s}}{\sqrt{1-\delta_{3s}^2}}(1-\delta_{3s})(1-\rho) + \frac{2\sqrt{2}+1}{(1-\delta_{3s})(1-\rho)}.$$

Assume that $\delta_{3s} < 0.4859$ *and let* $\mathbf{x}^{(m)}$ *be the output of Algorithm 2 after* m *iterations. Then:*

$$\frac{\|\mathbf{x}^* - \mathbf{x}^{(m)}\|_2}{\|\mathbf{x}^*\|_2} \leq \rho^m + \tau\frac{\sqrt{1+\delta_s}}{1-\epsilon_\Phi^s}(\epsilon_\Phi^s + \epsilon_{\mathbf{y}}).$$

It is also worthwhile to mention the following result which quantifies the effect of perturbation on the RIC:

Theorem 1.1.7 (Herman and Strohmer, 2010 [54])**.** *Suppose that* $\tilde{\Phi} = \Phi + \Delta\Phi$*. Let* $\tilde{\delta}_s$ *and* δ_s *be the restricted isometry constants for* $\tilde{\Phi}$ *and* Φ *respectively. Then*

$$\tilde{\delta}_s \leq (1+\delta_s)(1+\epsilon_\Phi^s)^2 - 1.$$

1.2 Spectral Graph Theory and Clustering

In this dissertation, we will apply the idea of compressive sensing and sparse solution to study the local community or clustering structure in graphs. We adapt the standard representation of a graph $G = (V, E)$, where V is the set of nodes and E is the set of edges. For convenience, in the case that the graph is finite, we identify the set V with $[n] = \{1, \cdots, n\}$. We use $A = (a_{ij})_{i,j\in[n]} \in \{0,1\}^{n\times n}$ to denote the adjacency matrix, where $a_{ij} = 1$ if and only if there is an edge connecting nodes i and j, otherwise $a_{ij} = 0$. Note that the notion of adjacency matrix can be extended to weighted graph, in which case $A = (a_{ij})_{i,j\in[n]} \in \mathbb{R}^{n\times n}$. Furthermore, we use $d_i := \sum_{j=1}^n A_{ij}$ to denote the degree for the i-th node, $i = 1, \cdots, n$, and the matrix $D = diag(d_1, \cdots, d_n) \in \mathbb{R}^{n\times n}$ to denote the matrix whose diagonal entries are $d_1, \cdots, d_n$.

1.2.1 Graph Laplacians

Let us first introduce the Laplacians of graph, their properties and their connection to graph clustering.

Definition 1.2.1 (Graph Laplacians)**.** *The unnormalized (combinatorial) Laplacian is defined as* $L_{com} := D - A$*. The normalized, symmetric Laplacian is defined as* $L_{sym} := I - D^{-1/2}AD^{-1/2}$*. The random walk Laplacian is defined as* $L_{rw} := I - D^{-1}A$*.*

For matrices L_{com}, L_{sym}, L_{rw}, we summarize the following important properties in Lemma 1.2.1 and Lemma 1.2.2. Their proofs can be found in proposition 1 and 3 in [90].

Lemma 1.2.1 (Properties of L_{com})**.** *The matrix* L_{com} *satisfies the following properties:*

- *for every* $\mathbf{x} \in \mathbb{R}^n$*, we have* $\mathbf{x}^\top L_{com}\mathbf{x} = \frac{1}{2}\sum_{i,j=1}^{n} a_{ij}(x_i - x_j)^2$*.*
- L_{com} *is symmetric and positive semi-definite.*
- *The smallest eigenvalue of* L_{com} *is* $\lambda_1 = 0$*, the corresponding eigenvector is the constant one vector* $\mathbf{1}$*.*
- L_{com} *has* n *non-negative, real-valued eigenvalues* $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$*.*

Lemma 1.2.2 (Properties of L_{sym} and L_{rw})**.** *The matrix* L_{sym} *and* L_{rw} *satisfy the following properties:*

- *for every* $\mathbf{x} \in \mathbb{R}^n$*, we have* $\mathbf{x}^\top L\mathbf{x} = \frac{1}{2}\sum_{i,j=1}^{n} a_{ij}\left(\frac{x_i}{\sqrt{d_i}} - \frac{x_j}{\sqrt{d_j}}\right)^2$*.*
- λ *is an eigenvalue of* L_{sym} *with eigenvector* u *if and only if* λ *is an eigenvalue of* L_{rw} *with eigenvector* $w = D^{1/2}u$*.*
- 0 *is an eigenvalue of* L_{rw} *with constant vector* $\mathbf{1}$ *as eigenvector.* 0 *is an eigenvalue of* L_{sym} *with eigenvector* $D^{1/2}\mathbf{1}$*.*
- L_{sym} *and* L_{rw} *are positive semi-definite and have* n *non-negative, real-valued eigenvalues* $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$*.*

The following result (see proposition 4 in [90] for its proof) serves as the foundation for developing our local clustering algorithms in Chapter 2.

Lemma 1.2.3 (Number of connected components and spectra of L_{rw} and L_{sym})**.** *Let G be an undirected graph with non-negative weights. Then the multiplicity k of the eigenvalue 0 of both L_{rw} and L_{sym} equals to the number of connected components $C_1, \cdots, C_k$ in the graph. For L_{rw}, the eigenspace of 0 is spanned by the indicator vectors of $\mathbf{1}_{C_1}, \cdots, \mathbf{1}_{C_k}$ of those components. For L_{sym}, the eigenspace of 0 is spanned by the vectors of $D^{1/2}\mathbf{1}_{C_1}, \cdots, D^{1/2}\mathbf{1}_{C_k}$.*

1.2.2 Spectral Clustering

The graph Laplacian plays a key role in spectral clustering algorithm, which is arguably the most well-known approach to partitional clustering. Spectral clustering debuted in [39] with pioneering work on the two-cluster case, later gaining prominence in the realm of k-cluster analysis through the influential contributions in [119, 106]. We refer interested readers to the survey article [90], which gives an excellent explanation of spectral clustering. There are many variants of spectral clustering algorithm since its debut, let us include the version which makes use of the random walk Laplacian in Algorithm 3.

Algorithm 3: Spectral Clustering

Data: Adjacency matrix $A \in \mathbb{R}^{n\times n}$, number of clusters k.

Result: Clusters $C_1, \cdots, C_k$.

1 Compute the random walk normalized Laplacian matrix $L_{rw} \in \mathbb{R}^{n\times n}$;

2 Find the k eigenvectors $\mathbf{u}_1, \cdots, \mathbf{u}_k$ corresponding to the k smallest eigenvalues of L_{rw};

3 Let $U \in \mathbb{R}^{n\times k}$ be the matrix consisting of columns $\mathbf{u}_1, \cdots, \mathbf{u}_k$. Let $\mathbf{r}_i \in \mathbb{R}^k$, for $i = 1, \cdots, n$, denote the i-th row of U;

4 Cluster the points $\{\mathbf{r_1}, \cdots, \mathbf{r}_n\} \subset \mathbb{R}^k$ into k clusters $\tilde{C}_1, \cdots, \tilde{C}_k$ using k-means;

5 Build the desired clusters $C_1, \cdots, C_k$ via $C_a = \{i : \mathbf{r}_i \in \tilde{C}_a\}$ for $i = 1, \cdots, n$;

There are also other ways of clustering a graph, let us give a very broad overview of various graph clustering approaches in the literature, with the focus on (semi-supervised) local clustering.

1.2.3 Overview of Graph Clustering and Local Clustering

Traditional graph clustering problem assumes the underlying data structure as a graph where data points are the nodes and the connections between data points are the edges. It assigns each node into a unique cluster, assuming there are no multi-class assignments. For nodes with high connection density, they are considered in the same cluster, and for nodes with low connection density, they are considered in different clusters. Since the task is to learn the clustering patterns by investigating the underlying graph structure, it is an unsupervised learning task. Many unsupervised graph clustering algorithms have been developed through decades. For example, spectral clustering [106], which is based on the eigen-decomposition of Laplacian matrices of either weighted or unweighted graphs. Based on this, many variants of spectral clustering algorithms have been proposed, such as [147] and [57]. Another category is the graph partition based method such as finding the optimal cut [27, 28]. It is worthy noting that spectral clustering and graph partition have the same essence [90].

Spectral clustering has become one of the popular modern clustering algorithms since it enjoys the advantage of exploring the intrinsic data structures. It is simple to implement, and it often outperforms the traditional algorithm such as k-means. However, one of the main drawbacks of spectral clustering is its high computational cost, so it is usually not applicable to large datasets. Meanwhile, the spectral clustering method does not perform well on the auxiliary graphs which are generated from certain shapes of numerical data, e.g., elongated band shape data and moon shape data. In addition, many other clustering methods have been developed, such as the low rank and sparse representations based methods [86, 58], deep embedding based methods [144], and graph neural network based methods [59, 134]. Besides the unsupervised way, some semi-supervised graph clustering methods have also been proposed [66, 62, 113].

These clustering algorithms, whether unsupervised or semi-supervised, are all global clustering algorithms, which means that the algorithms output all the clusters simultaneously. However, it is often to people's interests in only finding a single target cluster which contains

the given labels, without worried too much about how the remaining part of graph will be clustered. Such an idea is very useful in detecting small-scale structure in a large-scale graph. This type of problem is referred to as *local clustering*. This concept was initially introduced in the computer science literature in [124, 125]. Their motivation was to extend clustering techniques to large graphs, denoted for which conventional partitional clustering techniques were impractical. Their proposed algorithm finds a target cluster in almost linear time and is a motivating example of the diffusion-based local clustering algorithm. Further examples of this type of algorithm include the algorithms proposed in [5, 18, 63]. Another different family of approaches to this problem are the local spectral methods. Its idea was first proposed in [92] to find a local analogue of the second eigenvector of the Laplacian, which resembles spectral clustering. In the work of [51, 83, 84, 120], the approach is to first subsample a graph that is much smaller in size compared to the original graph, but very likely to contain the target cluster, then apply spectral methods to extract the target cluster from this subsampled graph. It is worth of pointing out that [42] have kindly put several methods of local graph clustering into software, including [5, 43, 81, 135, 139]. A related problem, as discussed in the statistics literature, pertains to the problem of *cluster extraction*. This was motivated in [148] by the task of distinguishing a cluster from a background of vertices that do not belong in any cluster. Such method is further developed in [143]. We do not discuss further.

More recently, new two-stage approaches based on making the cut of graph and compressive sensing are proposed in [70, 77] where they took a novel perspective by considering the way of finding the optimal cut as an improvement from an initial cut via finding a sparse solution to a linear system. Their results were further improved by [117], where the cut was taken to be the entire graph so that it excluded possibility of missing any target vertices from the initial cut. One notable feature of any local clustering algorithm is that it can, in principle, be iterated to yield a partition over the entire vertices in the graph. That is, if one proceeds by removing all clusters previously found and then extracting the next cluster,

one ends up with a partition of the entire graph. Let us call this feature "one cluster at a time". On the other hand, retaining the clusters which being in the previous iterations permits the presence of overlapping clusters. Therefore, local clustering encompasses both partitional and overlapping clustering tasks. In Chapter 2 and 3, we will give a more detailed discussion of the approaches which are proposed in [77] and [117] for semi-supervised local clustering task.

1.3 The Issue of Curse of Dimensionality

Let us shift our attention now to briefly discussing the well-known bottleneck inherent in the computation of high-dimensional learning and approximation — commonly referred to as the *curse of dimensionality*.

Recently, deep learning algorithms have shown a great success in many fronts of research, from image analysis, audio analysis, biological data analysis, to name a few. Incredibly, after a deep learning training of thousands of images, a computer can tell if a given image is a cat, or a dog, or neither of them with very reasonable accuracy. In addition, there are plenty of successful stories such that deep learning algorithms can sharpen, denoise, enhance an image after an intensive training. See, e.g. [47] and [95]. The 3D structure of a DNA can be predicted very accurately by using the DL approach. The main ingredient in DL algorithms is the neural network approximation based on ReLU functions. We refer to [22] and [25] for detailed explanation of the neural network approximation in deep learning algorithms.

Learning a multi-dimensional data set is like approximating a multivariate function. The computation of a good approximation suffers from the curse of dimension. For example, suppose that $f \in C([0,1]^d)$ with $d \gg 1$. One usually uses Weierstrass theorem to have a polynomial P_f of degree n such that

$$\|f - P_f\|_\infty \leq \epsilon$$

for any given tolerance $\epsilon > 0$. As the dimension of polynomial space $= \binom{n+d}{n} \approx n^d$ when $n > d$, one will need at least $N = O(n^d)$ data points in $[0,1]^d$ to distinguish different

polynomials in $\mathbb{P}_n$ and hence, to determine this P_f. Notice that many good approximation schemes enable P_f to approximate f at the rate of $O((1/n)^m)$ if f is of m times differentiable. In terms of the number N of data points which should be greater than or equal to the dimension of polynomial space $\mathbb{P}_n$, i.e. $N \geq n^d$, the order of approximation is $O(1/(N^{1/d})^m)$. When $d \gg 1$ is bigger, the order of approximation is less. This phenomenon is the so-called curse of dimension. Sometimes, such a computation is also called intractable. Similarly, if one uses a tensor product of B-spline functions to approximate $f \in C([0,1]^d)$, one needs to subdivide the $[0,1]^d$ into n^d small subcubes by hyperplanes parallel to the axis planes. As over each small subcube, the spline approximation S_f of f is a polynomial of degree k, e.g., $k = 3$ if the tensor product of cubic splines are used. Even with the smoothness, one needs $N = O(k^d)$ data points and function values at these data points in order to determine a polynomial piece of S_f over each subcube. Hence, over all subcubes, one needs $O(n^d k^d)$. It is known that the order of approximation of S_f is $O(1/n^{k+1})$ if f is of $k+1$ times coordinatewise differentiable. In terms of $N = O(n^d k^d)$ points over $[0,1]^d$, the approximation order of S_f will be $O(k^{k+1}/N^{(k+1)/d})$. More precisely, in [26], the researchers showed that the approximation order $O(1/N^{1/d})$ can not be improved for smooth functions in Sobolev space $W^{k,p}$ with L_p norm ≤ 1. In other words, the approximation problem by using multivariate polynomials or by tensor product B-splines is intractable.

Furthermore, many researchers have worked on using ridge functions, neural networks, and ReLU activation functions to approximate multidimensional functions. The orders of all these approximations in L_2 and in L_∞ norms show the curse of dimensionality. See [93, 110, 108, 7] for detailed statements and proofs. That is, the approximation problem by using the neural networks is intractable. However, there is a way to obtain the dimension independent approximate rate as explained by Barron in [8]. Let $\Gamma_{B,C}$ be the class of functions f defined over $B = \{\mathbf{x} \in \mathbb{R}^d, \|\mathbf{x}\| \leq 1\}$ such that $C_f \leq C$, where C_f is defined as follows:

$$C_f = \int_{\mathbb{R}^d} |\omega||\widetilde{f}(\omega)|d\omega$$

with $|\omega| = (\omega \cdot \omega)^{1/2}$ and $\widetilde{f}$ is the Fourier transform of f.

Theorem 1.3.1 (Barron, 1993). *For every function f in $\Gamma_{B,C}$, every sigmoidal function ϕ, every probability measure μ, and every $n \geq 1$, there exists a linear combination of sigmoidal functions $f_n(x)$, a shallow neural network, such that*

$$\int_B (f(\mathbf{x}) - f_n(\mathbf{x}))^2 d\mu(\mathbf{x}) \leq \frac{(2C)^2}{n}.$$

The coefficients of the linear combination in f_n may be restricted to satisfy $\sum_{k=1}^{n} |c_k| \leq 2C$, and $c_0 = f(0)$.

It is worthy of noting that although the approximation rate is independent of the dimension in the L_2 norm sense, the constant C can be exponentially large in the worst case scenario as pointed out in [8]. This work leads to many recent studies on the properties and structures of Barron space $\Gamma_{B,C}$ and its extensions, e.g. the spectral Barron spaces and the generalization using ReLU or other more advanced activation functions instead of sigmodal functions above, e.g., [64, 31, 32, 34, 33, 121, 122], and the references therein. More recently, the super approximation power is introduced in [118] which uses the floor function, exponential function, step function, and their compositions as the activation function and can achieve the exponential approximation rate.

In Chapter 4 and 5, we turn our attention to Kolmogorov superposition theorem [65, 88] and will see how it plays a role in the study of the rate of approximation for neural network computation in deep learning algorithms, and how it can break the curse of dimension when approximating high dimensional functions over $\mathbb{R}^d$ for $d \geq 2$.

Chapter 2

Semi-supervised Local Clustering via Least Squares

Local clustering aims at extracting a local structure inside a graph without the necessity of knowing the entire graph structure. As the local structure is usually small in size compared to the entire graph, one can think of it as a compressive sensing problem where the indices of target cluster can be thought as a sparse solution to a linear system. For convenience, let us assume the target cluster is the first cluster C_1 for the rest of discussion. The semi-supervised local clustering problem we are interested in solving is:

Suppose $G = (V, E)$ is a graph with underlying cluster $C_1, \cdots, C_k$ where $V = \cup_{i=1}^{n} C_i$, $C_i \cap C_j = \emptyset$ for $1 \leq i, j \leq k$, $i \neq j$. Given a set of labeled vertices $\Gamma \subset C_1$, which we call them seeds, assuming the size of Γ is small relative to the size of C_1. The goal is to extract all the vertices in the target cluster C_1.

Based on the pioneering work [70] of local clustering via compressive sensing, we developed two approaches [77] and [117] with sequentially better performance. The major results of these two works are summarized in this and next chapters. For the rest of discussion in these two chapters, we will focus on undirect simple but weighted graphs.

2.1 Preliminaries and Notations

Let us introduce some more notations which we will use later. Suppose for the moment we have information about structure of the underlying clusters for each vertex, then it is useful to write G as a union of two edge-disjoint subgraphs $G = G^{in} \cup G^{out}$ where $G^{in} = (V, E^{in})$ consists of only intra-connection edges, and $G^{out} = (V, E^{out})$ consists of only inter-connection edges. We will use d_i^{in} to denote the degree of vertex i in the subgraph G^{in}, and d_i^{out} to denote

the degree of vertex i in the subgraph G^{out}. We will also use A^{in} and L^{in} to denote the adjacency matrix and graph Laplacian associated with G^{in}, and A^{out} and L^{out} to denote the adjacency matrix and graph Laplacian associated with G^{out}. Note that these notations are just for convenience for the analysis in the next section, in reality we will have no assurance about which cluster each individual vertex belongs to, so we will have no access to A^{in} and L^{in}. It is also worthwhile to point out that $A = A^{in} + A^{out}$ but $L \neq L^{in} + L^{out}$ in general. Furthermore, we will use $|L|$ or $|\mathbf{y}|$ to denote the matrix or vector where each its entry is replaced by the absolute value, and we will use $|V|$ to denote the size of V whenever V is a set. In the later sections, we will use L and L^{in} to indicate L_{rw} and L^{in}_{rw} respectively, and use L_C and L^{in}_C to denote the submatrices of L and L^{in} with column indices subset $C \subset V = [n]$ respectively. For convenience, let us formulate the notations being used through Chapter 2 and Chapter 3 into Table 2.1.

Table 2.1: Table of Notations

Symbols	Interpretations
G	A general graph of interest
$\|G\|$	Size of G
V	Set of vertices of graph G
$\|V\|$	Size of V
C_1	Target Cluster
Γ	Set of Seeds
T	Removal set from V
E	Set of edges of graph G
E^{in}	Subset of E which consists only intra-connection edges
E^{out}	Subset of E which consists only inter-connection edges
G^{in}	Subgraph of G on V with edge set E^{in}
G^{out}	Subgraph of G on V with edge set E^{out}
A	Adjacency matrix of graph G
A^{in}	Adjacency matrix of graph G^{in}
A^{out}	Adjacency matrix of graph G^{out}
L	Random walk graph Laplacian of G
L^{in}	Random walk graph Laplacian of G^{in}
L^{out}	Random walk graph Laplacian of G^{out}
L_C	submatrix of L with column indices $C \subset V$
L_C^{in}	submatrix of L^{in} with column indices $C \subset V$
$\|M\|$	Entrywised absolute value operation on matrix M
$\|M\|_2$	$\|\cdot\|_2$ norm of matrix M
$\|\mathbf{v}\|$	Entrywised absolute value operation on vector $\mathbf{v}$
$\|\mathbf{v}\|_2$	$\|\cdot\|_2$ norm of vector $\mathbf{v}$.
$\mathbf{1}_C$	Indicator vector on subset $C \subset V$
$\triangle$	Set symmetric difference
Ker	Kernel of the linear map induced by a matrix
$Span$	Spanning set of a set of vectors
$\mathcal{L}_s(\mathbf{v})$	$\{i \in [n] : v_i$ among s largest-in-magnitude entries in $\mathbf{v}\}$

2.2 Model Assumptions

We make the following assumption for our graph model in the asymptotic perspective.

Assumption 2.2.1. *Suppose $G = (V, E)$ can be partitioned into $k = O(1)$ connected components such that $V = C_1 \cup \cdots \cup C_k$, where each C_i is the underlying vertex set for each connected component of G.*

(I) The degree of each vertex is asymptotically the same for vertices belong to the same cluster C_i.

(II) The degree d_i^{out} is small relative to degree d_i^{in} asymptotically for each vertex $i \in V$.

The random graphs which satisfies assumption (I) is not uncommon, for example, the Erdös-Rényi (ER) model $G(n, p)$ with $p \sim \frac{\omega(n)\log(n)}{n}$ for any $\omega(n) \to \infty$, see [35] and [19]. A natural generalization of the ER model is the stochastic block model (SBM) [55], which is a generative model for random graphs with certain edge densities within and between underlying clusters, such that the edges within clusters are more dense than the edges between clusters. In the case of each cluster has the same size and the intra- and inter-connection probability are the same among all the vertices, we have the symmetric stochastic block model (SSBM). It is worthwhile to note that the information theoretical bound for exact cluster recovery in SBM are given in [1] and [2]. It was also shown in [70] that a general SBM under certain assumptions of the parameters can be clustered by using a compressive sensing approach. Our model requires a weaker assumption than the one in [70], indeed, we remove the assumption imposed on the eigenvalues of L in [70]. Therefore, our model will be applicable to a broader range of random graphs.

2.3 Local Clustering based on Least Squares Approach

Our analysis is based on the following key observation. Suppose that graph G has k connected components $C_1, \cdots, C_k$, i.e., $L = L^{in}$. Suppose further that we temporarily have access to the information about the structure of L^{in}. Then we can write the graph Laplacian L^{in} into

a block diagonal form

$$L = L^{in} = \begin{pmatrix} L_1^{in} & & & \\ & L_2^{in} & & \\ & & \ddots & \\ & & & L_k^{in} \end{pmatrix}. \tag{2.1}$$

Suppose now we are interested in finding the cluster with the smallest number of vertices, say C_1, which corresponds to L_1^{in}. By Lemma 1.2.3, $\{\mathbf{1}_{C_1}, \cdots, \mathbf{1}_{C_k}\}$ forms a basis of the kernel W_0 of L. Note that all the $\mathbf{1}_{C_i}$ have disjoint supports, so for $\mathbf{w} \in W_0$ and $\mathbf{w} \neq 0$, we can write

$$\mathbf{w} = \sum_{i=1}^{k} \alpha_i \mathbf{1}_{C_i} \tag{2.2}$$

with some $\alpha_i \neq 0$. Therefore, if $\mathbf{1}_{C_1}$ has the fewest non-zero entries among all elements of $W_0 \setminus \{\mathbf{0}\}$, then we can find it by solving the following minimization problem:

$$\min ||\mathbf{w}||_0 \quad \text{s.t.} \quad L^{in}\mathbf{w} = \mathbf{0} \quad \text{and} \quad \mathbf{w} \neq \mathbf{0}. \tag{2.3}$$

Problem (2.3) can be solved using method such as greedy algorithm in compressive sensing as explained in [70]. Let us propose a different approach and demonstrate that this proposed new approach is more effective numerically and require a fewer number of assumptions.

2.3.1 Least Squares Cluster Pursuit

To solve Problem (2.3), instead of finding C_1 directly, we can find the complement of C_1. Suppose there is a superset $\Omega \subset V$ such that $C_1 \subset \Omega$, and $C_i \not\subset \Omega$ for all $i = 2, \cdots, n$. Since $L^{in}\mathbf{1}_{C_1} = \mathbf{0}$, we have

$$L^{in}\mathbf{1}_{\Omega} = L^{in}(\mathbf{1}_{\Omega \setminus C_1} + \mathbf{1}_{C_1}) = L^{in}\mathbf{1}_{\Omega \setminus C_1} + L^{in}\mathbf{1}_{C_1} = L^{in}\mathbf{1}_{\Omega \setminus C_1}. \tag{2.4}$$

Letting $\mathbf{y} := L^{in}\mathbf{1}_{\Omega}$, then to find what are not in C_1 within Ω is equivalent to solve the following problem (2.5)

$$\arg\min_{\mathbf{x} \in \mathbb{R}^n} \|L^{in}\mathbf{x} - \mathbf{y}\|_2. \tag{2.5}$$

Note that solving problem (2.5) directly will give $\mathbf{x}^* = \mathbf{1}_\Omega \in \mathbb{R}^n$ and $\mathbf{x}^* = \mathbf{1}_{\Omega \setminus C_1} \in \mathbb{R}^n$ both as solutions. By setting the columns $L^{in}_{V \setminus \Omega} = 0$, solving problem (2.5) is equivalent to solving

$$\arg\min_{\mathbf{x} \in \mathbb{R}^{|\Omega|}} \|L^{in}_{\Omega} \mathbf{x} - \mathbf{y}\|_2. \tag{2.6}$$

Directly solving problem (2.6) gives at least two solutions $\mathbf{x}^* = \mathbf{1} \in \mathbb{R}^{|\Omega|}$ and $\mathbf{x}^* = \mathbf{1}_{C_1^\complement} \in \mathbb{R}^{|\Omega|}$, where $C_1^\complement$ indicates the complement set of C_1. Between these two solutions, the latter is much more informative for us to extract C_1 from Ω than the former. We need to find a way to avoid the non-informative solution $\mathbf{x}^* = \mathbf{1}$ but keep the informative solution $\mathbf{x}^* = \mathbf{1}_{C_1^\complement}$.

We can achieve this by removing a subset of columns from index set Ω. Let us use $T \subset \Omega$ to indicate the indices of column we aim to remove. Suppose we could choose T such that $T \subset C_1$. Now consider the following variation (2.7) of the minimization problem (2.6)

$$\arg\min_{\mathbf{x} \in \mathbb{R}^{|\Omega| - |T|}} \|L^{in}_{\Omega \setminus T} \mathbf{x} - \mathbf{y}\|_2. \tag{2.7}$$

Different from solving (2.6) which gives two solutions, solving (2.7) only gives one solution $\mathbf{x}^* = \mathbf{1}_{C_1^\complement}$, as $\mathbf{x}^* = \mathbf{1}$ is no longer a solution because of the removal of T. The solution $\mathbf{x}^* = \mathbf{1}_{C_1^\complement}$ is indeed still a solution to (2.7) because $L^{in}_{\Omega \setminus T} \mathbf{1}_{C_1^\complement} = L^{in} \mathbf{1}_{\Omega \setminus C_1} = 0$. Furthermore, the solution to (2.7) is unique since it is a least squares problem with matrix $L^{in}_{\Omega \setminus T}$ of full column rank, therefore $\mathbf{x}^* = \mathbf{1}_{C_1^\complement}$ is the unique solution to (2.7).

However, there is no way in theory we can select T and assure the condition $T \subset C_1$. In practice, the way we choose T is based on the following observation. Suppose $L = L^{in}$, $\Omega \supset C_1$ and $\Omega \not\supset C_i$ for $i = 2, \cdots k$. Then $|L_a^\top| \cdot |\mathbf{y}| = 0$ for all $a \in C_1$, and $|L_a^\top| \cdot |\mathbf{y}| > 0$ for all $a \in \Omega \setminus C_1$. Therefore, we can choose T in such a way that $|L_t^\top| \cdot |\mathbf{y}|$ is small for all $t \in T \subset \Omega$. These ideas lead to Algorithm 4.

Algorithm 4: Least Squares Cluster Pursuit

Data: Adjacency matrix A, vertex subset $\Omega \subset V$, least squares threshold parameter $\gamma \in (0,1)$, and rejection parameter $0.1 \leq R \leq 0.9$.

Result: $C_1^{\#} = \Omega \setminus W^{\#}$.

1 Compute $L = I - D^{-1}A$ and $\mathbf{y} = L\mathbf{1}_\Omega$;

2 Let T be the set of column indices of $\gamma \cdot |\Omega|$ smallest components of the vector $|{L_\Omega}^\top| \cdot |\mathbf{y}|$;

3 Let $\mathbf{x}^{\#}$ be the solution to

$$\underset{\mathbf{x}\in\mathbb{R}^{|\Omega|-|T|}}{\arg\min} \|L_{\Omega\setminus T}\mathbf{x} - \mathbf{y}\|_2 \tag{2.8}$$

obtained by using an iterative least squares solver;

4 Let $W^{\#} = \{i : \mathbf{x}_i^{\#} > R\}$;

Remark 2.3.1. *We impose the absolute value rather than direct dot product in order to have fewer cancellation between vector components when summing over the entrywised products. In practice, the value of* $\gamma \in (0,1)$ *will not affect the performance too much as long as its value is not too extreme. We find that* $0.15 \leq \gamma \leq 0.4$ *works well for our numerical experiments.*

Remark 2.3.2. *In practice, we choose to use MATLAB's* lsqr *function to solve the least squares problem (2.8). As we will see in Lemma 2.3.2, our problem is well-conditioned, so it is also possible to solve the normal equation exactly for problems which are not in a very large scale. However, we choose to solve it iteratively over exactly because the quality of the numerical solution is not essential for our task here, we are only interested in an approximated solution as we can use the cutoff* R *number for clustering.*

Remark 2.3.3. *As indicated in [70], we can reformulate problem (2.3) as solving*

$$\underset{\mathbf{x}\in\mathbb{R}^n}{\arg\min}\{\|L\mathbf{x} - \mathbf{y}\|_2 : \quad \|x\|_0 \leq s\} \tag{2.9}$$

by applying the greedy algorithms such as subspace pursuit [21] and compressive sensing matching pursuit (CoSaMP) [104]. Or alternatively, we can consider LASSO, see [114] and

[132], formulation of the problem

$$\arg\min_{\mathbf{x}\in\mathbb{R}^n}\{\|L\mathbf{x}-\mathbf{y}\|_2^2+\lambda\|\mathbf{x}\|_1\}=\arg\min_{\mathbf{x}\in\mathbb{R}^n}\{\|L\mathbf{x}-\mathbf{y}\|_2^2+\lambda\|\mathbf{x}\|_0\}. \tag{2.10}$$

The reason that Lasso is a good way to interpret this problem is that the solution $\mathbf{x}^*$ *we are trying to solve for is the sparse indicator vector which satisfies* $\|\mathbf{x}^*\|_1=\|\mathbf{x}^*\|_0$*. We do not analyze it further here.*

However, in reality we have no access to L^{in}, what we know only is L, and in general $L\neq L^{in}$. We argue that the solution to the perturbed problem (2.8) associated with L will not be too far away from the solution to the unperturbed (2.7) problem associated with L^{in}, if the difference between L and L^{in} is relative small. Let us make this precise by first quoting the following standard result in numerical analysis.

Lemma 2.3.1. *Let* $\|\cdot\|$ *be an operator norm,* $A\in\mathbb{R}^{n\times n}$ *be a non-singular square matrix,* $\mathbf{x}\in\mathbb{R}^n$*,* $\mathbf{y}\in\mathbb{R}^n$*. Let* $\tilde{A}$*,* $\tilde{\mathbf{x}}$*,* $\tilde{\mathbf{y}}$ *be perturbed measurements of* A*,* $\mathbf{x}$*,* $\mathbf{y}$ *respectively. Suppose* $A\mathbf{x}=\mathbf{y}$*,* $\tilde{A}\tilde{\mathbf{x}}=\tilde{\mathbf{y}}$*, and suppose further* $cond(A)<\frac{\|A\|}{\|\tilde{A}-A\|}$*, then*

$$\frac{\|\tilde{\mathbf{x}}-\mathbf{x}\|}{\|\mathbf{x}\|}\leq\frac{cond(A)}{1-cond(A)\frac{\|\tilde{A}-A\|}{\|A\|}}\Big(\frac{\|\tilde{A}-A\|}{\|A\|}+\frac{\|\tilde{\mathbf{y}}-\mathbf{y}\|}{\|\mathbf{y}\|}\Big).$$

The above lemma asserts that the size of $\mathrm{cond}(A)$ is significant in determining the stability of the solution $\mathbf{x}$ with respect to small perturbations on A and $\mathbf{y}$. For the discussion from now on, we will use $\|\cdot\|$ to denote the standard vector or matrix induced two-norm $\|\cdot\|_2$ unless state otherwise. The next lemma claims the invertibility of $(L^{in}_{\Omega\setminus T})^\top L^{in}_{\Omega\setminus T}$ and gives an estimation bound of its condition number.

Lemma 2.3.2. *Let* $V=\cup_{i=1}^k C_i$ *be the disjoint union of* $k=O(1)$ *underlying clusters with size* n_i *and assume* (I)*. Let* d_j *be the degree for vertex* $j\in V=[n]$*,* $n_1=\min_{i\in[k]}n_i$*, and suppose* $\Omega\subset V$ *be such that* $\Omega\supset C_1$ *and* $\Omega\not\supset C_i$ *for* $i=2,\cdots k$*. Then*

(i) If $T\subset C_1$*, then* $(L^{in}_{\Omega\setminus T})^\top L^{in}_{\Omega\setminus T}$ *is invertible.*

(ii) Suppose further $\lceil \frac{3n_1}{4} \rceil \leq |T| < n_1$ *and* $\lceil \frac{5n_1}{4} \rceil \leq |\Omega| \leq \lceil \frac{7n_1}{4} \rceil$. *Then*

$$cond\big((L^{in}_{\Omega\setminus T})^\top L^{in}_{\Omega\setminus T}\big) \leq 4$$

almost surely as $n_1 \to \infty$, *e.g. when* $n \to \infty$.

Proof. Without loss of generality, let us assume that the column indices of $L^{in}_{\Omega\setminus T}$ are already permuted such that the indices number is in the same order relative to their underlying clusters. The invertibility of $(L^{in}_{\Omega\setminus T})^\top L^{in}_{\Omega\setminus T}$ follows directly from the fact that $L^{in}_{\Omega\setminus T}$ is of full column rank. So let us show that $L^{in}_{\Omega\setminus T}$ is of full column rank. Because of the reordering, $L^{in}_{\Omega\setminus T}$ is in a block diagonal form

$$L^{in}_{\Omega\setminus T} = \begin{pmatrix} L^{in}_{C_1\setminus T} & & & \\ & L^{in}_{\Omega\cap C_2} & & \\ & & L^{in}_{\Omega\cap C_3} & \\ & & & \ddots \end{pmatrix}.$$

It is then suffices to show each block is of full column rank. By Lemma 1.2.3, each of $L^{in}_{C_i}$ has $\lambda = 0$ as an eigenvalue with multiplicity one, and the corresponding eigenspace is spanned by $\mathbf{1}_{C_i}$. Hence $rank(L^{in}_{C_i}) = |C_i| - 1$. Now suppose by contradiction that the columns of $L^{in}_{C_1\setminus T}$ are linearly dependent, so there exists $\mathbf{v} \neq \mathbf{0}$ such that $L^{in}_{C_1\setminus T}\mathbf{v} = \mathbf{0}$, or $L^{in}_{C_1\setminus T}\mathbf{v} + L^{in}_{T} \cdot \mathbf{0} = \mathbf{0}$. This means that $\mathbf{u} = (\mathbf{v}, \mathbf{0})$ is an eigenvector associated to eigenvalue zero, which contradicts the fact that the eigenspace is spanned by $\mathbf{1}_{C_i}$. Therefore $L^{in}_{C_1\setminus T}$ is linearly independent, hence $L^{in}_{C_1\setminus T}$ is of full column rank. For C_i with $i \geq 2$, since $C_i \notin \Omega$, $\Omega \cap C_i$ is a proper subset of C_i. The strategy above applies as well. Therefore all blocks in $L^{in}_{\Omega\setminus T}$ are of full column rank, so $L^{in}_{\Omega\setminus T}$ is of full column rank.

Now since $(L^{in}_{\Omega\setminus T})^\top L^{in}_{\Omega\setminus T}$ is in a block form, to estimate the condition number, we only need to estimate the largest and smallest eigenvalues for each block. Writing $L^{in}_{\Omega\setminus T} = [l_{ij}]$ and $(L^{in}_{\Omega\setminus T})^\top L^{in}_{\Omega\setminus T} = [s_{ij}]$, for each $i \in C_1 \setminus T$, $s_{ii} = \sum_{k=1}^{n} l_{ki}l_{ki} = \sum_{k=1}^{n} l_{ki}^2 = \sum_{k=1}^{n_1} l_{ki}^2 = 1 + \frac{1}{d_i^{in}}$, and for $i, j \in C_1 \setminus T$ with $i \neq j$, $s_{ij} = \sum_{k=1}^{n} l_{ki}l_{kj} = \sum_{k=1}^{n_1} l_{ki}l_{kj}$. Note that the probability

of having an edge between i and j given degree sequences $d_1, \cdots d_{n_1}$ equals to $\frac{d_i d_j}{\sum_{i\in C_1} d_i}$, as the existence of an edge between two vertices is proportional to their degrees. So l_{ij} equals to $-\frac{1}{d_i}$ with probability $\frac{d_i d_j}{\sum_{i\in C_1} d_i}$, which implies $\mathbb{E}(l_{ij}) = -\frac{d_j}{\sum_{i\in C_1} d_i}$; l_{ji} equals to $-\frac{1}{d_j}$ with probability $\frac{d_i d_j}{\sum_{i\in C_1} d_i}$, which implies $\mathbb{E}(l_{ji}) = -\frac{d_i}{\sum_{i\in C_1} d_i}$. Hence the expectation

$$\begin{aligned}\mathbb{E}(s_{ij}) &= \mathbb{E}(\sum_{k=1}^{n} l_{ki}l_{kj}) = \sum_{k=1}^{n}\mathbb{E}(l_{ki})\mathbb{E}(l_{kj}) = \sum_{k=1}^{n_1}\mathbb{E}(l_{ki})\mathbb{E}(l_{kj}) \\ &= \frac{d_i d_j}{\sum_{i\in C_1} d_i}\cdot(-\frac{1}{d_i}) + \frac{d_i d_j}{\sum_{i\in C_1} d_i}\cdot(-\frac{1}{d_j}) + \frac{d_k d_i}{\sum_{i\in C_1} d_i}\cdot\frac{d_k d_j}{\sum_{i\in C_1} d_i}\cdot(\frac{1}{d_k})^2 \\ &= -\frac{d_i + d_j}{\sum_{i\in C_1} d_i} + \frac{d_i d_j}{(\sum_{i\in C_1} d_i)^2} = -\frac{2}{n_1} + \frac{1}{n_1^2}.\end{aligned}$$

By the law of large numbers, $s_{ij} \to -\frac{2}{n_1} + \frac{1}{n_1^2}$ almost surely as $n_1 \to \infty$. Therefore for $i \in C_1 \setminus T$, we have

$$\sum_{j\in C_1\setminus T, j\neq i} |s_{ij}| \to |C_1 \setminus T|\cdot(\frac{2}{n_1} - \frac{1}{n_1^2}) \le \frac{n_1}{4}\cdot(\frac{2}{n_1} - \frac{1}{n_1^2}) \le \frac{1}{2}$$

almost surely as $n_1 \to \infty$. Similarly, for each $i \in C_k \cap (\Omega \setminus C_1)$, $k \ge 2$, we have $s_{ii} = 1 + \frac{1}{d_i^{in}}$, and $\sum_{j\in C_k\cap(\Omega\setminus C_1), j\neq i} |s_{ij}| \to \frac{n_1}{4}\cdot(\frac{2}{n_k} - \frac{1}{n_k^2}) \le \frac{1}{2}$ almost surely as $n_1 \to \infty$.

Now we apply Gershgorin's circle theorem to bound the spectrum of $(L^{in}_{\Omega\setminus T})^\top L^{in}_{\Omega\setminus T}$. For all $i \in \Omega \setminus T$, the circles are centered at $1 + \frac{1}{d_i}$, with radius less than or equal to $\frac{1}{2}$ almost surely, hence $\sigma_{\min}((L^{in}_{\Omega\setminus T})^\top L^{in}_{\Omega\setminus T}) \ge \frac{1}{2}$ and $\sigma_{\max}((L^{in}_{\Omega\setminus T})^\top L^{in}_{\Omega\setminus T}) \le \frac{3}{2} + \frac{1}{d_i} \le 2$. almost surely. Therefore we have

$$\operatorname{cond}\big((L^{in}_{\Omega\setminus T})^\top L^{in}_{\Omega\setminus T}\big) = \frac{\sigma_{\max}((L^{in}_{\Omega\setminus T})^\top L^{in}_{\Omega\setminus T})}{\sigma_{\min}((L^{in}_{\Omega\setminus T})^\top L^{in}_{\Omega\setminus T})} \le 4$$

almost surely, as desired. □

Remark 2.3.4. *Note that there is a minor difficulty in estimating the expectation of inner product between two different columns of $L^{in}_{\Omega\setminus T}$. The computation assumes the independence of degree distribution of each individual vertex within each cluster, but this may not be true in general for arbitrary graph. However, the independence will occur if the asymptotic uniformity of the degree distribution within each cluster is assumed, that is why our model needs this assumption.*

Now the perturbed problem (2.8) is equivalent to solving $(L_{\Omega\setminus T}^{\top}L_{\Omega\setminus T})\mathbf{x}^{\#} = L_{\Omega\setminus T}^{\top}\tilde{\mathbf{y}} = L_{\Omega\setminus T}^{\top}(L\mathbf{1}_{\Omega})$, while the unperturbed problem (2.7) is to solve $(L_{\Omega\setminus T}^{in})^{\top}L_{\Omega\setminus T}^{in}\mathbf{x}^{*} = (L_{\Omega\setminus T}^{in})^{\top}\mathbf{y} = (L_{\Omega\setminus T}^{in})^{\top}(L^{in}\mathbf{1}_{\Omega})$. Let $M := L - L^{in}$, $M_{\Omega} := L_{\Omega} - L_{\Omega}^{in}$, and $M_{\Omega\setminus T} := L_{\Omega\setminus T} - L_{\Omega\setminus T}^{in}$. Let us give an estimate for M.

Lemma 2.3.3. *Let L be the graph Laplacian of G and $M := L - L^{in}$. Let $\epsilon_i := \frac{d_i^{out}}{d_i}$ for all i and $\epsilon_{max} := \max_{i\in[n]}\epsilon_i$. Then $\|M\| \le 2\epsilon_{\max}$.*

Proof. Let δ_{ij} denote the Kronecker delta symbol, observe that

$$L_{ij} := \delta_{ij} - \frac{1}{d_i}A_{ij} = \delta_{ij} - \frac{1}{d_i^{in}+d_i^{out}}(A_{ij}^{in}+A_{ij}^{out}).$$

Since $\epsilon_i := \frac{d_i^{out}}{d_i}$, we have $\frac{1}{d_i} = \frac{1}{d_i^{in}+d_i^{out}} = \frac{1}{d_i^{in}} - \frac{\epsilon_i}{d_i^{in}}$. So we have

$$\begin{aligned} L_{ij} &= \delta_{ij} - \Big(\frac{1}{d_i^{in}} - \frac{\epsilon_i}{d_i^{in}}\Big)(A_{ij}^{in}+A_{ij}^{out}) \\ &= \Big(\delta_{ij} - \frac{1}{d_i^{in}}A_{ij}^{in}\Big) - \frac{1}{d_i^{in}}A_{ij}^{out} + \frac{\epsilon_i}{d_i^{in}}(A_{ij}^{in}+A_{ij}^{out}) \\ &= L_{ij}^{in} - \frac{1-\epsilon_i}{d_i^{in}}A_{ij}^{out} + \frac{\epsilon_i}{d_i^{in}}A_{ij}^{in}. \end{aligned}$$

Therefore $M_{ij} = -\frac{1-\epsilon_i}{d_i^{in}}A_{ij}^{out} + \frac{\epsilon_i}{d_i^{in}}A_{ij}^{in}$. To bound the spectral norm we apply Gershgorin's circle theorem, noting that $M_{ii} = 0$ for all i, hence

$$\begin{aligned} \|M\| &= \max\{|\lambda_i| : \lambda_i \quad \text{eigenvalue of} \quad M\} \le \max_i \sum_j |M_{ij}| \\ &= \max_i \sum_j \Big| -\frac{1-\epsilon_i}{d_i^{in}}A_{ij}^{out} + \frac{\epsilon_i}{d_i^{in}}A_{ij}^{in}\Big| \\ &\le \max_i \sum_j \Big| -\frac{1-\epsilon_i}{d_i^{in}}\Big|A_{ij}^{out} + \Big|\frac{\epsilon_i}{d_i^{in}}\Big|A_{ij}^{in} \\ &\le \max_i \Big\{\frac{1-\epsilon_i}{d_i^{in}}\sum_j A_{ij}^{out} + \frac{\epsilon_i}{d_i^{in}}\sum_j A_{ij}^{in}\Big\} \\ &= \max_i \Big\{\frac{1-\epsilon_i}{d_i^{in}}d_i^{out} + \frac{\epsilon_i}{d_i^{in}}d_i^{in}\Big\} = 2\max_i \epsilon_i = 2\epsilon_{\max}. \end{aligned}$$

This completes the proof. □

Next we will have the following result.

Lemma 2.3.4. $\|(L^{in}_{\Omega\setminus T})^\top L^{in}_\Omega \mathbf{1}_\Omega\| \geq \frac{\sqrt{|\Omega\setminus C_1|}}{2}$ *almost surely.*

Proof. Note that $\|(L^{in}_{\Omega\setminus T})^\top (L^{in}\mathbf{1}_\Omega)\| = \|(L^{in}_{\Omega\setminus T})^\top L^{in}_\Omega \mathbf{1}\|$. We want to give an estimate of $\|(L^{in}_{\Omega\setminus T})^\top L^{in}_\Omega \mathbf{1}\|$. Similar to the computation we did in Lemma 2.3.2, for each $i \in C_1 \setminus T$, we have $s_{ii} = 1 + \frac{1}{d^{in}_i}$, $\sum_{j\in C_1} s_{ij} = 0$, and $\sum_{j\in\Omega\setminus C_1} s_{ij} = 0$. For each $i \in C_k \cap (\Omega \setminus C_1)$, $k \geq 2$, we have $s_{ii} = 1 + \frac{1}{d^{in}_i}$, $\sum_{j\in C_1} s_{ij} = 0$, and $\sum_{j\in C_k\cap(\Omega\setminus C_1), j\neq i} s_{ij} \to \frac{n_1}{4}\cdot(-\frac{2}{n_k} + \frac{1}{n_k^2}) \geq -\frac{1}{2}$ almost surely. Therefore, the row sum of $(L^{in}_{\Omega\setminus T})^\top L^{in}_\Omega$ for row $i \in C_1 \setminus T$ equals to zero, and the row sum $(L^{in}_{\Omega\setminus T})^\top L^{in}_\Omega$ for row $i \in \Omega \setminus C_1$ larger than $\frac{1}{2}$ almost surely. Hence $\|(L^{in}_{\Omega\setminus T})^\top L^{in}_\Omega \mathbf{1}\| \geq \frac{\sqrt{|\Omega\setminus C_1|}}{2}$ almost surely. □

Now let us use previous lemmas to establish that the difference between perturbed solution and unperturbed solution is small in the order of $\epsilon_{\max}$.

Theorem 2.3.1. *Under the same assumptions as Lemma 2.3.2, let* $\mathbf{x}^\#$ *be the solution to the perturbed problem (2.8), and* $\mathbf{x}^* = \mathbf{1}_{C_1^c} \in \mathbb{R}^{|\Omega|-|T|}$ *which is the solution to the unperturbed problem (2.7). Then*

$$\frac{\|\mathbf{x}^\# - \mathbf{x}^*\|}{\|\mathbf{x}^*\|} = O(\epsilon_{\max})$$

almost surely for large n_1.

Proof. Let $B = (L^{in}_{\Omega\setminus T})^\top L^{in}_{\Omega\setminus T}$, $\tilde{B} = (L_{\Omega\setminus T})^\top L_{\Omega\setminus T}$, $\mathbf{y} = L^{in}\mathbf{1}_\Omega$, $\tilde{\mathbf{y}} = L\mathbf{1}_\Omega$. We will apply Lemma 2.3.2 with B, $\tilde{B}$, $\mathbf{y}$, $\tilde{\mathbf{y}}$.

First by Lemma 2.3.3, we have $\|M\| \leq 2\epsilon_{\max}$. Therefore

$$\begin{aligned}
\|\tilde{B} - B\| &= \big\|(L_{\Omega\setminus T})^\top L_{\Omega\setminus T} - (L^{in}_{\Omega\setminus T})^\top L^{in}_{\Omega\setminus T}\big\| \\
&= \big\|(L^{in}_{\Omega\setminus T})^\top M_{\Omega\setminus T} + M^\top_{\Omega\setminus T} L^{in}_{\Omega\setminus T} + M^\top_{\Omega\setminus T} M_{\Omega\setminus T}\big\| \\
&\leq \big\|(L^{in}_{\Omega\setminus T})^\top M_{\Omega\setminus T}\big\| + \big\|M^\top_{\Omega\setminus T} L^{in}_{\Omega\setminus T}\big\| + \big\|M^\top_{\Omega\setminus T} M_{\Omega\setminus T}\big\| \\
&\leq \big(2\|L^{in}_{\Omega\setminus T}\| + \|M_{\Omega\setminus T}\|\big)\cdot\|M_{\Omega\setminus T}\| \\
&\leq \big(2\|L^{in}_{\Omega\setminus T}\| + \|M\|\big)\cdot\|M\| \\
&\leq 4\epsilon_{\max}\cdot\big(\|L^{in}_{\Omega\setminus T}\| + \epsilon_{\max}\big).
\end{aligned}$$

For each $i \in \Omega \setminus T$, we have $\|L_i\| \geq 1$, and $\sigma_{\max}((L^{in}_{\Omega\setminus T})^\top L^{in}_{\Omega\setminus T}) = \|(L^{in}_{\Omega\setminus T})^\top L^{in}_{\Omega\setminus T}\| = \sigma^2_{\max}(L^{in}_{\Omega\setminus T}) = \|L^{in}_{\Omega\setminus T}\|^2 \geq \max_{i\in\Omega\setminus T}\|L_i\|^2 \geq 1$. Hence

$$\begin{aligned}\frac{\left\|(L_{\Omega\setminus T})^\top L_{\Omega\setminus T} - (L^{in}_{\Omega\setminus T})^\top L^{in}_{\Omega\setminus T}\right\|}{\left\|(L^{in}_{\Omega\setminus T})^\top L^{in}_{\Omega\setminus T}\right\|} \leq& \frac{\big(2\|L^{in}_{\Omega\setminus T}\| + \|M\|\big)\cdot\|M\|}{\|L^{in}_{\Omega\setminus T}\|^2}\\ \leq& \frac{4\epsilon_{\max}}{\|L^{in}_{\Omega\setminus T}\|} + \frac{4\epsilon^2_{\max}}{\|L^{in}_{\Omega\setminus T}\|^2}\\ \leq& 4(\epsilon_{\max} + \epsilon^2_{\max}).\end{aligned} \tag{2.11}$$

We also have

$$\begin{aligned}\|\tilde{\mathbf{y}} - \mathbf{y}\| &= \|(L_{\Omega\setminus T})^\top(L\mathbf{1}_\Omega) - (L^{in}_{\Omega\setminus T})^\top(L^{in}\mathbf{1}_\Omega)\|\\ &= \|(L^{in}_{\Omega\setminus T} + M_{\Omega\setminus T})^\top(L_\Omega\mathbf{1}_\Omega) - (L^{in}_{\Omega\setminus T})^\top(L^{in}_\Omega\mathbf{1}_\Omega)\|\\ &= \|\big((L^{in}_{\Omega\setminus T})^\top M_\Omega + M^\top_{\Omega\setminus T}L^{in}_\Omega + M^\top_{\Omega\setminus T}M_\Omega\big)\cdot\mathbf{1}_\Omega\|\\ &\leq \sqrt{|\Omega|}\cdot\big(\|(L^{in}_{\Omega\setminus T})^\top M_\Omega\| + \|M^\top_{\Omega\setminus T}L^{in}_\Omega\| + \|M^\top_{\Omega\setminus T}M_\Omega\|\big)\\ &\leq \sqrt{|\Omega|}\cdot\big(2\|L^{in}_\Omega\| + \|M_\Omega\|\big)\cdot\|M_\Omega\|\\ &\leq 4\sqrt{|\Omega|}\cdot\big(\|L^{in}_\Omega\| + \epsilon_{\max}\big)\cdot\epsilon_{\max}.\end{aligned}$$

Next by Lemma 2.3.4, $\|(L^{in}_{\Omega\setminus T})^\top L^{in}_\Omega\mathbf{1}_\Omega\| \geq \frac{\sqrt{|\Omega\setminus C_1|}}{2}$ almost surely. Therefore

$$\begin{aligned}\frac{\|(L_{\Omega\setminus T})^\top L\mathbf{1}_\Omega - (L^{in}_{\Omega\setminus T})^\top L^{in}\mathbf{1}_\Omega\|}{\|(L^{in}_{\Omega\setminus T})^\top L^{in}\mathbf{1}_\Omega\|} \leq& \frac{4\sqrt{|\Omega|}\cdot\big(\|L^{in}_\Omega\| + \epsilon_{\max}\big)\cdot\epsilon_{\max}}{\sqrt{|\Omega\setminus C_1|}/2}\\ \leq& 8\sqrt{5}\epsilon_{\max}\cdot\big(\|L^{in}_\Omega\| + \epsilon_{\max}\big)\\ \leq& 8\sqrt{5}\epsilon_{\max}\cdot\big(\sqrt{2} + \epsilon_{\max}\big)\\ =& 8\sqrt{10}\epsilon_{\max} + 8\sqrt{5}\epsilon^2_{\max}.\end{aligned}$$

The second inequality holds since $|\Omega| \geq \lceil\frac{5n_1}{4}\rceil$. The third inequality holds since $\sigma_{\max}((L^{in}_\Omega)^\top L^{in}_\Omega) \leq 2$, which comes from the similar reasoning as in Lemma 2.3.2 by using Gershgorin's circle theorem. Consequently, we have $\|L^{in}_\Omega\| \leq \sqrt{2}$. Now putting Lemma 2.3.2 and Lemma 2.3.1

together with $B = (L^{in}_{\Omega\setminus T})^\top L^{in}_{\Omega\setminus T}$, $\tilde{B} = (L_{\Omega\setminus T})^\top L_{\Omega\setminus T}$, $\mathbf{y} = L^{in}\mathbf{1}_\Omega$, $\tilde{\mathbf{y}} = L\mathbf{1}_\Omega$, we have

$$\begin{aligned}\frac{\|\mathbf{x}^\# - \mathbf{x}^*\|}{\|\mathbf{x}^*\|} &\le \frac{cond\big((L^{in}_{\Omega\setminus T})^\top L^{in}_{\Omega\setminus T}\big)\cdot\big(4\epsilon_{\max} + 4\epsilon^2_{\max} + 8\sqrt{10}\epsilon_{\max} + 8\sqrt{5}\epsilon^2_{\max}\big)}{1 - cond\big((L^{in}_{\Omega\setminus T})^\top L^{in}_{\Omega\setminus T}\big)\cdot\big(4\epsilon_{\max} + 4\epsilon^2_{\max}\big)} \\ &\le \frac{16\big((1+2\sqrt{10})\epsilon_{\max} + (1+2\sqrt{5})\epsilon^2_{\max}\big)}{1 - 16\epsilon_{\max}(1+\epsilon_{\max})} = O(\epsilon_{\max}).\end{aligned}$$

□

Next we can estimate the size of the symmetric difference between output $C_1^\#$ and the true cluster C_1 relative to the size of C_1, the symmetric difference is defined as $C_1^\# \triangle C_1 := (C_1^\# \setminus C_1) \cup (C_1 \setminus C_1^\#)$. Let us state another lemma before we establish the result.

Lemma 2.3.5. *Let $T \subset [n]$, $\mathbf{v} \in \mathbb{R}^n$, and $W^\# = \{i : \mathbf{v}_i > R\}$. Suppose $\|\mathbf{1}_T - \mathbf{v}\| \le D$, then $|T \triangle W^\#| \le \frac{D^2}{\min\{R^2,(1-R)^2\}}$.*

Proof. Let $U^\# = [n] \setminus W^\#$ and write $\mathbf{v} = \mathbf{v}_{U^\#} + \mathbf{v}_{W^\#}$, where $\mathbf{v}_{U^\#}$ and $\mathbf{v}_{W^\#}$ are the parts of $\mathbf{v}$ supported on $U^\#$ and $W^\#$. Then we can write

$$\|\mathbf{1}_T - \mathbf{v}\|^2 = \|\mathbf{1}_{T\cap W^\#} - (\mathbf{v}_{W^\#})_{T\cap W^\#}\|^2 + \|(\mathbf{v}_{W^\#})_{W^\#\setminus T}\|^2 + \|\mathbf{1}_{T\setminus W^\#} - \mathbf{v}_{U^\#}\|^2.$$

Note that $\|(\mathbf{v}_{W^\#})_{W^\#\setminus T}\|^2 \ge R^2 \cdot |W^\# \setminus T|$ and $\|\mathbf{1}_{T\setminus W^\#} - \mathbf{v}_{U^\#}\|^2 \ge (1-R)^2 \cdot |T \setminus W^\#|$. We have

$$\begin{aligned}\|\mathbf{1}_T - \mathbf{v}\|^2 &\ge \|(\mathbf{v}_{W^\#})_{W^\#\setminus T}\|^2 + \|\mathbf{1}_{T\setminus W^\#} - \mathbf{v}_{U^\#}\|^2 \\ &\ge R^2 \cdot |W^\# \setminus T| + (1-R)^2 \cdot |T \setminus W^\#| \\ &\ge \min\{R^2, (1-R)^2\} \cdot (|W^\# \setminus T| + |T \setminus W^\#|) \\ &= \min\{R^2, (1-R)^2\} \cdot |T \triangle W^\#|.\end{aligned}$$

Therefore $|T \triangle W^\#| \le \frac{\|\mathbf{1}_T - \mathbf{v}\|^2}{\min\{R^2,(1-R)^2\}} \le \frac{D^2}{\min\{R^2,(1-R)^2\}}$ as desired. □

Theorem 2.3.2. *Under the same assumptions as Theorem 2.3.1, we have*

$$\frac{|C_1^\# \triangle C_1|}{|C_1|} \le O(\epsilon^2_{\max}).$$

In other words, the error rate of successfully recovering C_1 is at most a constant multiple of $\epsilon^2_{\max}$.

Proof. From Theorem 2.3.1, we have $\|\mathbf{x}^{\#} - \mathbf{x}^{*}\| = \|\mathbf{x}^{\#} - \mathbf{1}_{\Omega \setminus C_1}\| \leq O(\epsilon_{\max}) \cdot \|\mathbf{x}^{*}\| \leq O(\epsilon_{\max}\sqrt{n_1})$. By Lemma 2.3.5, we get $|W^{\#} \triangle (\Omega \setminus C_1)| \leq O(\epsilon_{\max}^2 n_1)$. Since $C_1^{\#} = \Omega \setminus W^{\#}$, it then follows $|C_1^{\#} \triangle C_1| \leq O(\epsilon_{\max}^2 n_1)$, hence $\frac{|C_1^{\#} \triangle C_1|}{|C_1|} = O(\epsilon_{\max}^2)$ as desired. □

2.3.2 Random Walk Threshold

In order to apply Algorithm 4, we need a "nice" superset which contains C_1. The task for this subsection is to find such a superset Ω from the given seeds Γ. We will apply a simple diffusion based random walk algorithm on G to find such Ω. This leads to Algorithm 5, which is described in [70] as well. However, the difference between our random walk threshold algorithm and the one in [70] is that the threshold parameter δ here is heuristically chosen to be larger than the corresponding threshold parameter in [70]. This is another advantage of our method as it will increase the chances of having C_1 entirely contained in Ω. Such a choice is made based on the natural differences of our approaches. It is worthwhile to point out that there are also other sophisticated algorithms such as the ones described in [5], [63] and [139] which can achieve the same goal. We avoid using these methods here as our purpose is just to implement a fast way of obtaining a set $\Omega \supset C_1$.

Algorithm 5: Random Walk Threshold

Data: Adjacency matrix A, a random walk threshold parameter $\delta \in (0, 1)$, a set of seed vertices $\Gamma \subset C_1$, estimated size $\hat{n}_1 \approx |C_1|$, and depth of random walk $t \in \mathbb{Z}^{+}$.

Result: $\Omega = \Omega \cup \Gamma$.

1 Compute $P = AD^{-1}$ and $\mathbf{v}^{(0)} = D\mathbf{1}_{\Gamma}$;

2 Compute $\mathbf{v}^{(t)} = P^t \mathbf{v}^{(0)}$;

3 Define $\Omega = \mathcal{L}_{(1+\delta)\hat{n}_1}(\mathbf{v}^{(t)})$;

The thresholding operator $\mathcal{L}_s(\cdot)$ is defined as

$$\mathcal{L}_s(\mathbf{v}) := \{i \in [n] : v_i \text{ among } s \text{ largest entries in } \mathbf{v}\}.$$

The motivation of Algorithm 5 is the following intuitive observation. Suppose we are given seed vertices $\Gamma \subset C_1$, then by starting from Γ, since the edges within each cluster are more dense than those between different clusters, the probability of staying within C_1 will be much higher than entering other clusters C_i, for $i \neq 1$, in a short amount of depth. Therefore, by performing a random walk up to a certain depth t, e.g., $t = 3$, we will have a well approximated set Ω such that C_1 is almost surely contained in Ω. Let us make this more precisely in Theorem 2.3.3.

Theorem 2.3.3. *Assume $|\Gamma| = O(1)$ and $t = O(1)$ in Algorithm 5, the probability $\mathbb{P}\big(C_1 \subset \Omega\big) \geq \mathbb{P}\big(\sum_{j\in C_1} \mathbf{v}_j^{(t)} = \|\mathbf{v}^{(t)}\|_1\big) \geq 1 - O(\epsilon_{\max})$. In other words, the probability that the t-steps random walk with seed vertices Γ being not in C_1 is at most a constant multiple of $\epsilon_{\max}$.*

Proof. Let us first consider the case $|\Gamma| = 1$. Suppose $\Gamma = \{s\}$. Then we have $\mathbb{P}\big(\sum_{j\in C_1} \mathbf{v}_j^{(0)} = \|\mathbf{v}^{(0)}\|_1\big) = \mathbb{P}(\mathbf{v}_s^{(0)} = \|\mathbf{v}^{(0)}\|_1) = 1$. It is also easy to see that $\mathbb{P}\big(\sum_{j\in C_1} \mathbf{v}_j^{(1)} = \|\mathbf{v}^{(1)}\|_1\big) = d_i^{in}/d_i = 1 - \epsilon_i \geq 1 - \epsilon_{\max}$. For $t \geq 2$, we have $\mathbb{P}\big(\sum_{j\in C_1} \mathbf{v}_j^{(t)} = \|\mathbf{v}^{(t)}\|_1\big) \geq (1 - \epsilon_{\max}) \cdot \mathbb{P}\big(\sum_{j\in C_1} \mathbf{v}_j^{(t-1)} = \|\mathbf{v}^{(t-1)}\|_1\big)$. So by assuming $\mathbb{P}\big(\sum_{j\in C_1} \mathbf{v}_j^{(t-1)} = \|\mathbf{v}^{(t-1)}\|_1\big) \geq (1-\epsilon_{\max})^{t-1} \geq 1 - (t-1)\epsilon_{\max}$, we have $\mathbb{P}\big(\sum_{j\in C_1} \mathbf{v}_j^{(t)} = \|\mathbf{v}^{(t)}\|_1\big) \geq (1 - \epsilon_{\max})^t \geq 1 - t\epsilon_{\max} = 1 - O(\epsilon_{\max})$.

Suppose now $|\Gamma| > 1$, we can apply the above argument to each individual vertex in Γ, where the random walk starting from each vertex can be considered independently, therefore we have $\mathbb{P}\big(\sum_{j\in C_1} \mathbf{v}_j^{(t)} = \|\mathbf{v}^{(t)}\|_1\big) \geq (1 - t\epsilon_{\max})^{|\Gamma|} \geq 1 - t\epsilon_{\max}|\Gamma| = 1 - O(\epsilon_{\max})$. □

Remark 2.3.5. *It is worthwhile to note that we do not want t to be too large, one reason is that Theorem 2.3.3 tells us the probability of staying within the target cluster C_1 decreases as t increases. An alternative interpretation is that we can treat our graph G, suppose connected, as a time homogeneous finite state Markov chain with evenly distributed transition probability determined by the vertex degree between adjacent vertices. Since G is connected, it is certainly irreducible and aperiodic. By the fundamental theorem of Markov chains, the limiting probability of finally being at each vertex will be the same, regardless of what the seed set Γ is. Meanwhile, we do not want t to be too small as well, otherwise the random walk will*

not be able to explore all the reachable vertices. There is also a trade-off between the size of Γ *and the random walk depth* t*, where a smaller size of* Γ *usually induces a larger* t *in order to fully explore the target cluster.*

2.3.3 Local Cluster Extraction

Let us now combine the previous two subroutines into Algorithm 6. In practice, we may want to vary the number of iterations $MaxIter$ based on the number of examples in the data set in order to achieve a better performance. For the purpose theoretical analysis, let us fix $MaxIter = 1$.

Algorithm 6: Least Squares Clustering (LSC)

Data: Adjacency matrix A, a random walk threshold parameter $\delta \in (0, 1)$, a set of seed vertices $\Gamma \subset C_1$, estimated size $\hat{n}_1 \approx |C_1|$, depth of random walk $t \in \mathbb{Z}^+$, least squares parameter $\gamma \in (0, 0.8)$, and rejection parameter $R \in [0, 1)$.

Result: $C_1^{\#}$.

```
for i = 1, ··· , MaxIter do
    Ω ⟵ Random Walk Threshold (A, Γ, n̂_1, ϵ, t);
    Γ ⟵ Least Squares Cluster Pursuit (A, Ω, R, γ);
end
Let C_1^# = Γ;
```

Remark 2.3.6. *The hyperparameter* $MaxIter$ *in the algorithm is usually choosen based on the size of initial seed vertices* Γ *relative to* n*, we do not have a formal way of choosing the best* $MaxIter$ *rather than choose it heuristically. In practice, we believe* $MaxIter \leq 3$ *will do a very good job most of the time.*

The analysis in previous two subsections gives that the difference between true cluster C_1 and the estimated $C_1^{\#}$ is relative small compared to the size of C_1, this can be written more formally using the asymptotic notation.

Theorem 2.3.4. *Suppose* $\epsilon_{\max} = o(1)$ *and* $MaxIter = 1$*, then under the assumptions of Theorem 2.3.2 and 2.3.3, we have* $\mathbb{P}\Big(\frac{|C_1^{\#} \triangle C_1|}{|C_1|} \leq o(1)\Big) = 1 - o(1)$.

Proof. By Theorem 2.3.3, we know that the probability of $\Omega \supset C_1$ after performing Algorithm 5 is $1 - O(\epsilon_{\max}) = 1 - o(1)$. By Theorem 2.3.2, the error rate is at most a constant multiple of $\epsilon_{\max}^2$ after performing Algorithm 4. Putting them together, we have $\mathbb{P}\Big(\frac{|C_1^{\#} \triangle C_1|}{|C_1|} \leq o(1)\Big) = 1 - o(1)$. □

2.3.4 From Local to Global

We can make one step further by applying Algorithm 6 iteratively on the entire graph to extract all the underlying clusters. That is, we remove $C_i^{\#}$ each time after the Algorithm 6 finds it, and update the graph G by removing the subgraph spanned by vertices $C_i^{\#}$ successively. This leads to Algorithm 7. We will not analyze further the theoretical guarantees of the iterative version the algorithm, but rather provide with numerical examples in the last section of next chapter to show its effectiveness and efficiency.

Algorithm 7: Iterative Least Squares Clustering (ILSC)

Data: Adjacency matrix A, random walk threshold parameter $\delta \in (0, 1)$, least squares parameter $\gamma \in (0, 0.8)$, rejection parameter $R \in [0, 1)$, depth of random walk $t \in \mathbb{Z}^+$. Seed vertices for each cluster $\Gamma_i \subset C_i$, estimated size $\hat{n}_i \approx |C_i|$ for $i = 1, \cdots k$.

Result: $C_1^{\#}, \cdots, C_k^{\#}$.

```
for i = 1, ··· , k do
    Let C_i^# be the output of Least Squares Clustering;
    Let G^(i) be the subgraph spanned by C_i^#;
    Updates G ← G \ G^(i);
end
```

Remark 2.3.7. *It is worth noting that Algorithm 7 extracts one cluster at a time, which is different from most of other global unsupervised clustering algorithms. In practice, those*

global clustering methods could have impractically high run time [103] or tricky to implement [2]. In contrast, our method requires much lower computational time and can be implemented easily. In addition, the "one cluster at a time" feature of our method provides more flexibility for problems under certain circumstances.

2.4 Computational Complexity

In this section, let us discuss the run time of the algorithms introduced previously.

Theorem 2.4.1. *Algorithm 5 requires* $O(nd_{\max}t + n\log(n))$ *operations, where* t *is the depth of the random walk.*

Proof. Notice that if A, D, P are stored as sparse matrices, then for each t in the second step of Algorithm 5, it requires $O(nd_{\max})$, where $d_{\max}$ is the maximal degrees among all the vertices. Therefore the algorithm requires $O(nd_{\max}t + n\log(n))$, where the $O(n\log(n))$ part comes from the third step of sorting. In practice, the random walk depth t is $O(1)$ with respect to the graph size n, therefore we have $O(nd_{\max} + n\log(n))$. □

Theorem 2.4.2. *Algorithm 4 requires* $O(nd_{\max} + n\log(n))$ *operations.*

Proof. For Algorithm 4, its first step requires $O(nd_{\max})$, second step requires $O(nd_{\max} + n\log(n))$, where the $O(nd_{\max})$ part comes from matrix vector multiplication, and $O(n\log(n))$ part comes from sorting. For its third step, to avoid solving the normal equation exactly for large scale problems, we recommend using an iterative mehod, for example conjugate gradient descent (we use MATLAB's *lsqr* operation in our implementation). As we have shown the matrices are associated with well behaved condition numbers, it requires only a constant number of iterations to get a well approximated least squares solution to problem (2.8). Since the cost for each iteration in conjugate gradient descent equals to a few operations of matrix vector multiplication, which is $O(nd_{\max})$, the total cost for Algorithm 4 is $O(nd_{\max} + n\log(n))$. □

As a consequence, the total run time for Algorithm 6 is $O(nd_{\max}+n\log(n))$. if the number of clusters $k = O(1)$, then Algorithm 7 also runs in $O(nd_{\max} + n\log(n))$.

Remark 2.4.1. *The computational scheme of our methods follow the similar framework as CP+RWT in [70]. However, one of the differences between these two approaches is that we apply* lsqr *to solve the least squares problem (2.8), but CP+RWT applies* $O(\log n)$ *iterations of subspace pursuit algorithm to solve (2.8), and each its subspace pursuit is implemented with* lsqr *as a subroutine. So essentially, our proposed method is* $O(\log n)$ *times cheaper than CP+RWT. We can also see this difference by comparing the run times for our numerical experiments in the last section of next chapter.*

Chapter 3

Semi-supervised Local Clustering via Compressive Sensing

One of the major concerns for the local clustering approaches proposed in [70] and [77] is the low quality of the initial cut. In this chapter, we introduce a more recent approach for local clustering. Our approach discussed in this chapter improves the aforementioned two works by making the initial cut to be the entire graph and hence overcomes the issue that missing vertices of the target cluster from the initial cut are not recoverable in the later stage. We will discuss both the theoretical framework and conduct comprehensive experiments to showcase the effectiveness of this approach.

3.1 Local Clustering based on Compressive Sensing Approach

Recall that the local clustering task can be considered as a compressive sensing problem in the following way. Suppose the vertices have been sorted according to their memberships, i.e., the first n_1 rows and columns in L^{in} corresponds to all the vertices in C_1, the last n_k rows and columns corresponds to all the vertices in C_k, etc,.

Let $L^{in}_{-1} = (\ell^{in}_2, \cdots, \ell^{in}_n)$ be the matrix obtained from $L^{in} = (\ell^{in}_1, \cdots, \ell^{in}_n)$ by deleting the first column from C_1. Let us take a look of a specific example of L^{in}, which is shown in equation (3.1). For the graph associated with this L^{in}, all the clusters have size three. The symbol $*$ equals to $-1/2$, and all the other entries in the off-diagonal blocks equal to zero.

$$L_{-1}^{in} = \begin{pmatrix} * & * & & & & & & & \\ 1 & * & & & & & & & \\ * & 1 & & & & & & & \\ & & 1 & * & * & & & & \\ & & * & 1 & * & & & & \\ & & * & * & 1 & & & & \\ & & & & & \ddots & & & \\ & & & & & & 1 & * & * \\ & & & & & & * & 1 & * \\ & & & & & & * & * & 1 \end{pmatrix} = (\ell_2^{in}, \cdots, \ell_n^{in}). \tag{3.1}$$

Then we can easily verify the desired solution to the compressive sensing problem

$$\min \|\mathbf{x}\|_0 \quad s.t. \quad L_{-1}^{in}\mathbf{x} = -\ell_1^{in} \tag{3.2}$$

is $\mathbf{x}^* = (1, 1, 0, \cdots, 0)'$. The significance of this formulation is that the nonzero components in $\mathbf{x}^*$ correspond to the indices of vertices which belong to the target cluster C_1. This gives us the intuitive idea of how to apply compressive sensing for solving local clustering problem.

However, we usually do not have access to L^{in} or L_{-1}^{in}, what we do have access to are L and L_{-1}. We can relax the exact equality condition to approximately equal to, so the problem becomes

$$\min \|\mathbf{x}\|_0 \quad s.t. \quad L_{-1}\mathbf{x} \approx \mathbf{y}, \tag{3.3}$$

where $\mathbf{y}$ is the row sum vector of L_{-1}. Let $\mathbf{x}^{\#}$ be the solution to (3.3). Suppose the graph has a good underlying clusters structure, in other words, the entries in the off-diagonal block of L_{-1} have very small magnitude, i.e., $L_{-1} \approx L_{-1}^{in}$. Then we should have $\mathbf{y} \approx \mathbf{y}^{in}$, and hence the difference between $\mathbf{x}^{\#}$ and $\mathbf{x}^*$ should be small in certain sense. We can then use some cut-off number $R \in (0, 1)$ to separate the coordinates of $\mathbf{x}^{\#}$ and therefore extract the target cluster from the entire graph.

3.2 Main Algorithm

In general, we can remove more than just one column. That is, we remove a set $T \subset V$ in a somewhat smart way, with the hope that $T \subset C_1$, and then we solve

$$\min \|\mathbf{x}\|_0 \quad s.t. \quad \|L_{V\setminus T}\mathbf{x} - \mathbf{y}\|_2 \leq \epsilon. \tag{3.4}$$

Or equivalently, we solve

$$\underset{\mathbf{x}\in\mathbb{R}^{|V|-|T|}}{\arg\min} \{\|L_{V\setminus T}\mathbf{x} - \mathbf{y}\|_2 : \|\mathbf{x}\|_0 \leq s\} \tag{3.5}$$

where vector $\mathbf{y}$ is the row sum vector of $L_{V\setminus T}$ and s is the sparsity constraint.

Naively, if the size of Γ is not too small, then we can just choose $T = \Gamma$. However, for the scope of our problem, the size of Γ is assumed to be small relative to the size of C_1, therefore this choice does not work well in practice. Instead, we select T based on a heuristic criterion (as described in step 4) on a candidate set Ω which is obtained from a random walk originates from Γ. We also find that the size of T does not matter too much based on our exploration in the experiments. The idea is summarized in Algorithm 8 as CS-LCE. We give a more detailed explanation about several aspects of the algorithm in Remark 3.2.1 and Remark 3.2.2. More generally, we can apply CS-LCE iteratively to extract all the clusters one at a time.

Algorithm 8: Compressive Sensing for Local Cluster Extraction (CS-LCE)

Data: Adjacency matrix A, a small set of seeds $\Gamma \subset C_1$, estimated size $\hat{n}_1 \approx |C_1|$, random walk threshold parameter $\epsilon \in (0,1)$, random walk depth $t \in \mathbb{Z}^+$, sparsity parameter $\gamma \in [0.1, 0.5]$, rejection parameter $R \in [0.1, 0.9]$.

Result: The target cluster C_1.

1 Compute $P = AD^{-1}$, $\mathbf{v}^0 = D\mathbf{1}_\Gamma$, and $L = I - D^{-1}A$;

2 Compute $\mathbf{v}^{(t)} = P^t\mathbf{v}^{(0)}$;

3 Define $\Omega = \mathcal{L}_{(1+\epsilon)\hat{n}_1}(\mathbf{v}^{(t)})$;

4 Let T be the set of column indices of $\gamma \cdot |\Omega|$ smallest components of the vector $|L_\Omega^\top| \cdot |L\mathbf{1}_\Omega|$;

5 Set $\mathbf{y} := L\mathbf{1}_{V\setminus T}$. Let $\mathbf{x}^\#$ be the solution to

$$\underset{\mathbf{x}\in\mathbb{R}^{|V|-|T|}}{\arg\min} \{\|L_{V\setminus T}\mathbf{x} - \mathbf{y}\|_2 : \|\mathbf{x}\|_0 \leq (1-\gamma)\hat{n}_1\} \tag{3.6}$$

obtained by using $O(\log n)$ iterations of *Subspace Pursuit* [21];

6 Let $W^\# = \{i : \mathbf{x}_i^\# > R\}$;

7 **return** $C_1^\# = W^\# \cup T$.

We would like to point out the major differences between CS-LCE with its counterparts CP+RWT in [70] and LSC in [77]. The key difference is that the latter two methods only be able to extract target cluster from the initial cut Ω, since it is assumed that $C_1 \subset \Omega$ in these two methods before extracting all the vertices in C_1, and once Ω fails to contain any vertex in C_1, there is no chance for CP+RWT or LSC to recover those vertices in the later stage. However, such an assumption is not needed in CS-LCE. Since the sensing matrix in CS-LCE is associated with all the vertices corresponding to $V \setminus T$, it is very probable for CS-LCE to still be able to find the vertices which are in C_1 but not in Ω.

Remark 3.2.1. *The purpose of Ω is solely for obtaining the set T, and the vector $\mathbf{y}$ is computed by adding up all the columns with indices in the set $V \setminus T$. This is another key*

difference between CS-LCE and CP+RWT [70] and LSC [77], whereas the latter two methods directly use Ω to obtain $\mathbf{y}$.

Remark 3.2.2. *The rationale for choosing an iterative approach such as Subspace Pursuit over other sophisticated optimization algorithms for solving (3.6) comes from the nature of our task. Since the task is clustering, all we need is a relative good estimated solution instead of the exact solution, then we can use a cutoff number R in Algorithm 8 to separate the aimed cluster from the remaining of the graph. Due to the nature of an iterative approach, the convergence is usually fast at the beginning and slow in the end, so we can stop early in the iteration to save the computational cost once the estimated solution is roughly "close enough" to the true solution.*

3.3 Theoretical Analysis

For convenience, let us fix $\gamma = 0.4$ for the rest of discussion. We want to make sure the output $C_1^{\#}$ from Algorithm 8 is as close to the true cluster C_1 as possible. In order to investigate more towards this aspect, let us use $\mathbf{x}^*$ to denote the solution to the unperturbed problem:

$$\mathbf{x}^* := \underset{\mathbf{x}\in\mathbb{R}^{|V|-|T|}}{\arg\min} \{\|L^{in}_{V\setminus T}\mathbf{x} - \mathbf{y}^{in}\|_2 : \|\mathbf{x}\|_0 \leq 0.6n_1\} \tag{3.7}$$

where $\mathbf{y}^{in} = L^{in}\mathbf{1}_{V\setminus T}$. Let $\mathbf{x}^{\#}$ be the solution to (3.6), the perturbed problem, with $\gamma = 0.4$.

Let us first establish the correctness of having $\mathbf{x}^*$ equals to an indicator vector as the solution to (3.7), and then conclude that $\mathbf{x}^{\#} \approx \mathbf{x}^*$ if $L \approx L^{in}$ in a certain sense. Once this is established, we will be able to conclude $C_1^{\#} \approx C_1$. These results are summarized in the following as a series of theorems and lemma.

Theorem 3.3.1. *Suppose $T \subset C_1$. Then $\mathbf{x}^* = \mathbf{1}_{C_1\setminus T} \in \mathbb{R}^{|V|-|T|}$ is the unique solution to (3.7).*

Proof. Note that for $\mathbf{y}^{in} = L^{in}\mathbf{1}_{V\setminus T}$, we can rewrite it as $\mathbf{y}^{in} = L^{in}_{V\setminus T}\mathbf{1}$ where $\mathbf{1} \in \mathbb{R}^{|V|-|T|}$. It is straightforward to check $\mathbf{x}^* = \mathbf{1}_{C_1\setminus T}$ is a solution to (3.7). The rest is to show it is unique.

Suppose otherwise, then since $L^{in}_{V\setminus T}\mathbf{1}_{C_1\setminus T} = \mathbf{y}^{in}$, we want to find $\mathbf{x} \in \mathbb{R}^{|V|-|T|}$ and $\mathbf{x} \neq \mathbf{1}_{C_1\setminus T}$ such that $L^{in}_{V\setminus T}(\mathbf{x}-\mathbf{1}) = \mathbf{0}$. Without loss of generality, let us assume the columns of L are permuted such that it is in the block diagonal form, i.e.,

$$L^{in}_{V\setminus T} = \begin{pmatrix} L^{in}_{C_1\setminus T} & & & \\ & L^{in}_{C_2} & & \\ & & \ddots & \\ & & & L^{in}_{C_n} \end{pmatrix}.$$

Let us now show that $L^{in}_{C_1\setminus T}$ is of full column rank, i.e., the columns of $L^{in}_{C_1\setminus T}$ is linearly independent. We first observe the following fact. By Lemma 1.2.3, each of $L^{in}_{C_i}$ has $\lambda = 0$ as an eigenvalue with multiplicity one, and the corresponding eigenspace is spanned by $\mathbf{1}_{C_i}$. Now suppose by contradiction that the columns of $L^{in}_{C_1\setminus T}$ are linearly dependent, so there exists $\mathbf{v} \neq \mathbf{0}$ such that $L^{in}_{C_1\setminus T}\mathbf{v} = \mathbf{0}$, or $L^{in}_{C_1\setminus T}\mathbf{v} + L^{in}_{T} \cdot \mathbf{0} = \mathbf{0}$. This means that $\mathbf{u} = (\mathbf{v}, \mathbf{0})$ is an eigenvector associated to eigenvalue zero, which contradicts the fact that the eigenspace is spanned by $\mathbf{1}_{C_i}$. Therefore $L^{in}_{C_1\setminus T}$ is of full column rank.

Since $L^{in}_{C_1\setminus T}$ is of full column rank, and $Ker(L^{in}_{C_i}) = Span\{\mathbf{1}_{C_i}\}$ for $i \geq 2$. We conclude that $\mathbf{x}-\mathbf{1} \in Ker(L^{in}_{V\setminus T}) = Span\{\mathbf{1}_{C_2}, \cdots, \mathbf{1}_{C_n}\}$. Therefore in order to satisfy $\|\mathbf{x}\|_0 \leq 0.6n_1$, it is easy to see $\mathbf{x} = \mathbf{1} - \mathbf{1}_{C_2} - \mathbf{1}_{C_3} - \cdots - \mathbf{1}_{C_k} = \mathbf{1}_{C_1\setminus T}$, which results in a contradiction by our assumption. □

The next theorem shows that $\mathbf{x}^*$ and $\mathbf{x}^\#$ are close to each other if L and L^{in} are close.

Theorem 3.3.2. *Let $M := L - L^{in}$. Suppose $T \subset C_1$, $\|M\|_2 = o(n^{-1/2})$ and $\delta_{1.8n_1}(L) = o(1)$. Then*

$$\frac{\|\mathbf{x}^\# - \mathbf{x}^*\|_2}{\|\mathbf{x}^*\|_2} = o(1). \tag{3.8}$$

Proof. Recall that $\mathbf{x}^{\#}$ is the output to (3.6) after $O(\log n)$ iterations of *Subspace Pursuit*. By our assumption on M, we have

$$\begin{aligned}\|\mathbf{y}-\mathbf{y}^{in}\|_2 &= \|L\mathbf{1}_{V\setminus T}-L^{in}\mathbf{1}_{V\setminus T}\|_2 = \|(L-L^{in})\mathbf{1}_{V\setminus T}\|_2\\ &\le \|M\|_2\|\mathbf{1}_{V\setminus T}\|_2 \le o(n^{-1/2})\cdot\sqrt{n} = o(1).\end{aligned}$$

Then applying Theorem 2.5 in [70], we get the desired result. □

Lemma 3.3.1. *Consider $K\subset[n]$, any $\mathbf{v}\in\mathbb{R}^n$, and $W^{\#}=\{i:\mathbf{v}_i>R\}$. If $\|\mathbf{1}_K-\mathbf{v}\|_2\le D$, then $|K\triangle W^{\#}|\le\frac{D^2}{\min\{(1-R)^2,R^2\}}$.*

Proof. Let $U^{\#}=[n]\setminus W^{\#}$, we can write $\mathbf{v}=\mathbf{v}_{U^{\#}}+\mathbf{v}_{W^{\#}}$ where $\mathbf{v}_{U^{\#}}$ and $\mathbf{v}_{W^{\#}}$ are the components of $\mathbf{v}$ supported on $U^{\#}$ and $W^{\#}$ respectively. Then we have

$$\begin{aligned}\|\mathbf{1}_K-\mathbf{v}\|_2^2 &= \|\mathbf{1}_K-\mathbf{v}_{U^{\#}}-\mathbf{v}_{W^{\#}}\|_2^2\\ &= \|\mathbf{1}_{K\setminus W^{\#}}-\mathbf{v}_{U^{\#}}\|_2^2+\|\mathbf{v}_{W^{\#}\setminus T}\|_2^2\\ &+\|\mathbf{1}_{K\cap W^{\#}}-\mathbf{v}_{K\cap W^{\#}}\|_2^2\\ &\ge \|\mathbf{1}_{K\setminus W^{\#}}-\mathbf{v}_{U^{\#}}\|_2^2+\|\mathbf{v}_{W^{\#}\setminus T}\|_2^2\\ &\ge (1-R)^2\cdot|K\setminus W^{\#}|+R^2\cdot|W^{\#}\setminus K|\\ &\ge \min\{(1-R)^2,R^2\}(|K\setminus W^{\#}|+|W^{\#}\setminus K|)\\ &= \min\{(1-R)^2,R^2\}|K\triangle W^{\#}|.\end{aligned}$$

Therefore $\|\mathbf{1}_K-\mathbf{v}\|_2\le D$ implies $|K\triangle W^{\#}|\le\frac{D^2}{\min\{(1-R)^2,R^2\}}$ as desired. □

Theorem 3.3.3. *Suppose $T\subset C_1$. Then*

$$\frac{|C_1\triangle C_1^{\#}|}{|C_1|}\le o(1) \tag{3.9}$$

Proof. It is equivalent to show $|C_1\triangle C_1^{\#}|\le o(n_1)$. Note that $\mathbf{x}^*=\mathbf{1}_{C_1\setminus T}$. By Theorem 3.3.2, we get $\|\mathbf{1}_{C_1\setminus T}-\mathbf{x}^{\#}\|_2\le o(\|\mathbf{1}_{C_1\setminus T}\|_2)=o(\sqrt{n_1})$. We then apply Lemma 3.3.1 with $K=C_1\setminus T$, $W^{\#}=C_1^{\#}$, and $\mathbf{v}=\mathbf{x}^{\#}$ to get $|(C_1\setminus T)\triangle C_1^{\#}|\le o(n_1)$. Therefore $|C_1\triangle C_1^{\#}|\le o(n_1)$. □

3.4 Experiments

In this section, we evaluate Algorithm 8 on various synthetic and real datasets and compare its performance with several baselines. For all experiments, we perform 100 individual runs. Additional details about the experiments are provided in section 3.5. For reproducibility, we make our code available at: `https://github.com/zzzzms/LocalClustering`.

Datasets. We use simulated stochastic block model, simulated geometric data with three particular shapes, network data on political blogs [3], OptDigits[1], AT&T Database of Faces[2], MNIST[3], and USPS[4] as our benchmark datasets.

Baselines and Settings. We adopt the LSC [77], CP+RWT [70], HK-Grow [63], PPR [5], ESSC [143], LBSA [120], and several other modern semi-supervised clustering algorithms as our baseline methods. For our experiments of stochastic block model, the only target cluster is the most dominant cluster, i.e., the cluster with the highest connection probability. For all other experiments, all of the clusters are considered as our target clusters, and we apply CS-LCE iteratively to extract all of them. We use Jaccard index to measure the performance of one cluster tasks and use mean accuracy across all clusters to measure the performance of multiple clusters tasks.

3.4.1 Simulated Data

Symmetric Stochastic Block Model. The stochastic block model is a generative model for random graphs with certain edge densities within and between underlying clusters. The edges within clusters are denser than the edges between clusters. In the case of each cluster has the same size and the intra- and inter-connection probability are the same among all vertices, we have the symmetric stochastic block model $SSBM(n, k, p, q)$. The parameter n is the size of the graph, k is the number of clusters, p is the probability of intra-connectivity,

[1] `https://archive.ics.uci.edu/ml/datasets/optical+recognition+of+handwritten+digits`

[2] `https://git-disl.github.io/GTDLBench/datasets/att_face_dataset/`

[3] `http://yann.lecun.com/exdb/mnist/`

[4] `https://git-disl.github.io/GTDLBench/datasets/usps_dataset/`

and q is the probability of inter-connectivity. In our experiments, we fix $k = 3$ and vary n among $600, 1200, 1800, 2400, 3000$. We choose $p = 5 \log n/n, q = \log n/n$. With five labeled vertices as seeds, we achieve average Jaccard index and logarithm of average run time shown in Figure 3.1. We can see CS-LCE outperforms all other baselines with a reasonable running time.

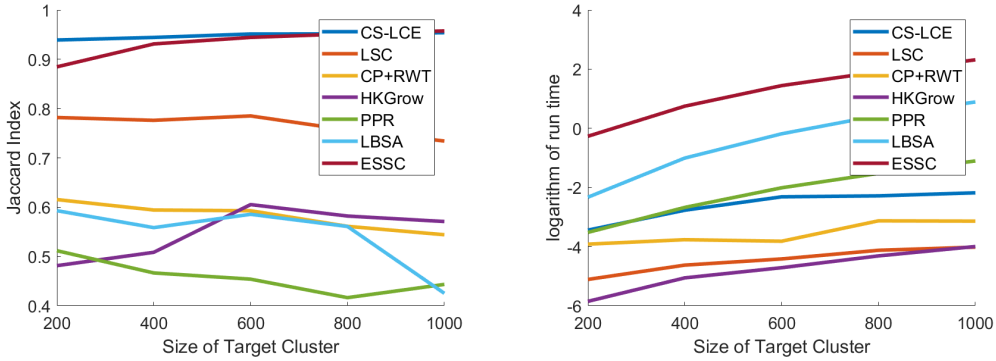


Figure 3.1: Jaccard Index and Logarthim of Running Time on SSBM.

General Stochastic Block Model. In a more general stochastic block model $SBM(\mathbf{n}, k, P)$, where n and k are the same as symmetric case. The matrix P indicates the connection probability within each individual cluster and between different clusters. In our experiments, we fix $k = 3$, and the size of clusters are chosen as $\mathbf{n} = (n_1, 2n_1, 5n_1)$ where n_1 is chosen from $\{200, 400, 600, 800, 1000\}$. We set the connection probability matrix $P = [p, q, q; q, p, q; q, q, p]$ where $p = \log^2(8n_1)/(8n_1)$ and $q = 5 \log(8n_1)/(8n_1)$. With five labeled vertices as seeds, the average Jaccard index and logarithm of average run time of CS-LCE compared with several other algorithms are shown in Figure 3.2.

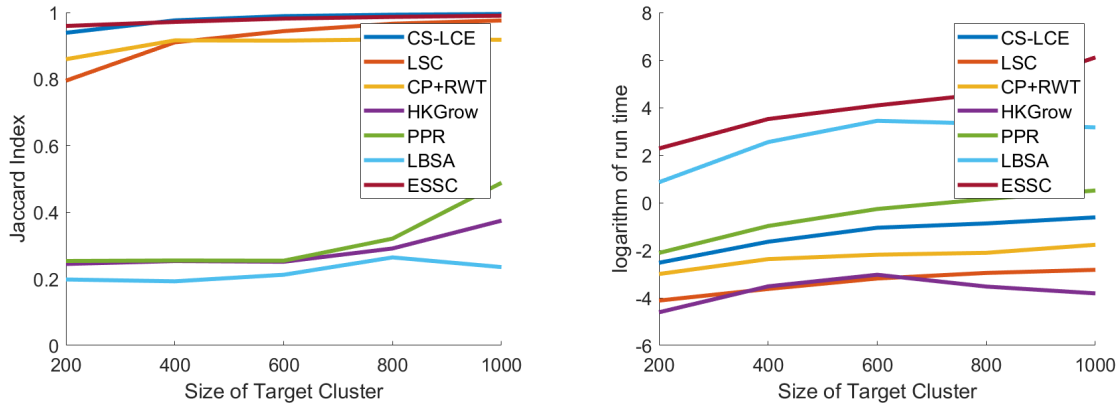


Figure 3.2: Jaccard Index and Logarthim of Running Time on SBM.

Geometric Data. We also simulated three high dimensional datasets in Euclidean space where the projections of the clusters onto two dimensional plane look like three lines, three circles, or three moons. See Figure 3.3 for an illustration of them. These datasets are often used as benchmark for data clustering and they are also described in [96] with slightly different parameters. Because of the shape of underlying clusters, traditional k-means clustering or spectral clustering fail on these contrived datasets. In our experiments, for each dataset, we randomly select 10 seeds for each of the cluster. The mean accuracy and standard deviation of CS-LCE compared with LSC [77] and CP+RWT [70] are given in Table 3.1. A more detailed description of this simulated dataset is given in section 3.5.

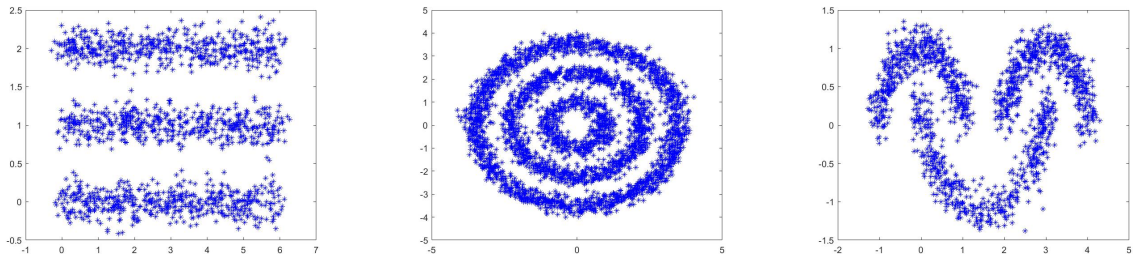


Figure 3.3: 2D Visualizations of Geometric Data.

Table 3.1: Mean Accuracy and Standard Deviation on Geometric Data (%)

Datasets	Three Lines	Three Circles	Three Moons
LSC	89.0 (5.53)	96.2 (3.71)	85.3 (1.88)
CP+RWT	82.1 (9.06)	96.1 (5.09)	85.4 (1.33)
CS-LCE	**92.4** (8.13)	**97.6** (4.69)	**96.8** (0.89)

3.4.2 Human Face Images

The AT&T Database of Faces contains gray-scale images for 40 different people of pixel size 92×112. Images of each person are taken under 10 different conditions, by varying the three perspectives of faces, lighting conditions, and facial expressions. We use part of this dataset by randomly selecting 10 people such that each individual is associated with 10 pictures of themselves. The selected dataset and desired recovery are shown in Figure 3.4.

The mean accuracy and standard deviation of CS-LCE compared with LSC [77], CP+RWT [70], and spectral clustering (SC) are summarized in Table 3.2. Note that spectral clustering method is unsupervised, hence its accuracy does not affected by the label ratios.

Figure 3.4: *Left*: Randomly Permuted AT&T Faces. *Right*: Desired Recovery of all Clusters.

Table 3.2: Mean Accuracy and Standard Deviation on AT&T Data (%)

Label Ratios	10 %	20 %	30 %
LSC	94.8 (3.32)	97.8 (1.18)	98.2 (0.77)
CP+RWT	93.7 (3.34)	97.8 (1.44)	98.3 (0.43)
SC	95.8 (0.00)	95.8 (0.00)	95.8 (0.00)
CS-LCE	**98.0** (1.90)	**99.1** (0.79)	**99.3** (0.59)

3.4.3 Network Data

"The political blogosphere and the 2004 US Election" [3] dataset contains a list of political blogs that were classified as liberal or conservative with links between blogs. An illustration of this dataset is shown in Figure 3.5. The state-of-the-art result on this dataset is given in [2]. Their simplified algorithm gave a successful classification 37 times out of 40 trials, and each of the successful trials correctly classified all but 56 to 67 of the 1,222 vertices in the graph main component.

In our experiments, given one labeled seed, CS-LCE succeeds 35 trials out of a total of 40 trials. Among these 35 successful trials, the average number of misclassified node in the graph main component is 49, which is comparable to the state-of-the-art result. We note that LSC [77] also succeeds 35 out of 40 trials, but the average number of misclassified node equals to 55. We also note that CP+RWT [70] fails on this dataset.

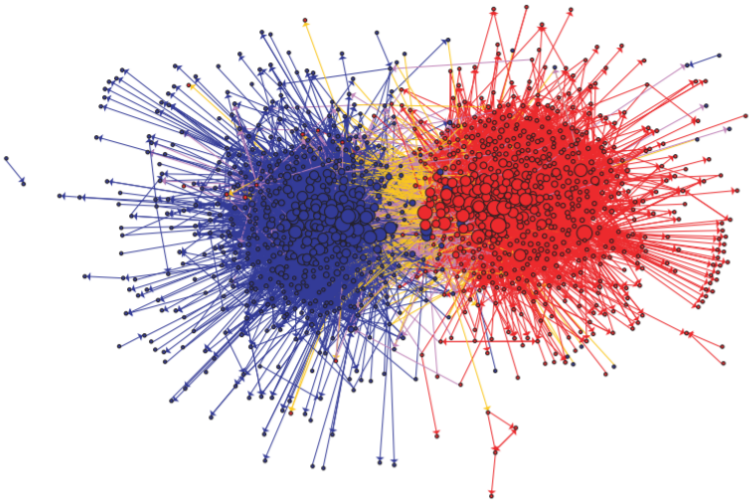


Figure 3.5: Community structure of political blogs. Red for conservative and blue for liberal. Orange links go from liberal to conservative, and purple ones from conservative to liberal. The size of each blog reflects the number of other blogs that link to it [3].

3.4.4 Digits Data

OptDigits. This dataset contains grayscale images of handwritten digits from 0 to 9 of size 8×8. There are a total of 5620 images and each cluster has approximately 560 images. The average Jaccard index and logarithm of average run time of CS-LCE compared with several other algorithms are shown in Figure 3.6. we exclude PPR and ESSC in the comparison as they either too slow to run or the accuracy is too low.

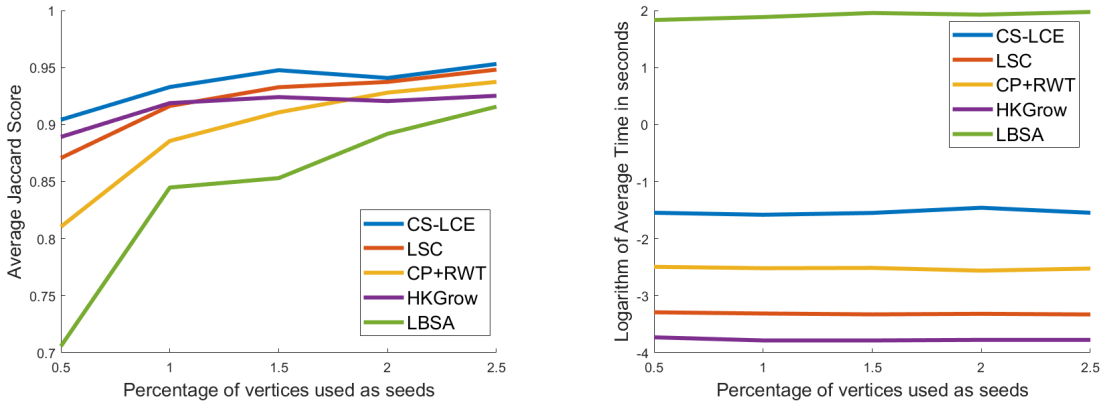


Figure 3.6: Jaccard Index and Logarthim of Running Time on OptDigits.

MNIST and USPS. The MNIST dataset consists of 70000 grayscale images of the handwritten digits 0-9 of size 28×28 with approximately 7000 images of each digit. The USPS data set contains 9298 grayscale images, obtained from the scanning of handwritten digits from envelopes by the U.S. postal service. We test CS-LCE, LCS, CP+RWT, and several other modern semi-supervised methods on these two datasets, the results are show in Table 3.3, Table 3.4, and Table 3.5. It is worth pointing out that in Table 3.3 and Table 3.4, we have only very few labeled data for our tasks. If one uses a neural network method to train for classification of images, then it usually needs more labeled data for training. In Table 3.5, we compare CS-LCE with several other constraint clustering algorithms. In each constrained clustering algorithms, the total number of pairwise constraints are set to equal to the total data points. Therefore in order to have a fair comparison, we choose a certain amount of labeled data in CS-LCE such that the total pairwise constraints are the same.

Table 3.3: Mean Accuracy and Standard Deviation on USPS (%)

Label Ratios	0.2 %	0.3%	0.4%
CP+RWT [70]	68.9 (3.17)	73.3 (2.76)	76.6 (2.59)
LSC [77]	72.3 (3.54)	77.1 (3.42)	80.4 (3.20)
CS-LCE [117]	**76.8** (3.37)	**80.1** (3.14)	**84.1** (2.53)

Table 3.4: Mean Accuracy and Standard Deviation on MNIST (%)

Label Ratios	0.05 %	0.10 %	0.15 %
CP+RWT [70]	74.1 (3.13)	79.7 (2.43)	85.0 (2.37)
LSC [77]	77.0 (3.47)	83.6 (2.76)	88.8 (2.52)
CS-LCE [117]	**85.3** (2.67)	**89.8** (1.91)	**93.2** (1.76)

Table 3.5: Mean Accuracy on MNIST and USPS (%)

	MNIST	USPS
KM-cst [9]	54.27	68.18
AE+KM [91]	74.09	70.28
AE+KM-cst [9]	75.98	71.87
DEC [144]	84.94	75.81
IDEC [50]	83.85	75.86
SDEC [113]	86.11	76.39
CS-LCE [117]	**96.02**	**82.10**

3.4.5 Image Color Clustering

Our approach also has applications in image color clustering. i.e., we can treat certain component in the image as a cluster and extract it out from the entire image. See Figure 3.7 as an example, where the "tomato" and "happy birthday" components are extracted from the two images respectively.

For our proposed local clustering algorithm, there are several directions which deserve more investigation. For example, is there a better way to choose a removal set T in Algorithm 8, and also how to extend the current method to the unsupervised setting, i.e., if no

seeds are given, how to select a small portion of seeds and make sure these seeds are all from the same cluster. We leave these to future work.

Figure 3.7: Examples of Image Colors Clustering

3.5 Further Details on Experiments and Implementation

Let us provide more details of the datasets usage, hyperparameters, and data preprocessing for the conducted experiments in this section.

3.5.1 Description of Geometric Data

Three Lines. The three lines are generated by sampling points uniformly at random in the two dimensional x-y plane where the x coordinate is between 0 and 6 and y coordinate equals to 0, 1, and 2 respectively. We draw 1200 points in each line to create three clusters. We then embed each data point into $\mathbb{R}^{100}$ by appending zeros and then adding Gaussian random

noise to each coordinate with mean 0 and standard deviation 0.15.

Three Circles. The three circles are generated by sampling points uniformly at random from three concentric circles of radii 1, 2.4 and 3.8 respectively. We draw around 500 points from the smallest circle, around 1200 points from the middle circle and around 1900 points from the largest circle (the numbers are chosen so that the total number of points is 3600). We then embed each data point into $\mathbb{R}^{100}$ by appending zeros and then adding Gaussian random noise to each coordinate with mean 0 and standard deviation 0.15.

Three Moons. The three moons are generated by sampling points uniformly at random from the upper semicircle of radius 1 centered at (0,0), the lower semi-circle of radius 1.5 centered at (1.5, 0.4) and the upper semi-circle of radius 1 centered at (3,0). We draw 1200 points in each semi-circle to create three clusters. We then embed each data point into $\mathbb{R}^{100}$ by appending zeros and then adding Gaussian random noise to each coordinate with mean 0 and standard deviation 0.15.

3.5.2 Hyperparameters Setup

For each cluster to be recovered, we sampled the seed vertices Γ_i uniformly from C_i for all of our implementations. We fix the rejection parameter $R = 0.1$, the random walk depth $t = 3$ and random walk threshold parameter $\epsilon = 0.8$ for all of our implementations. We fix the least squares threshold parameter with $\gamma = 0.2$ for all experiments. All the numerical experiments are implemented in MATLAB and can be run on a local machine.

3.5.3 Image Data Preprocessing

For our approach, the images data coming from each of the AT&T, OptDigits, MNIST, USPS dataset have to be firstly constructed into an auxiliary graph before feeding into the algorithm. We adopt the following way to build the auxiliary graphs.

Let $\mathbf{x}_i \in \mathbb{R}^n$ be the vectorization of an image from the original data set, we define the following affinity matrix of the K-NN auxiliary graph based on Gaussian kernel according

to [61] and [147],.

$$A_{ij} = \begin{cases} e^{-\|\mathbf{x}_i - \mathbf{x}_j\|^2/\sigma_i\sigma_j} & \text{if} \quad \mathbf{x}_j \in NN(\mathbf{x}_i, K) \\ 0 & \text{otherwise} \end{cases}$$

The notation $NN(\mathbf{x}_i, K)$ indicates the set of K-nearest neighbours of $\mathbf{x}_i$, and $\sigma_i := \|\mathbf{x}_i - \mathbf{x}_i^{(r)}\|$ where $\mathbf{x}_i^{(r)}$ is the r-th closest point of $\mathbf{x}_i$. Note that the above A_{ij} is not necessary symmetric, so we consider $\tilde{A}_{ij} = A^T A$ for symmetrization. Alternatively, one may also want to consider $\tilde{A} = \max\{A_{ij}, A_{ji}\}$ or $\tilde{A} = (A_{ij} + A_{ji})/2$. We use $\tilde{A}$ as the input adjacency matrix for our algorithms. In our implementation, we choose the local scaling parameters $K = 5$, $r = 3$ for AT&T faces images, and $K = 15$, $r = 10$ for OptDigits, MNIST, USPS.

Chapter 4

Function Approximation via Kolmogorov Superposition Theorem

In this chapter, we propose to use Kolmogorov superposition theorem (KST) to study the approximation rate of high dimensional continuous functions. More specifically, we show that there is a dense subclass of functions in $C([0,1]^d)$ which can be approximated by using the representation of KST with a dimension independent approximation rate $O(1/n)$, with n being the number of knots of the linear spline functions over $[0,1]$. Moreover, the approximation constant in our approach increases quadratically in the dimension d, and the number of parameters used in such neural network approximation equals is $O(nd)$. The results in this chapter are summarized in [76].

4.1 Kolmogorov Superposition Theorem

One of the suprising result in approximation theory is the Kolmogorov superposition Theorem (KST), sometimes it is also called Kolmogorov–Arnold representation theorem. Let us recall the statement of KST first. We will introduce two versions of KST which appear in [65] and [88], respectively.

Theorem 4.1.1 (Kolmogorov Superposition Theorem – original version [65])**.** *Let $f \in C([0,1]^d)$, then there exists continuous functions $g_q : \mathbb{R} \to \mathbb{R}$ and $\phi_{qp} : [0,1] \to \mathbb{R}$ such that*

$$f(x_1, \cdots, x_d) = \sum_{q=0}^{2d} g_q \left(\sum_{p=0}^{d} \phi_{qp}(x_p) \right). \tag{4.1}$$

The significance of this surprising result can be summarized succinctly: Addition is the only continuous multivariate function. There have been many improvements of KST over

the years. Lorentz [87] pointed out that the outer function g_q can be chosen to be the same, while Sprecher [126] showed that one can take $\phi_{qp} = \lambda_p \phi_q$. Henkin [53] and Fridman [44] pointed out that the inner functions ϕ_{qp} can be chosen to be Hölder continuous with exponent $\alpha \in (0, 1]$ and Lipschitz continuous, respectively. Sprecher [127, 128, 129, 130] also showed that inner functions can be replaced by one single inner function with an appropriate shift in its argument through the constructive form of KST. His proofs were finally correctly established in [11] and [12]. An excellent explanation of the history about the development of KST can be found in [102]. We now turn our attention to the Lorentz's version of KST [88], which is more useful for the development of our approach in function approximation.

Theorem 4.1.2 (Kolmogorov Superposition Theorem – Lorentz's version [88])**.** *There exist* $0 < \lambda_p \leq 1$, $p = 1, \cdots, d$, *and strictly increasing* α*-Hölder continuous functions* $\phi_q(x) : [0, 1] \to [0, 1]$, $q = 0, \cdots, 2d$, *with exponent* $\alpha \in (0, 1)$, *such that for every* $f \in C([0, 1]^d)$, *there exists a continuous function* $g \in C([0, d])$, *such that*

$$f(x_1, \cdots, x_d) = \sum_{q=0}^{2d} g\left(\sum_{p=1}^{d} \lambda_p \phi_q(x_p)\right). \tag{4.2}$$

Some notable features of the representation formula (4.2) are the following. Firstly, there is only one outer function g associated with f. Secondly, the number $2d+1$ in the summands can not be further reduced [107, 131]. Thirdly, the inner functions can not be chosen to be continuously differentiable [136, 137, 87].

The upshot for this representation is: for any continuous function $f \in C([0, 1]^d)$, there is a continuous function $g_f \in C([0, d])$ so that f can be represented by g_f via (4.2). Conversely, given any continuous function $g \in C([0, d])$, we can produce a continuous function $f_g \in C([0, 1]^d)$ by using the representation formula (4.2). Such a correspondence between f and g is one-to-one. Therefore we can use what we understand about univariate continuous functions to understand multivariate continuous functions.

It is worthy noting that KST also has some nice topology and machine learning interpretations. KST essentially established that all d dimensional compact metrizable spaces can

be embedded into $\mathbb{R}^N$ if and only if $N \geq 2d + 1$. KST also guarantees that any continuous statistical or machine learning model, after a suitable embedding, is a sum of generalized additive models. There have been many generalizations and extensions of KST over the past few decades. Ostrand [107] showed that KST holds on compact metric spaces. Doss [30] and Demko [24] extended KST to $\mathbb{R}^n$ for unbounded and bounded continuous functions, respectively. Feng [38] generalized KST to locally compact and finite dimensional separable metric spaces.

It is straightforward to see that the representation formula (4.2) mimics the structure of a two-layer neural network where the inner and outer functions can be considered as activation functions. However, there have been debates over decades on whether such a representation via KST is useful. Girosi and Poggio [46] claimed that some degree of smoothness is required for inner and outer functions in order for the approximation to generalize and stabilize against noise. Lin and Unbehauen [85] made a similar conclusion by noting that all information carried by f must be contained in the univariate function g hence learning the latter is not any easier than learning the former. On the other hand, Køurkovà [68, 69] countered some of the criticisms from Girosi and Poggio by giving a constructive way to approximate the univariate outer function g through linear combinations of the smooth sigmoid function. She also bounded the number of units needed for a desired approximation. This has in turn generated further interest in the study of neural network and approximation.

Indeed, KST has been actively studied which echoes the fast development of neural network computing [20, 97, 110]. Hecht-Nielsen [52] was among the first to draw a connection between KST and neural networks. This inspired much of the later works on universality of two-layer neural networks. However, Hecht-Nielsen was doubtful about the direct usefulness of this connection because no construction of the outer function was known then and he mentioned the possibility of learning the outer function from input-output examples. Later on, Igelnik and Parikh [60] proposed a neural network algorithm using spline functions to approximate both the inner and outer functions. More recently, active research has been

conducted on neural network approximation via KST and achieves promising results [100, 115, 36]. However, these results are not directly based on the representation formula (4.1.2) and can be impractical to implement in practice. To the best of the authors' knowledge, there is yet no approximation scheme that is directly based on KST and is straightforward to deploy in practice.

In this chapter, we propose to study the rate of approximation for ReLU neural network via KST. We propose a special neural network structure via the representation of KST, which can achieve a dimension independent approximation rate $O(1/n)$ with the approximation constant increasing quadratically in the dimension when approximating a dense subset of continuous functions. The number of parameters used in such a network increases linearly in n. Furthermore, we shall provide a numerical scheme to practically approximate d dimensional continuous functions by using at most $O(dn)$ number of pivotal locations for function value evaluattion instead of the whole equally-spaced $O(n^d)$ data locations, and such a set of pivotal locations are independent of target functions.

The subsequent sections of this chapter are structured as follows. In section 4.2, we introduce the neural network representation via KST and explain how to approximate multivariate continuous functions with no curse of dimensionality for a dense class of functions. We also establish the approximation result for any continuous function based on the modulus of continuity of the K-outer function. In section 4.3, we introduce KB-splines based on the spline approximation of K-outer function, and its smoothed version LKB-splines. We will show that KB-splines are indeed the bases for functions in $C([0,1]^d)$. In section 4.4, we numerically demonstrate in 2D and 3D that LKB-splines can approximate functions in $C([0,1]^d)$ very well. Furthermore, we provide a computational strategy based on matrix cross approximation to find a sparse solution using a few number of LKB-splines to achieve the same approximation order as the original approximation. This leads to the new concept of pivotal point set from any dense point set P over $[0,1]^d$ such that the discrete least squares

(DLS) fitting based on the pivotal point set has the similar rooted mean squares error to the DLS fitting based on the original data set P.

4.2 ReLU Network Approximation via KST

We will use σ_1 to denote ReLU function through the rest of discussion. It is easy to see that one can use linear splines to approximate K-inner (continuous and monotone increasing) functions $\phi_q, q = 0, \cdots, 2d$ and also approximate the K-outer (continuous) function g. We refer to Theorem 20.2 in [111]. On the other hand, we can easily see that any linear spline function can be written in terms of linear combination of ReLU functions and vice versa, see, e.g. [22, 25]. We shall include another proof later in this chapter. Hence, we have

$$L_q(t) := \sum_{j=1}^{N_q} c_{q,j}\sigma_1(t - y_{qj}) \approx \phi_q(t)$$

for $q = 0, \cdots, 2d$ and

$$S_g(t) := \sum_{k=1}^{N_g} w_k\sigma_1(t - y_k) \approx g,$$

where g is the K-outer function of a continuous function f. Based on KST and the universal approximation theorem [20, 56, 110], it follows that

Theorem 4.2.1 (Universal Approximation Theorem (cf. [123])**.** *Suppose that $f \in C([0,1]^d)$ is a continuous function. For any given $\epsilon > 0$, there exist coefficients $w_k, k = 1, \cdots, N_g$, $y_k \in [0, d], k = 1, \cdots, N_g$, $c_{q,j}, j = 1, \cdots, N_i$ and $y_{q,j} \in [0,1], j = 1, \cdots, N_i$ such that*

$$|f(x_1, \cdots, x_d) - \sum_{q=0}^{2d}\sum_{k=1}^{N_g} w_k\sigma_1(\sum_{i=1}^{d}\lambda_i\sum_{j=1}^{N_q} c_{q,j}\sigma_1(x_i - y_{qj}) - y_k)| \leq \epsilon. \tag{4.3}$$

In fact, many results similar to the above (4.3) have been established using other activation functions (cf. e.g. [20], [69], [100], and etc.).

4.2.1 K-Lipschitz Function

To establish the rate of convergence for Theorem 4.2.1, we introduce a new concept. For each continuous function $f \in C([0,1]^d)$, let g_f be the K-outer function associated with f. Let

$$KL = \{f : \text{ K-outer function } g_f \text{ is Lipschitz continuous}\} \tag{4.4}$$

be the class of K-Lipschitz continuous functions. Note that when f is a constant, its K-outer function $g = \frac{1}{d+1}f$ is also constant (cf. [13]) and hence, is Lipschitz continuous. That is, the function class KL is not empty. On the other hand, we can use any univariate Lipschitz continuous function g such as $g(t) = Ct, g(t) = \sin(Ct), g(t) = \exp(-Ct)$, $g(t) = \sin(Ct^2/2)$, etc.. over $[0,d]$ to define a multivariate function f by using the formula (4.2) of KST, where C is any constant. Then these newly defined f are continuous over $[0,1]^d$ and are belong to the function class KL. It is easy to see that the class KL is dense in $C([0,1]^d)$ by Weierstrass approximation theorem. See Theorem 4.2.2 below. For another example, let $g(t)$ be a B-spline function of degree $k \geq 1$, the associated multivariate function is in KL. We shall use such B-spline functions for the K-outer function g approximation in a later section.

Theorem 4.2.2. *For any $f \in C([0,1]^d)$ and any $\epsilon > 0$, there exists a K-Lipschitz continuous function K such that*

$$\|f - K\|_\infty \leq \epsilon. \tag{4.5}$$

Proof. By Kolmogorov superposition theorem, we can write

$$f(x_1, \cdots, x_d) = \sum_{q=0}^{2d+1} g(\sum_{i=1}^{d} \lambda_i \phi_q(x_i)).$$

By Weierstrass approximation theorem, there exists a polynomial p such that $|p(t) - g(t)| \leq \frac{\epsilon}{2d+1}$ for all $t \in [0,d]$. Such a polynomial p is certainly a Lipschitz continuous function over $[0,d]$.

Therefore, by Letting $K(x_1, \cdots, x_d) = \sum_{q=0}^{2d+1} p(\sum_{i=1}^{d} \lambda_i \phi_q(x_i)) \in KL$, we have

$$\begin{aligned} |f(x_1, \cdots, x_d) - K(x_1, \cdots, x_d)| &= |\sum_{q=0}^{2d+1} g(\sum_{i=1}^{d} \lambda_i \phi_q(x_i)) - \sum_{q=0}^{2d+1} p(\sum_{i=1}^{d} \lambda_i \phi_q(x_i))| \\ &\leq \sum_{q=0}^{2d+1} |g(\sum_{i=1}^{d} \lambda_i \phi_q(x_i)) - p(\sum_{i=1}^{d} \lambda_i \phi_q(x_i))| \\ &\leq (2d+1) \cdot \frac{\epsilon}{(2d+1)} = \epsilon. \end{aligned}$$

□

Note that the neural network being used for approximation in expression (4.3) is a special class of neural network with two hidden layers of widths $(2d+1)dN_q$ and $(2d+1)N_g$ respectively. Let us call this special class of neural networks the Kolmogorov network, or K-network in short and use $\mathcal{K}_{m,n}$ to denote the K-network of two hidden layers with widths $(2d+1)dm$ and $(2d+1)n$ based on ReLU activation function, i.e.,

$$\mathcal{K}_{m,n}(\sigma_1) = \{\sum_{q=0}^{2d} \sum_{k=1}^{dn} w_k \sigma_1(\sum_{i=1}^{d} \sum_{j=1}^{m} c_{qj} \sigma_1(x_i - y_{qj}) - y_k), w_k, c_{qj} \in \mathbb{R}, y_k \in [0, d], y_{qj} \in [0, 1]\}. \tag{4.6}$$

The parameters in $\mathcal{K}_{m,n}$ are w_k, y_k, $k = 1, \cdots, dn$, and c_{qj}, y_{qj}, $q = 0, \cdots, 2d$, $j = 1, \cdots, m$. Therefore the total number of parameters equals to $2dn + 2(2d+1)m$. In particular if $m = n$, the total number of parameters in this network is $(6d+2)n$. We are now ready to state one of the main results in this chapter.

Theorem 4.2.3. *Let $f \in C([0,1]^d)$. Suppose that f is in the KL class. Let C_f be the Lipschitz constant of the K-outer function associated with f. We have*

$$\inf_{s \in \mathcal{K}_{n,n}(\sigma_1)} \|f - s\|_{C([0,1]^d)} \leq \frac{C_f (2d+1)^2}{n}. \tag{4.7}$$

The significance of this result is, for a dense subclass of continuous functions, we need only $(6d+2)n$ parameters to achieve the approximation rate $O(1/n)$ with the approximation constant increasing quadratically in the dimension. That is, the curse of dimensionality is broken for functions in this dense subclass. In other words, the computation becomes

tractable. The above result improved the similar one in [100]. Also, the researchers in [101] showed that the KST can break the curse of dimension for band-limited functions. Our result breaks the curse of dimensionality for a different class of functions. On the other hand, many researchers used the smoothness of f to characterize the approximation of ReLU neural networks. See [89, 145, 146], where the approximation rate on the right-hand side of (4.7) is $O(n^{-s/d})$ with s being the smoothness of the function f. Their approximation rates suffer from the curse of dimension. In terms of the smoothness of the K-outer function, our result above is believed to be the correct rate of convergence. In addition, we shall extend the argument to the setting of K-Hölder continuous functions and present the convergence in terms of K-modulus of smoothness.

4.2.2 Proof of Theorem 4.2.3

To prove Theorem 4.2.3, we need some preparations. Let us begin with the space $\mathcal{N}(\sigma_1) = \text{span}\{\sigma_1(\mathbf{w}^\top \mathbf{x} - b), b \in \mathbb{R}, \mathbf{w} \in \mathbb{R}^d\}$ which is the space of shallow networks of ReLU functions. It is easy to see that all linear polynomials over $\mathbb{R}^d$ are in $\mathcal{N}(\sigma_1)$. The following result is known (cf. e.g. [22]). For self-containedness, we include a different proof.

Lemma 4.2.1. *For any linear polynomial s over $\mathbb{R}^d$, there exist coefficients $c_i \in \mathbb{R}$, bias $t_i \in \mathbb{R}$ and weights $\mathbf{w}_i \in \mathbb{R}^d$ such that*

$$s(\mathbf{x}) = \sum_{i=1}^{n} c_i \sigma_1(\mathbf{w}_i \cdot \mathbf{x} + t_i), \forall \mathbf{x} \in [0,1]^d. \tag{4.8}$$

That is, $s \in \mathcal{N}(\sigma_1)$.

Proof. It is easy to see that a linear polynomial x can be exactly reproduced by using the ReLU functions. For example,

$$x = \sigma_1(x), \forall x \in [0,1]. \tag{4.9}$$

Hence, any component x_j of $\mathbf{x} \in \mathbb{R}^d$ can be written in terms of (4.8). Indeed, choosing $\mathbf{w}_i = \mathbf{e}_j$, from (4.9), we have

$$x_j = \sigma_1(\mathbf{e}_j \cdot \mathbf{x}), \quad \mathbf{x} \in [0,1]^d, j = 1, \cdots, d.$$

Next we claim a constant 1 is in $\mathbb{N}(\sigma_1)$. Indeed, given a partition $\mathcal{P}_n = \{a = x_0 < x_1 < \cdots < x_n = b\}$ of interval $[a, b]$, let

$$h_0(x) = \begin{cases} \dfrac{x - x_1}{x_0 - x_1} & x \in [x_0, x_1] \\ 0 & x \in [x_1, x_n] \end{cases}, \tag{4.10}$$

$$h_i(x) = \begin{cases} \frac{x - x_{i-1}}{x_i - x_{i-1}} & x \in [x_{i-1}, x_i] \\ \frac{x - x_{i+1}}{x_i - x_{i+1}} & x \in [x_i, x_{i+1}] \\ 0 & x \notin [x_{i-1}, x_{i+1}] \end{cases}, \qquad i = 1, \cdots, n-1 \tag{4.11}$$

$$h_n(x) = \begin{cases} \dfrac{x - x_{n-1}}{x_n - x_{n-1}} & x \in [x_{n-1}, x_n] \\ 0 & x \in [x_0, x_{n-1}] \end{cases}. \tag{4.12}$$

be a set of piecewise linear spline functions over $\mathcal{P}_n$. Then we know $S_1^0(\mathcal{P}_n) = \text{span}\{h_i, i = 0, \cdots, n\}$ is a linear spline space. It is well-known that $1 = \sum_{i=0}^{n} h_i(x)$. Now we note the following formula:

$$h_i(x) = \frac{\sigma_1(x - x_{i-1})}{(x_i - x_{i-1})} + w_i \sigma_1(x - x_i) + \frac{\sigma_1(x - x_{i+1})}{(x_{i+1} - x_i)}, \tag{4.13}$$

where $w_i = -1/(x_i - x_{i-1}) - 1/(x_{i+1} - x_i)$. It follows that any spline function in $S_1^0(\mathcal{P}_n)$ can be written in terms of ReLU functions. In particular, we can write $1 = \sum_{i=0}^{n} h_i(x) = \sum_{i=0}^{n} c_i^0 \sigma_1(x - x_i)$ by using (4.13).

Hence, for any linear polynomial $s(\mathbf{x}) = a + \sum_{j=1}^{d} c_j x_j$, we have

$$s(\mathbf{x}) = a \sum_{i=0}^{n} c_i^0 \sigma_1(x - x_i) + \sum_{i=1}^{d} c_i \sigma_1(\mathbf{e}_j \cdot \mathbf{x}) \in \mathbb{N}(\sigma_1).$$

This completes the proof. □

The above result shows that any linear spline is in the ReLU neual networks. Also, any ReLU neual network in $\mathbb{R}^1$ is a linear spline. We are now ready to prove Theorem 4.2.3. We begin with the standard modulus of continuity. For any continuous function $g \in C[0, d])$, we

define the modulus of continuity of g by

$$\omega(g,h) = \max_{\substack{x\in[0,d]\\ 0<t\le h}} |g(x+t)-g(x)| \tag{4.14}$$

for any $h>0$. To prove the result in Theorem 4.2.3, we need to recall some basic properties of linear splines (cf. [116]). The following result was established in [111].

Lemma 4.2.2. *For any function in $f \in C[a,b]$, there exists a linear spline $S_f \in S_1^0(\triangle)$ such that*

$$\|f - S_f\|_{\infty,[a,b]} \le \omega(f, |\triangle|) \tag{4.15}$$

where $S_1^0(\triangle)$ is the space of all continuous linear splines over the partition $\triangle = \{a = t_0 < t_1 < \cdots < t_n = b\}$ with $|\triangle| = \max_i |t_i - t_{i-1}|$.

In order to know the rate of convergence, we need to introduce the class of function of bounded variation. We say a function f is of bounded variation over $[a,b]$ if

$$\sup_{\forall a=x_0<x_1<\cdots<x_n=b} \sum_{i=1}^{n} |f(x_i) - f(x_{i-1})| < \infty$$

We let $V_a^b(f)$ be the value above when f is of bounded variation. The following result is known (cf. [116])

Lemma 4.2.3. *Suppose that f is of bounded variation over $[a,b]$. For any $n \ge 1$, there exists a partition $\triangle$ with n knots such that*

$$dist(f, S_1^0(\triangle))_\infty = \inf_{s\in S_1^0(\triangle)} \|f-s\|_\infty \le \frac{V_a^b(f)}{n+1}.$$

Let $L = \{f \in C([0,1]^d) : |f(\mathbf{x}) - f(\mathbf{y})| \le L_f|\mathbf{x}-\mathbf{y}|, \forall \mathbf{x},\mathbf{y} \in [0,1]^d\}$ be the class of Lipschitz continuous functions. We can further establish

Lemma 4.2.4. *Suppose that f is Lipschitz continuous over $[a,b]$ with Lipschitz constant L_f. For any $n \ge 1$, there exists a partition $\triangle$ with n interior knots such that*

$$dist(f, S_1^0(\triangle))_\infty \le \frac{L_f(b-a)}{2(n+1)}.$$

Proof. We use a linear interpolatory spline S_f. Then for $x \in [x_i, x_{i+1}]$,

$$\begin{aligned} f(x) - S_f(x) &= f(x) - f(x_i)\frac{x_{i+1}-x}{x_{i+1}-x_i} - f(x_{i+1})\frac{x-x_i}{x_{i+1}-x_i}. \\ &= \frac{(x_{i+1}-x)(f(x)-f(x_i))}{x_{i+1}-x_i} + \frac{(x-x_i)(f(x)-f(x_{i+1}))}{x_{i+1}-x_i} \\ &\leq L_f\frac{(x-x_i)(x_{i+1}-x)}{x_{i+1}-x_i} + L_f\frac{(x_{i+1}-x)(x-x_i)}{x_{i+1}-x_i} \leq \frac{L_f}{2}(x_{i+1}-x_i). \end{aligned}$$

Hence, $|f(x) - S_f(x)| \leq L_f(b-a)/(2(n+1))$ if $x_{i+1} - x_i = (b-a)/(n+1)$. This completes the proof. □

Furthermore, if f is Lipschitz continuous, so is the linear interpolatory spline S_f. In fact, we have

$$|S_f(x) - S_f(y)| \leq 2L_f|x-y|. \tag{4.16}$$

We are now ready to prove one of our main results in this chapter.

Proof of Theorems 4.2.3. Since ϕ_q are univariate increasing functions mapping from $[0,1]$ to $[0,1]$, they are bounded variation with $V_0^1(\phi_q) \leq 1$. By Lemma 4.2.3, there are linear spline functions L_q such that $|L_q(t) - \phi_q(t)| \leq 1/(n+1)$ for $q = 0, \cdots, 2d$.

For K-outer function g, when g is Lipschitz continuous, there is a linear spline S_g with dn distinct interior knots over $[0,d]$ such that

$$\sup_{t\in[0,d]} |g(t) - S_g(t)| \leq \frac{dC_g}{2(nd+1)} \leq \frac{C_g}{2n},$$

where C_g is the Lipschitz constant of g by using Lemma 4.2.4. Now we first have

$$|f(\mathbf{x}) - \sum_{q=0}^{2d} S_g(\sum_{i=1}^{d} \lambda_i\phi_q(x_i))| \leq \sum_{q=0}^{2d} |g(\sum_{i=1}^{d} \lambda_i\phi_q(x_i)) - S_g(\sum_{i=1}^{d} \lambda_i\phi_q(x_i))| \leq \frac{(2d+1)C_g}{2n}.$$

Next since g is Lipschitz continuous, so is S_g. Thus, by (4.16) and Lemma 4.2.3, we have

$$|S_g(\sum_{i=1}^{d} \lambda_i\phi_q(x_i)) - S_g(\sum_{i=1}^{d} \lambda_i L_q(x_i))| \leq 2C_g \sum_{i=1}^{d} \lambda_i|\phi_q(x_i) - L_q(x_i)| \leq \frac{d\cdot 2C_g}{(n+1)}.$$

Let us put the above estimates together to have

$$|f(\mathbf{x}) - \sum_{q=0}^{2d} S_g(\sum_{i=1}^{d} \lambda_i L_q(x_i))| \leq \frac{(2d+1)C_g}{2n} + \frac{(2d+1)2dC_g}{(n+1)} \leq \frac{(2d+1)^2C_g}{n}.$$

The conclusion of Theorem 4.2.3 follows. □

4.2.3 Functions Beyond the K-Lipschitz Class

As K-outer function g may not be Lipschitz continuous, we next consider a class of functions which is of Hölder continuity. Letting $\alpha \in (0, 1]$, we say g is in $C^{0,\alpha}$ if

$$\sup_{x,y\in[0,d]} \frac{|g(x) - g(y)|}{|x - y|^\alpha} \leq L_\alpha(g) < \infty. \tag{4.17}$$

Using such a continuous function g, we can define a multivariate continuous function f by using the formula in Theorem 4.1.2. Let us extend the analysis of the proof of Lemma 4.2.4 to have

Lemma 4.2.5. *Suppose that g is Hölder continuous over $[0, d]$, say $g \in C^{0,\alpha}$ with $L_\alpha(g)$ for some $\alpha \in (0, 1]$. For any $n \geq 1$, there exists a partition $\triangle$ with n interior knots such that*

$$dist(g, S_1^0(\triangle))_\infty \leq \frac{L_\alpha(g)d^\alpha}{2(n + 1)^\alpha}.$$

Similarly, we can define a class of functions which is K-Hölder continuous in the sense that K-outer function g is Hölder continuity $\alpha \in (0, 1)$. For each univariate g in $C^{0,\alpha}([0, d])$, we define f using the KST formula (4.2). Then we have a new class of continuous functions which will satisfy (4.18). The proof is a straightforward generalization of the one for Theorem 4.2.3, we leave it to the interested readers.

Theorem 4.2.4. *For each continuous function $f \in C([0, 1]^d)$, let g be the K-outer function associated with f. Suppose that g is in $C^{0,\alpha}([0, d])$ for some $\alpha \in (0, 1]$. Then*

$$\inf_{s\in\mathcal{K}_{n,n}(\sigma_1)} \|f - s\|_{C([0,1]^d)} \leq \frac{(2d + 1)^2 L_\alpha(g)}{n^\alpha}. \tag{4.18}$$

Finally, in this section, we study the K-modulus of continuity. For any continuous function $f \in C([0, 1]^d)$, let g_f be the K-outer function of f based on the KST. Then we use $\omega(g_f, h)$ which is called the K-modulus of continuity of f to measure the smoothness of g_f. Due to the uniform continuity of g_f, we have linear spline S_{g_f} over an equally-spaced knot sequence such that

$$|g_f(t) - S_{g_f}(t)| \leq \omega(g_f, h), \quad \forall t \in [0, d] \tag{4.19}$$

for any $h > 0$, e.g. $h = 1/n$ for a positive integer n. It follows that

$$|g_f(\sum_{i=1}^{d} \lambda_i \phi_q(x_i)) - S_{g_f}(\sum_{i=1}^{d} \lambda_i \phi_q(x_i)| \leq \omega(g_f, h), \tag{4.20}$$

for any $(x_1, \cdots, x_d) \in [0,1]^d$. Since $\phi_q, q = 0, \cdots, 2d$ are monotonically increasing, we use Lemma 4.2.3 to have linear splines L_q such that $|L_q(t) - \phi_q(t)| \leq h$ since $V_0^1(\phi_q) \leq 1$. We now estimate

$$|S_{g_f}(\sum_{i=1}^{d} \lambda_i \phi_q(x_i)) - S_{g_f}(\sum_{i=1}^{d} \lambda_i L_q(x_i))| \tag{4.21}$$

for $q = 0, \cdots, 2d$. Note that

$$|\sum_{i=1}^{d} \lambda_i \phi_q(x_i) - \sum_{i=1}^{d} \lambda_i L_q(x_i)| \leq \sum_{i=1}^{d} |\phi_q(x_i) - L_q(x_i)| \leq dh.$$

The difference of the above two points in $[0, d]$ is separated by at most d subintervals with length h and hence, we will have

$$|S_{g_f}(\sum_{i=1}^{d} \lambda_i \phi_q(x_i)) - S_{g_f}(\sum_{i=1}^{d} \lambda_i L_q(x_i))| \leq 2d \cdot \omega(g_f, h) \tag{4.22}$$

since S_{g_f} is a linear interpolatory spline of g_f. It follows that

$$\begin{aligned}
& |f(x_1, \cdots, x_n) - \sum_{q=0}^{2d} S_{g_f}(\sum_{i=1}^{d} \lambda_i L_q(x_i))| \\
\leq & \sum_{q=0}^{2d} |g_f(\sum_{i=1}^{d} \lambda_i \phi_q(x_i)) - S_{g_f}(\sum_{i=1}^{d} \lambda_i \phi_q(x_i))| + \sum_{q=0}^{2d} |S_{g_f}(\sum_{i=1}^{d} \lambda_i \phi_q(x_i)) - S_{g_f}(\sum_{i=1}^{d} \lambda_i L_q(x_i))| \\
\leq & (2d+1)\omega(g_f, h) + (2d+1)2d \cdot \omega(g_f, h).
\end{aligned}$$

Therefore, we conclude the following theorem.

Theorem 4.2.5. *For any continuous function $f \in C[0,1]^d$, let g_f be the K-outer function associated with f. Then*

$$\inf_{s \in \mathcal{K}_{n,n}(\sigma_1)} \|f - s\|_{C([0,1]^d)} \leq (2d+1)^2 \omega(g_f, 1/n). \tag{4.23}$$

However, it is not easy to see if the K-outer function g_f is Lipschitz continuous when given a continuous functions f. To do so we have to compute g_f from f first. To this end, we implemented Lorentz's constructive proof of KST in MATLAB by following the steps in pages $168-174$ in [88]. See [13] for another implementation based on Maple and MATLAB. We noticed that the curve g_f behaviors very badly for many smooth functions f. Even if f is a linear polynomial in the 2-dimensional space, the K-outer function g still behaviors very widey although we can use K-network with two hidden layers to approximate this linear polynomial f arbitrarily well in theory. This may be a big hurdle to prevent researchers in [46], [52], [69], [60], [13], and etc. from successful applications based on Kolmogorov spline network. We circumvent the difficulty of having such a wildly behaved K-outer function g by introducing KB-splines and the denoised counterpart LKB-splines in this section. In addition, we will explain how to use them to well approximate high dimensional functions in a later section..

First of all, we note that the implementation of these $\phi_q, q = 0, \cdots, 2d$ is not easy. Numerical ϕ_q's are not accurate enough. Indeed, letting $z_q(x_1, \cdots, x_d) = \sum_{i=1}^{d} \lambda_i \phi_q(x_i)$, Consider the transform:

$$T(x_1, \cdots, x_d) = (z_0, z_1, \cdots, z_{2d}) \tag{4.24}$$

which maps from $[0,1]^d$ to $\mathbb{R}^{2d+1}$. Let $Z = \{T(x_1, \cdots, x_d), (x_1, \cdots, x_d) \in [0,1]^d\}$ be the image of $T([0,1]^d) \subset \mathbb{R}^{2d+1}$. It is easy to see that the image is closed. The theory in [88] explains that the map T is one-to-one and continuous. As the dimension of Z is much larger than d, the map T is like a well-known Peano curve which maps from $[0,1]$ to $[0,1]^2$ and hence, the implementation of T, i.e., the implementation of ϕ_q's is not possible to be accurate. However, we are able to compute these ϕ_q and decompose g such that the reconstruction of constant function is exact. Let us present two examples to show that our numerical implementation is reasonable. For convenience, let us use images as 2D functions and compute their K-outer functions g and then reconstruct the images back. In Figure 4.1, we can see that the

reconstruction is very good visually although K-outer functions g are oscillating very much. It is worthwhile to note that such reconstruction results have also been reported in [13]. Certainly, these images are not continuous functions and hence we do not expect that g to be Lipschitz continuous. But these reconstructed images serves as a "proof" that our computational code works numerically.

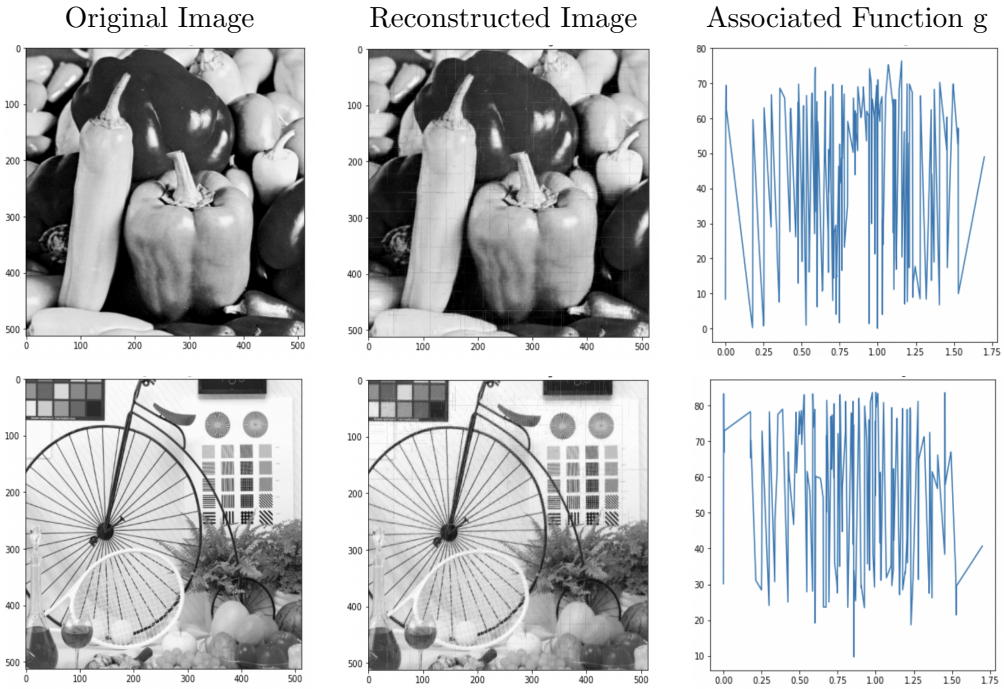


Figure 4.1: Original image (left column), reconstructed image (middle column), and associated K-outer function g (right column).

Next we present a few examples of smooth functions whose K-outer functions may not be Lipschitz continuous in Figure 4.2. Note that the reconstructed functions are very noisy, in fact they are too noisy to believe that the implementation of the KST can be useful. In order to see that these noisy functions are indeed the original functions, we applied a penalized least squares method based on bivariate spline method (to be explained later in the chapter). That is, after denoising, the reconstructed functions are very close to the exact original functions as shown in Figure 4.2. That is, the denoising method is successful which motivates us to adopt this approach to approximate any continuous functions.

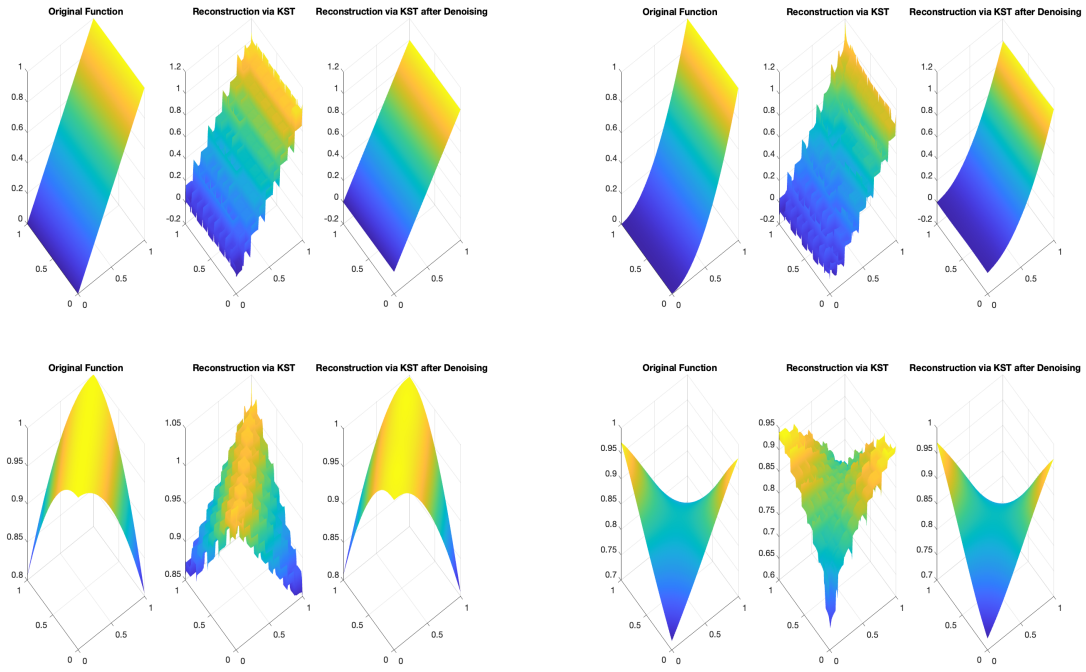


Figure 4.2: Top left: reconstruction of $f(x,y) = x$. Top right: reconstruction of $f(x,y) = x^2$. Bottom left: reconstruction of $f(x,y) = \cos(2(x-y)/\pi)$. Bottom right: reconstruction of $f(x,y) = \sin(1/(1+(x-0.5)(y-0.5)))$.

4.3.1 KB-splines

To this end, we first use standard uniform B-splines to form some subclasses of K-Lipschitz continuous functions. Let $\triangle_n = \{0 = t_1 < t_2 < \cdots < t_{dn} < d\}$ be a uniform partition of interval $[0,d]$ and let $b_{n,i}(t) = B_k(t - t_i), i = 1, \cdots, dn$ be the standard B-splines of degree k with $k \geq 1$. For simplicity, we only explain our approach based on linear B-splines for the theoretical aspect while using other B-splines (e.g. cubic B-splines) for the numerical experiments. We define KB-splines by

$$KB_{n,j}(x_1, \cdots, x_d) = \sum_{q=0}^{2d} b_{n,j}\left(\sum_{i=1}^{d} \lambda_i \phi_q(x_i)\right), j = 1, \cdots, dn. \tag{4.25}$$

It is easy to see that each of these KB-splines defined above is nonnegative. Due to the property of B-splines: $\sum_{i=1}^{dn} b_{n,i}(t) = 1$ for all $t \in [0,d]$, we have the following property of KB-splines:

Theorem 4.3.1. *We have* $\sum_{i=1}^{dn} KB_{n,i}(x_1, \cdots, x_d) = 1$ *and hence,* $0 \leq KB_{n,i} \leq 1$.

Proof. The proof is immediate by using the fact $\sum_{i=1}^{dn} b_{n,i}(t) = 1$ for all $t \in [0, d]$. □

Remark 4.3.1. *The property in Theorem 4.3.1 is called the partition of unit which makes the computation stable. We note that a few of these dn KB-splines will be zero since* $0 < \lambda_i \leq 1$ *and* $\min\{\lambda_i, i = 1, \cdots, d\} < 1$. *The number of zero KB-splines is dependent on the choice of* λ_i, $i = 1, \cdots, d$.

Another important result is that these KB-splines are linearly independent.

Theorem 4.3.2. *The nonzero KB-splines* $\{KB_{n,j} \neq 0, j = 1, \cdots, dn\}$ *are linearly independent.*

Proof. Suppose there are c_j, $j = 1, 2, \cdots, dn$ such that $\sum_{j=1}^{dn} c_j KB_{n,j}(x_1, \cdots, x_d) = 0$ for all $(x_1, \cdots, x_d) \in [0, 1]^d$. Then we want to show $c_j = 0$ for all $j = 1, 2, \cdots, dn$. Let us focus on the case $d = 2$ as the proof for general case d is similar. Suppose $n > 0$ is a fixed integer and we use the notation $z_q = \sum_{i=1}^{2} \lambda_i \phi_q(x_i)$ as above. Then based on the graphs of ϕ_q in Figure 4.3, we can choose $x_1 = \delta$ and $x_2 = 0$ with $0 < \delta \leq 1$ small enough such that $KB_{n,j}(x_1, 0) = \sum_{q=0}^{4} b_j(z_q(\delta, 0)) = 0$ for all $j = 3, 4, \cdots, 2n$. Therefore in order to show the linear independence of $KB_{n,j}, j = 1, 2, \cdots, 2n$, it is suffices to show $\sum_{j=1}^{2} c_j KB_{n,j}(x_1, x_2) = 0$ implies $c_1 = c_2 = 0$. Let us confine $x_1 \in [0, \delta]$ and $x_2 = 0$. Then we have

$$0 = c_1 KB_{n,1}(x_1, x_2) + c_2 KB_{n,2}(x_1, x_2) = c_1(\sum_{q=0}^{4} b_1(z_q)) + c_2(\sum_{q=0}^{4} b_2(z_q))$$

$$= c_1(\sum_{q=0}^{4} b_1(z_q)) + c_2(5 - \sum_{q=0}^{4} b_1(z_q)) = (c_1 - c_2)(\sum_{q=0}^{4} b_1(z_q)) + 5c_2,$$

where we have used the fact that $b_1(x) + b_2(x) = 1$ over $[0, 1/n]$. Since c_2 is constant, and $\sum_{q=0}^{4} b_1(z_q)$ is not constant when x_1 varies between 0 and δ, we must have $c_1 = c_2$. Hence $c_2 = 0$ and therefore $c_1 = 0$.

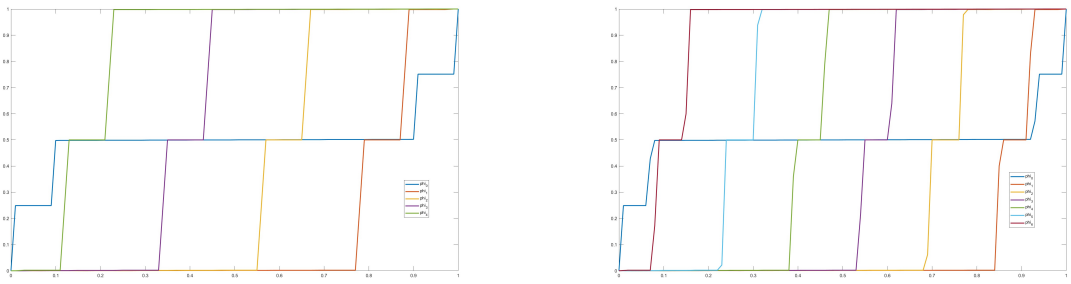


Figure 4.3: Left: ϕ_q, $q = 0, 1, 2, 3, 4$, for 2D. Right: ϕ_q, $q = 0, 1, 2, 3, 4, 5, 6$, for 3D.

In the same fashion, we can choose $\tilde{x}_1$ and $\tilde{\delta}$ such that $KB_{n,j} = \sum_{q=0}^{4} b_j(z_q(\tilde{x}_1, 0)) = 0$ for all $j = 1, 2, \cdots, 2n$ except for $j = k, k+1$. By the similar argument as above, we have $c_k = c_{k+1} = 0$. By varying k between 1 and $2n$, we get $c_j = 0$ for all $j = 1, 2, \cdots, n$. □

Since span$\{b_{n,i}, i = 1, \cdots, nd\}$ will be dense in $C[0, d]$ when $n \to \infty$, we can conclude that span$\{KB_{n,j}, j = 1, \cdots, nd\}$ will be dense in $C([0, 1]^d)$. That is, we have

Theorem 4.3.3. *The KB-splines $KB_{n,j}(x_1, \cdots, x_d), j = 1, \cdots, nd$, are dense in $C([0, 1]^d)$ when $n \to \infty$.*

Proof. For any continuous function $f \in C([0, 1]^d)$, let $g_f \in C[0, d])$ be the K-outer function of f. For any $\epsilon > 0$, there is an integer $n > 0$ and a spline $S_{g_f} \in$ span$\{b_{n,i}, i = 1, \cdots, dn\}$ such that

$$\|g_f(t) - S_{g_f}(t)\|_\infty \leq \epsilon/(2d + 1)$$

for all t. Writing $S_{g_f}(t) = \sum_{i=1}^{dn} c_i(f) b_{n,i}(t)$, we have

$$\begin{aligned} & |f(x_1, \cdots, x_d) - \sum_{i=1}^{dn} c_i(f) KB_{n,i}(x_1, \cdots, x_d)| \\ = \; & |\sum_{q=0}^{2d} g(z_q(x_1, \cdots, x_d)) - \sum_{i=1}^{dn} c_i(f) \sum_{q=0}^{2d} b_{n,i}(z_q(x_1, \cdots, x_d))| \\ \leq \; & \sum_{q=0}^{2d} |g(z_q(x_1, \cdots, x_d)) - S_{g_f}(z_q(x_1, \cdots, x_d))| \leq (2d + 1)\epsilon/(2d + 1) = \epsilon. \end{aligned}$$

This completes the proof. □

4.3.2 LKB-splines

However, in practice, the KB-splines obtained in (4.25) are very noisy due to any implementation of ϕ_q's as we have explained before that the functions $z_q, q = 0, \cdots, 2d$, like Peano's curve. One has no way to have an accurate implementation. As demonstrated before, our denoising method can help. We shall call LKB-splines after denoising KB-splines.

Let us explain a multivariate spline method for denoising for $d = 2$ and $d = 3$. In general, we can use tensor product B-splines for denoising for any $d \geq 2$ which is the similar to what we are going to explain below. For convenience, let us consider $d = 2$ and let $\triangle$ be a triangulation of $[0, 1]^2$. For any degree $D \geq 1$ and smoothness $r \geq 1$ with $r < D$, let

$$S_D^r(\triangle) = \{s \in C^r([0, 1]^2) : s|_T \in \mathbb{P}_D, T \in \triangle\} \tag{4.26}$$

be the spline space of degree D and smoothness r with $D > r$. We refer to [73] for a theoretical detail and [6] for a computational detail. For a given data set $\{(x_i, y_i, z_i), i = 1, \cdots, N\}$ with $(x_i, y_i) \in [0, 1]^2$ and $z_i = f(x_i, y_i) + \epsilon_i, i = 1, \cdots, N$ with noises ϵ_i which may not be very small, the penalized least squares method (cf. [71] and [74]) is to find

$$\min_{s \in S_5^1(\triangle)} \sum_{i=1,\cdots,N} |s(x_i, y_i) - z_i|^2 + \lambda \mathcal{E}_2(s) \tag{4.27}$$

with $\lambda \approx 1$, where $\mathcal{E}_2(s)$ is the thin-plate energy functional defined as follows.

$$\mathcal{E}_2(s) = \int_\Omega |\frac{\partial^2}{\partial x^2} s|^2 + 2|\frac{\partial^2}{\partial x \partial y} s|^2 + |\frac{\partial^2}{\partial y^2} s|^2. \tag{4.28}$$

Multivariate splines have been studied for several decades and they have been used for data fitting (cf. [71], [74], [79], and [141]), numerical solution of partial differential equations (see, e.g. [72]). and data denoising (see, e.g. [79]).

We now explain that the penalized least squares method can produce a good smooth approximation of the given data. For convenience, let $S_{f,\epsilon}$ be the minimizer of (4.27) and write $\|f\|_{\mathcal{P}} = \sqrt{\frac{1}{N} \sum_{i=1}^N |f(x_i, y_i)|^2}$ is the rooted mean squares (RMS). If $f \in C^2([0, 1]^2)$, we have the following

Theorem 4.3.4. *Suppose that f is twice differentiable over $[0,1]^2$. Let $S_{f,\epsilon}$ be the minimizer of (4.27). Then we have*

$$\|f - S_{f,\epsilon}\|_{\mathcal{P}} \leq C\|f\|_{2,\infty}|\triangle|^2 + 2\|\epsilon\|_{\mathcal{P}} + \sqrt{\lambda \mathcal{E}_2(f)} \tag{4.29}$$

for a positive constant C independent of f, degree d, and triangulation $\triangle$.

To prove the above result, let us recall the following minimal energy spline $S_f \in S_5^1(\triangle)$ of data function f: letting $\triangle$ be a triangulation of $[0,1]^2$ with vertices $(x_i, y_i), i = 1, \cdots, N$, S_f is the solution of the following minimization:

$$\min_{S_f \in S_5^1(\triangle)} \{\mathcal{E}_2(S_f) : \quad S_f(x_i, y_i) = f(x_i, y_i), (x_i, y_i), i = 1, \cdots, N.\} \tag{4.30}$$

Then it is known that S_f approximates f very well if $f \in C^2([0,1]^2)$. We have

Theorem 4.3.5 (von Golitschek, Lai and Schumaker, 2002 [138])**.** *Suppose that $f \in C^2([0,1]^2)$. Then*

$$\|S_f - f\|_\infty \leq C\|f\|_{2,\infty}|\triangle|^2 \tag{4.31}$$

for a positive constant C independent of f and $\triangle$, where $\|f\|_{2,\infty}$ denotes the maximum norm of the second order derivatives of f over $[0,1]^2$ and $\|S_f - f\|_\infty$ is the maximum norm of $S_f - f$ over $[0,1]^2$.

of Theorem 4.3.4. Recall that $S_{f,\epsilon}$ is the minimizer of (4.27). We now use S_f to have

$$\begin{aligned}
\|f - S_{f,\epsilon}\|_{\mathcal{P}} &\leq \|z - S_{f,\epsilon}\|_{\mathcal{P}} + \|\epsilon\|_{\mathcal{P}} \leq \sqrt{\|z - S_{f,\epsilon}\|_{\mathcal{P}}^2 + \lambda \mathcal{E}_2(S_{f,\epsilon})} + \|\epsilon\|_{\mathcal{P}} \\
&\leq \sqrt{\|z - S_f\|_{\mathcal{P}}^2 + \lambda \mathcal{E}_2(S_f)} + \|\epsilon\|_{\mathcal{P}} \leq \|f - S_f\|_{\mathcal{P}} + 2\|\epsilon\|_{\mathcal{P}} + \sqrt{\lambda \mathcal{E}_2(f)} \\
&\leq C\|f\|_{2,\infty}|\triangle|^2 + 2\|\epsilon\|_{\mathcal{P}} + \sqrt{\lambda \mathcal{E}_2(f)},
\end{aligned}$$

where we have used a fact that $\mathcal{E}_2(S_f) \leq \mathcal{E}_2(f)$ which can be found in [138]. These complete the theorem of this section. □

If f is C^2 smooth, then $S_{f,\epsilon}$ will be a good approximation of f when the size $|\triangle|$ of triangulation is small, the thin plate energy $\mathcal{E}_2(f)$ with $\lambda > 0$ is small, and the noises $\|\epsilon\|_{\mathcal{P}}$

is small even though a few individual noises ϵ_i can be large. Note also that ϵ can be made small by increasing the accuracy of the implementaion of ϕ_q.

Now let us illustrate some examples of KB-splines and LKB-splines in Figure 4.4. One can see that the KB-splines are continuous but not smooth functions at all, while the LKB-splines are. These LKB-splines are much smoother and nicer, therefore can be used to approximate high dimensional continuous functions accurately. We leave the numerical results to the next chapter.

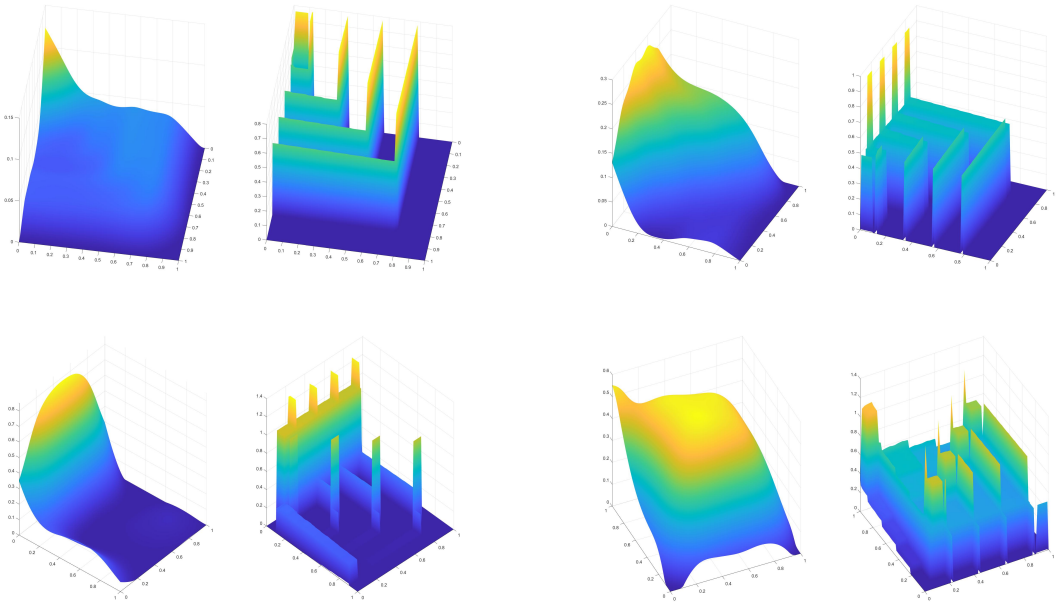


Figure 4.4: Some examples of linear LKB-splines (the first and third columns) which are the smoothed version of the corresponding linear KB-splines (the second and fourth columns).

4.4 Numerical Results for LKB-splines based Approximation in 2D and 3D

In this section, we will first demonstrate numerically that LKB-splines can approximate general continuous functions well based on $O(n^d)$ equally-spaced sampled data locations. Then we use the matrix cross approximation technique to show that there are at most $O(nd)$

locations among those $O(n^d)$ locations are pivotal. Therefore, we only need the function values at those pivotal locations in order to achieve a reasonable good approximation.

4.4.1 Numerical results based on $O(n^d)$ data points

Let us design numerical experiments to demonstrate the power of LKB-splines for approximating functions in $C([0,1]^d)$. We choose 41^d equally-spaced points $\mathbf{x}_i \in [0,1]^d$. For any continuous function $f \in C([0,1]^d)$, we use the function values at these data locations to find an approximation $F_n = \sum_{j=1}^{dn} c_j LKB_{n,j}$ by using discrete least squares (DLS) method. In other words, we solve the following minimization problem:

$$\min_{c_j} \|f - \sum_{j=1}^{dn} c_j LKB_{n,j}\|_{\mathcal{P}}, \tag{4.32}$$

where $\|f\|_{\mathcal{P}}$ is the RMS semi-norm based on the function values f over these 41^d sampled data points in $[0,1]^d$. We shall report the accuracy $\|f - F_n(f)\|_{\mathcal{PP}}$, where $\|f\|_{\mathcal{PP}}$ is the RMS semi-norm based on 101^d function values. The current computational power enables to do the numerical experiments for $d = 2$ with $n = 100, 200, \cdots, 10,000$ and for $d = 3$ with $n = 100, 200, \cdots, 1000$.

For $d = 2$, we choose the following 10 testing functions across different families of continuous functions to check the computational accuracy.

$$\begin{aligned}
f_1 &= (1 + 2x + 3y)/6; \qquad f_2 = (x^2 + y^2)/2; \qquad f_3 = xy; \\
f_4 &= (x^3 + y^3)/2; \qquad f_5 = 1/(1 + x^2 + y^2); \\
f_6 &= \cos(1/(1 + xy)); \qquad f_7 = \sin(2\pi(x + y)); \\
f_8 &= \sin(\pi x)\sin(\pi y); \qquad f_9 = \exp(-x^2 - y^2); \\
f_{10} &= \max(x - 0.5, 0)\max(y - 0.5, 0);
\end{aligned}$$

For $d = 3$, we choose the following 10 testing functions across different families of continuous functions to check the computational accuracy. The computational results are reported in Tables 4.1 and 4.2 (those columns associated with 41^3).

$$f_1 = (1 + 2x + 3y + 4z)/10; \qquad f_2 = (x^2 + y^2 + z^2)/3; \qquad f_3 = (xy + yz + zx)/3;$$

$$
\begin{aligned}
f_4 &= (x^3y^3 + y^3z^3)/2; \qquad f_5 = (x+y+z)/(1+x^2+y^2+z^2);\\
f_6 &= \cos(1/(1+xyz)); \qquad f_7 = \sin(2\pi(x+y+z));\\
f_8 &= \sin(\pi x)\sin(\pi y)\sin(\pi z); \qquad f_9 = \exp(-x^2-y^2-z^2);\\
f_{10} &= \max(x-0.5,0)\max(y-0.5,0)\max(z-0.5,0);
\end{aligned}
$$

Remark 4.4.1. *The major computational burden for the results in Tables 4.1 and 4.2 is the denoise of the KB-splines to get LKB-splines which requires a large amount of data points and values as the noises are everwhere over* $[0,1]^d$*. When dimension* $d \gg 1$ *gets large, one has to use an exponentially increasing number of points and KB-spline values by, say a tensor product spline method for denoising, and hence, the computational cost will suffer the curse of dimensionality. However, the denoising step can be pre-computed once for all the approximation tasks. That is, once we have the LKB-splines, the rest of the computational cost is no more than the cost of solving a least squares problem.*

This shows the power of LKB-splines in approximating general continuous functions. However, the number of data points being sampled in order to achieve such approximation error is $O(n^d)$. Therefore, when d gets large, we still need exponentially many sampled data. Our final goal in this chapter is to reduce this amount of data values. In fact, we only need at most $O(nd)$ number of sampled data instead of $O(n^d)$ in order to achieve the same order of approximation accuracy. Let us further explain our study in the next subsection.

Table 4.1: RMSEs (computed based on 101^2 equally-spaced locations) of the DLS fitting (4.32) based on 41^2 equally-spaced location and pivotal location in 2D.

	$n = 100$		$n = 1000$		$n = 10000$	
# sampled data	41^2	54	41^2	105	41^2	521
f_1	1.67e-05	2.60e-05	5.79e-06	1.02e-05	5.14e-07	1.21e-06
f_2	4.19e-04	8.92e-04	1.17e-04	2.61e-04	2.83e-05	6.62e-05
f_3	1.09e-04	2.19e-04	3.57e-05	7.46e-05	2.20e-05	5.67e-05
f_4	7.67e-04	1.70e-03	2.10e-04	5.11e-04	4.99e-05	1.11e-04
f_5	2.28e-04	5.04e-04	6.69e-05	1.47e-04	1.93e-05	4.08e-05
f_6	2.52e-04	6.51e-04	7.97e-05	1.94e-04	1.43e-05	2.73e-05
f_7	7.05e-02	1.32e-01	7.80e-03	2.25e-02	1.30e-03	3.30e-03
f_8	1.50e-03	2.29e-03	3.73e-04	1.01e-03	1.69e-04	4.37e-04
f_9	3.49e-04	7.97e-04	8.25e-05	1.98e-04	2.48e-05	5.52e-05
f_{10}	2.02e-03	3.79e-03	7.77e-04	1.82e-03	1.68e-04	4.10e-04

Table 4.2: RMSEs (computed based on 101^3 equally-spaced locations) of the DLS fitting (4.32) based on 41^3 equally-spaced location and pivotal location in 3D.

	$n = 100$		$n = 300$		$n = 1000$	
# sampled data	41^3	178	41^3	331	41^3	643
f_1	8.27e-06	2.25e-05	1.51e-06	4.20e-06	3.62e-07	7.48e-07
f_2	4.42e-05	1.68e-04	8.14e-06	2.18e-05	1.87e-06	4.11e-06
f_3	1.24e-05	3.79e-05	3.77e-06	9.41e-06	1.22e-06	2.53e-06
f_4	2.93e-04	5.60e-04	1.43e-04	2.55e-04	1.16e-04	2.63e-04
f_5	1.31e-04	3.46e-04	9.09e-05	1.66e-04	6.61e-05	1.20e-04
f_6	1.24e-04	3.22e-04	7.02e-05	1.34e-04	5.18e-05	1.09e-04
f_7	1.65e-02	5.29e-02	1.15e-02	1.71e-02	1.10e-02	1.85e-02
f_8	2.47e-03	8.28e-03	9.60e-04	1.94e-03	7.20e-04	1.19e-03
f_9	1.43e-04	3.84e-04	1.14e-04	2.01e-04	9.84e-05	3.95e-04
f_{10}	3.21e-04	9.74e-04	2.31e-04	4.00e-04	2.04e-04	3.91e-04

4.4.2 The pivotal data locations for breaking the curse of dimensionality

For convenience, let us use M to indicate the data matrix associated with the discrete least squares problem (4.32). In other words, for $1 \le j \le dn$, the jth column $M(:, j)$ consists of $\{LKB_{n,j}(\mathbf{x}_i)\}$ where $\mathbf{x}_i \in [0, 1]^d$ are those 41^d equally-spaced sampled points in 2D or 3D. Clearly, the experiment above requires 41^d data values which suffers from the curse of dimensionality. However, we in fact do not need such many data values. The main reason is that the data matrix M has many zero columns or near zero columns due to the fact that for many $i = 1, \cdots, nd$, the locations from 41^d equally-spaced points do not fall into the support of linear B-splines $b_{n,i}(t)$, $t \in [0, d]$, based on the map z_q. The structures of M are shown in Figure 4.5 for the case of $n = 1000$ when $d = 2$.

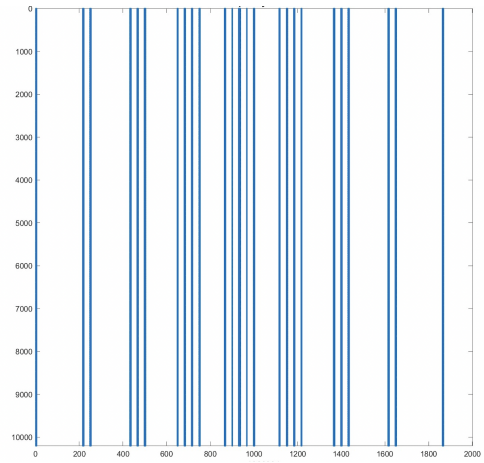


Figure 4.5: The sparsity pattern of data matrix for $n = 1000$.

That is, there are many columns in M whose entries are zero or near zero. Therefore, there exists a sparse solution to the discrete least squares fitting problem. We adopt the well-known orthogonal matching pursuit (OMP) (cf. e.g. [80]) to find a solution. For convenience, let us explain the sparse solution technique as follows. Over those 41^d points $\mathbf{x}_i \in [0,1]^d$, the columns in the matrix

$$M = [LKB_{n,j}(\mathbf{x}_i)]_{i=1,\cdots,41^d, j=1,\cdots,dn} \tag{4.33}$$

are not linearly independent. Let Φ be the normalized matrix of M in (4.33) and $\mathbf{b} = [f(\mathbf{x}_i)], i = 1, \cdots, 41^d$. Write $\mathbf{c} = (c_1, \cdots, c_{dn})^\top$, we look for

$$\min \|\mathbf{c}\|_0 : \Phi\mathbf{c} = \mathbf{b} \tag{4.34}$$

where $\|\mathbf{c}\|_0$ stands for the number of nonzero entries of $\mathbf{c}$. See many numerical methods in ([80]). The near zero columns in Φ also tell us that the data matrix associated with (4.32) of size $41^d \times dn$ is not full rank $r < dn$. The LKB-splines associated with these near zero columns do not play a role. Therefore, we do not need all dn LKB-splines. Furthermore, let us continue to explain that many data locations among these 41^d locations do not play an essential role.

To this end, we use the so-called matrix cross approximation (see [49, 48, 98, 45, 67, 4] and the literature therein). Let $r \geq 1$ be a rank of the approximation. It is known (cf. [49])

that when $M_{I,J}$ of size $r \times r$ has the maximal volume among all submatrices of M of size $r \times r$, we have

$$\|M - M_{:,J} M_{I,J}^{-1} M_{I,:}\|_C \leq (1+r)\sigma_{r+1}(M), \tag{4.35}$$

where $\|\cdot\|_C$ is the Chebyshev norm of matrix and $\sigma_{r+1}(M)$ is the $r+1$ singular value of M, $M_{I,:}$ is the row block of M associated with the indices in I and $M_{:,J}$ is the column block of M associated with the indices in J. The volume of a square matrix A is the absolute value of the determinant of A.

One mainly needs to find a submatrix $M_{I,J}$ of M such that $M_{I,J}$ has the maximal volume among all $r \times r$ submatrices of M. In practice, we use the concept called dominant matrix to replace the maximal volume. There are several algorithms, e.g. maxvol algorithm, available in the literature (cf. [48]). Recall that originally we need to solve the DLS problem $M\mathbf{x}_f \approx \mathbf{f} = [\mathbf{f}_I; \mathbf{f}_{I^c}]$. However, these greedy based maxvol algorithms enable us to find a good submatrix $M_{I,J}$ and solve a much simpler discrete least squares problem $M_{I,J}\widehat{\mathbf{x}} \approx \mathbf{f}_I$. According to Theorem 8 in [4], that $\widehat{\mathbf{x}}$ is a good approximation of $\mathbf{x}_f$. Let us call the data locations associated with row indices I the pivotal data locations. The pivotal data locations are shown in Figure 4.6. It is worthwhile to point out that such a set of pivotal locations is only dependent on the partition between $[0, d]$ when numerically build KB-splines, the sampled data when constructing LKB-splines, and the smoothing parameters for converting KB-splines to LKB-splines. However, such set of pivotal locations is independent of the target function to approximate.

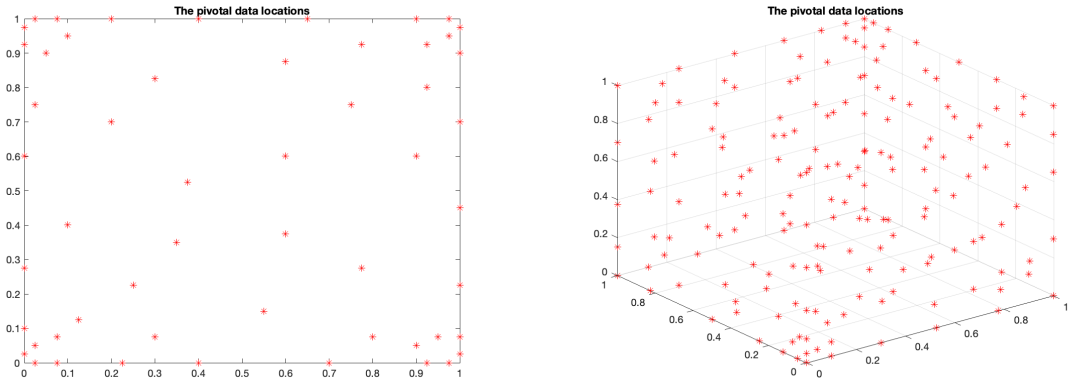


Figure 4.6: Pivotal data at 54 locations (selected from 41^2 equally-spaced locations) in 2D and 178 locations (selected from 41^3 equally-spaced locations) in 3D for $n = 100$.

We now present some numerical results in Table 4.1 and 4.2 (those columns associated with $54, 105, 521$ and $178, 331, 643$) to demonstrate that the numerical approximation results based on pivotal data locations have the same order as the results based on data locations sampled on the uniform grid. We therefore conclude that the curse of dimensionality for 2D and 3D function approximation can be overcome if we use LKB-splines with pivotal data sets.

To verify the approximation order $O(1/n)$ in Theorem 4.2.3, we plot the approximation errors of several aforementioned functions based on pivotal point locations against n using log-log scale. The results are shown in Figure 4.7. It is worthwhile to note that the slopes in these plots are associated with the exponent α in Theorem 4.2.4. In other words, if the slope of an convergence plot for a function is smaller than -1, then we can numerically conclude that such a function belongs to the class of K-Lipschitz continuous functions. If the slope α satisfies $-1 < \alpha < 0$, then we can numerically conclude that such a function belongs to the class of K-Hölder continuous functions with the outer function g belongs to $C^{0,\alpha}$. In other words, our computational method provide with a numerical approach to check if a multidimensional continuous function is K-Lipschitz or not.

Finally, in Figure 4.8, we plot the number of pivotal locations against n, we can see that the number of pivotal locations increasing linearly in n and the increasing rates (slopes) are at most d. That is, we only need $O(nd)$ data locations and $O(nd)$ LKB functions to approximate a multidimensional continuous function f with approximation rate $O(1/n)$ when f is K-Lipschitz continuous. Therefore, the curse of dimensionality is overcome when $d = 2$ and $d = 3$. We leave the numerical evidence for $d = 4$ or larger to the next chapter.

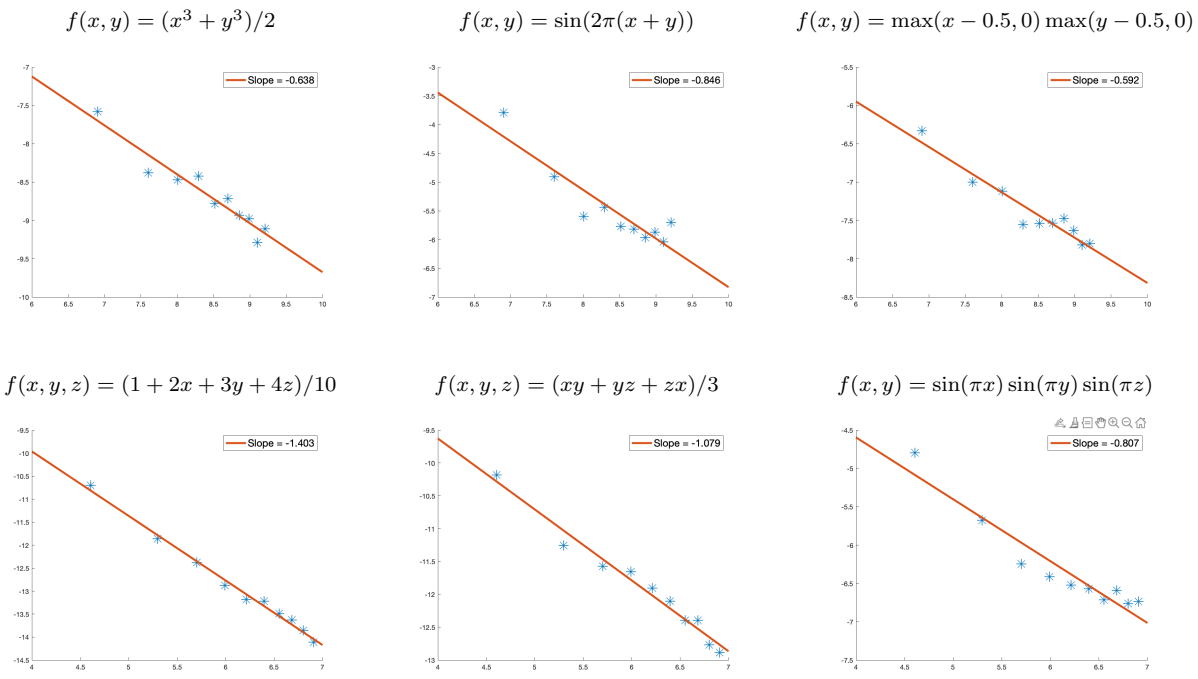


Figure 4.7: Plot of Convergence Rate using Log-log Scale for Functions in 2D and 3D.

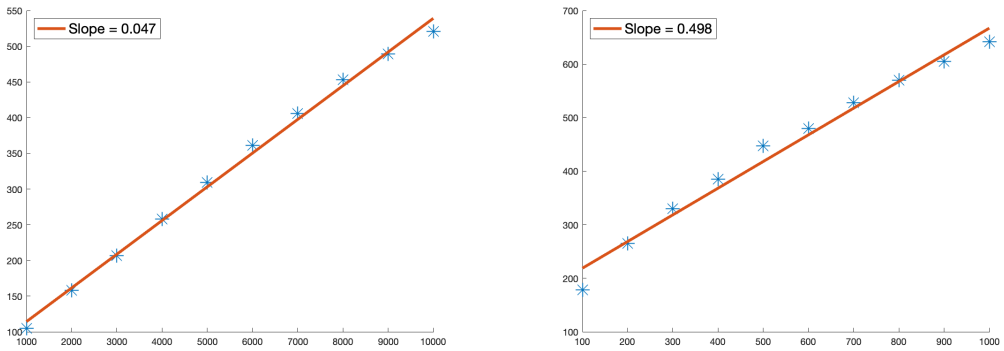


Figure 4.8: Number of Pivotal Locations (vertical axis) against n (horizontal axis) in 2D (left) and 3D (right).

Finally, let us end up this section with some important remarks.

Remark 4.4.2. *The pivotal data set depends on the degree of KB-splines and the LKB-splines which are dependent on the smoothing parameters and triangulation for converting KB-splines to LKB-splines. After LKB-splines are constructed, the pivotal point set is dependent on the discrete least squares (DLS) fitting method. For example, if we use randomly sampled points over* $[0,1]^d$ *for a DLS method instead of equally-space points, the pivotal point set is clearly different from the pivotal points based on the equally-spaced points over* $[0,1]^d$*. Even if we use* 81×81 *equally-spaced points instead of* 41×41 *equally-spaced points when constructing a DLS fitting based on LKB-splines over* $[0,1]^2$*, the location of the pivotal data are different and the size of pivotal data set is slightly bigger than the ones shown in Figure 4.6. Although we can apply this trick to find a pivotal point set for any discrete least squares method, it is the K-outer function* g *defined on* $[0,d]$ *which enables us to approximate the* g *using* nd *LKB-splines based on the pivotal point set with cardinality at most* nd *to achieve the rate* $O(1/n)$ *of approximation.*

Remark 4.4.3. *Certainly, there are many functions such as* $f(x,y) = \sin(100x)\sin(100y)$ *or* $f(x,y) = \tanh(100((2x-1)^2 + (2y-1)^2 - 0.25))$ *which the LKB-splines can not approximate well based on the pivotal data sets above. Such highly oscillated functions are hard to*

approximate even using other methods. One indeed needs a lot of the data (points and the function values over the points) in order to approximate them well. One may also consider to use Fourier basis as K-outer functions rather than B-splines basis to approximate such highly oscillated trigonometric functions via KST. We leave it as a future research topic.

Remark 4.4.4. *To reproduce the experimental results in this chapter, we uploaded our MATLAB codes in* `https://github.com/zzzms/KST4FunApproximation`. *In fact, we have tested more than 100 functions in 2D and 3D with pivotal data sets which enables us to approximate these functions very well.*

Chapter 5

The Optimal Approximation Rate Based on Linear LKB-splines

In this chapter, we extend the results obtained in the previous chapter and discuss how to achieve the best approximation rate for high dimemsional continuous functions based on the linear spline approximation of the K-outer function. More specifically, we show that there is a dense subclass in $C([0,1]^d)$ which can be approximated by using the representation of KST with a dimension independent approximation rate $O(1/n^2)$, with n being the number of knots of the over $[0,1]$. Moreover, the approximation constant in our approach increases linearly in the dimension d, and the number of parameters used in such neural network approximation equals is $O(nd)$. Finally, we show an application of our approach by solving Poisson equation numerically. The results in this chapter is summarized in [78].

5.1 K-Hölder Functions

We will consider a general class of continuous functions called K-Hölder class. Let us call the function g and ϕ_q in (4.1.2) the K-outer function and K-inner function respectively. For each continuous function $f \in C([0,1]^d)$, let

$$K_\beta := \{ f : \text{ K-outer function } g \text{ is } \beta\text{-Hölder continuous with exponent } 0 < \beta < 1 \} \quad (5.1)$$

be the class of K-Hölder continuous functions with exponent β. Recall that we say a function f is in $C^{0,\alpha}$ if

$$\sup_{x,y\in[0,d]} \frac{|f(x)-f(y)|}{|x-y|^\alpha} < \infty. \quad (5.2)$$

One can easily show

Theorem 5.1.1. *Under the KST representation (4.2). Suppose $f \in K_\beta$, then $f \in C^{0,\alpha\beta}$.*

Proof. Suppose $\mathbf{x}, \mathbf{y} \in [0,1]^d$, and $(x_1, \cdots, x_d) = \mathbf{x} \neq \mathbf{y} = (y_1, \cdots, y_d)$. Then

$$\begin{aligned}
\left|f(\mathbf{x}) - f(\mathbf{y})\right| &= \left|\sum_{q=0}^{2d} g\left(\sum_{i=1}^{d} \lambda_i \phi_q(x_i)\right) - \sum_{q=0}^{2d} g\left(\sum_{i=1}^{d} \lambda_i \phi_q(y_i)\right)\right| \\
&\leq \sum_{q=0}^{2d} \left|g\left(\sum_{i=1}^{d} \lambda_i \phi_q(x_i)\right) - g\left(\sum_{i=1}^{d} \lambda_i \phi_q(y_i)\right)\right| \\
&\leq \sum_{q=0}^{2d} C_1 \left|\sum_{i=1}^{d} \lambda_i \phi_q(x_i) - \sum_{i=1}^{d} \lambda_i \phi_q(y_i)\right|^{\beta} \leq \sum_{i=0}^{2d} C_1 \sum_{i=1}^{d} \lambda_i^{\beta} \left|\phi_q(x_i) - \phi_q(y_i)\right|^{\beta} \\
&\leq \sum_{q=0}^{2d} C_1 \sum_{i=1}^{d} \lambda_i^{\beta} C_2^{\beta} |x_i - y_i|^{\alpha\beta} \leq (2d+1) C_1 C_2^{\beta} \sum_{i=1}^{d} |x_i - y_i|^{\alpha\beta}.
\end{aligned}$$

This completes the proof. □

Now let us introduce two important subclasses of K-Hölder continuous functions: K-polynomials and KB-splines.

5.1.1 K-polynomials and K-monomials

Let us define the K-polynomial as

$$Kp_n(x_1, \cdots, x_d) = \sum_{q=0}^{2d+1} p_n \left(\sum_{i=1}^{d} \lambda_i \phi_q(x_i)\right), \tag{5.3}$$

where the function p_n is a univariate polynomial. We call it a K-monomial if $p_n(t) := t^n$, $n \geq 0$. Figure 5.1 shows some plots of different K-monomials with and without using the denoising/smoothing technique described in [76]. The significance of K-monomials are that the $span\{Kp_n\}_{n\geq 0}$ are dense in $C([0,1]^d)$. Let us call this result the K-Weierstrass theorem.

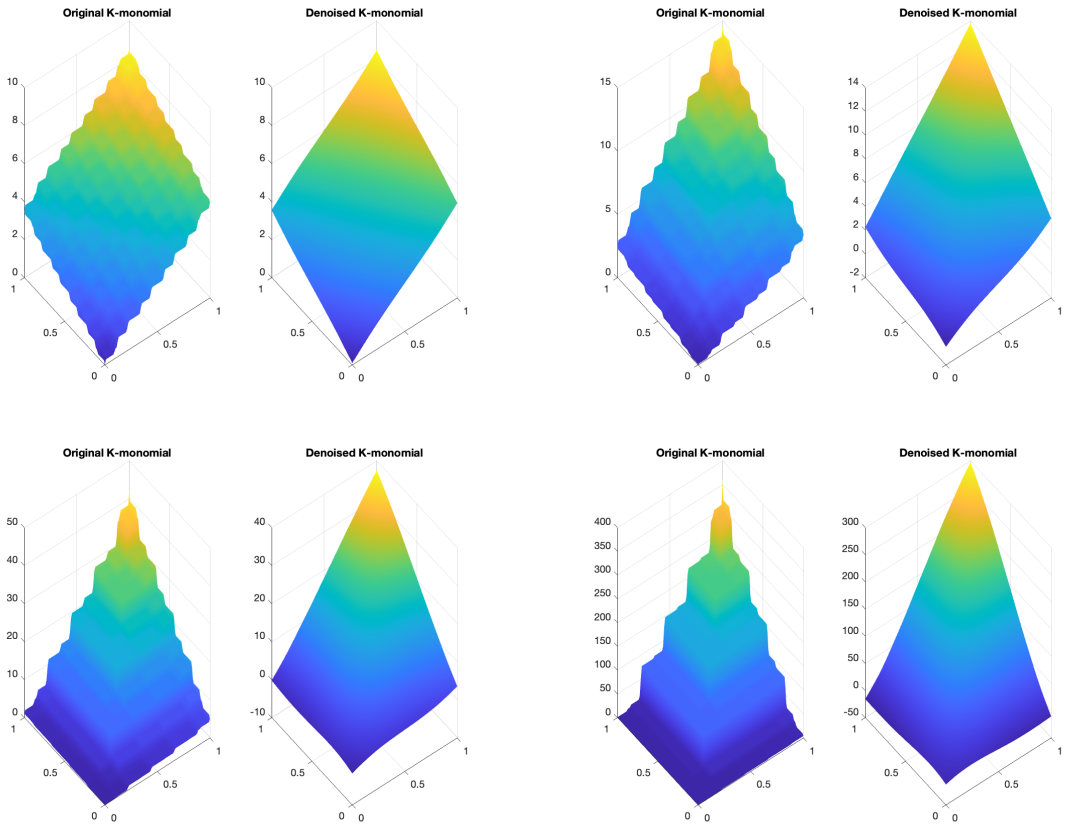


Figure 5.1: Examples of K-monomials (Top Row: $p_n(x) = x, x^2$. Bottom Row: $p_n(x) = x^4, x^8$).

Theorem 5.1.2 (K-Weierstrass Theorem)**.** *For any $f \in C([0,1]^d)$ and for any $\epsilon > 0$, there exists $K \in span\{Kp_n\}_{n\geq 0}$ such that*

$$\|f - K\|_\infty \leq \epsilon. \tag{5.4}$$

Proof. By Kolmogorov superposition theorem, we can write

$$f(x_1, \cdots, x_d) = \sum_{q=0}^{2d+1} g\left(\sum_{i=1}^{d} \lambda_i \phi_q(x_i)\right).$$

By Weierstrass theorem, there exists a polynomial p such that $|p(t) - g(t)| \leq \frac{\epsilon}{2d+1}$ for all $t \in [0, d]$. By letting $K(x_1, \cdots, x_d) = \sum_{q=0}^{2d+1} p\left(\sum_{i=1}^{d} \lambda_i \phi_q(x_i)\right) \in span\{Kp_n\}_{n\geq 0}$, we have

$$\begin{aligned}
|f(x_1, \cdots, x_d) - K(x_1, \cdots, x_d)| &= \left| \sum_{q=0}^{2d+1} g\left(\sum_{i=1}^{d} \lambda_i \phi_q(x_i)\right) - \sum_{q=0}^{2d+1} p\left(\sum_{i=1}^{d} \lambda_i \phi_q(x_i)\right) \right| \\
&\leq \sum_{q=0}^{2d+1} \left| g\left(\sum_{i=1}^{d} \lambda_i \phi_q(x_i)\right) - p\left(\sum_{i=1}^{d} \lambda_i \phi_q(x_i)\right) \right| \\
&\leq (2d+1) \cdot \frac{\epsilon}{(2d+1)} = \epsilon.
\end{aligned}$$

This completes the proof. □

Remark 5.1.1. *There are many $f \in C([0,1]^d)$ are K-Hölder continuous. Indeed, in addition to K-polynomials, we can use trigonmetric functions as K-outer function g to define high dimensional continuous functions called K-trigonometric functions via (4.2). Similarly, we can have K-exponential functions, K-logarithmic functions, etc,. In fact, any univariate Hölder continuous function g gives a K-Hölder continuous function f via Kolmogorov representation formula by using Theorem 4.1.2. Because these univariate functions g are of Hölder continuous, their corresponding f are in the K-Hölder continuous class.*

5.1.2 Linear KB-splines and LKB-splines

It is well known that linear spline function can be represented in terms of linear combination of ReLU functions and vice versa, see, e.g. [22], and [25]. Let $S_1^0(\triangle)$ be the space of all continuous linear splines over the partition $\triangle = \{0 = t_0 < t_1 < \cdots < t_n = 1\}$ with $|\triangle| = \max_i |t_i - t_{i-1}|$. For univariate function f, let $\omega(f, h) := \sup_{|x-y|\leq h} |f(x) - f(y)|$ be its modulus of continuity. From standard approximation theory (c.f. [111]), we know that

Lemma 5.1.1. *Suppose $f \in C([0,1])$, let $\triangle$ be a partition over $[0,1]$ with n knots. Then there exists a $L_f \in S_1^0(\triangle)$ such that*

$$\|f - L_f\|_\infty \leq \begin{cases} \omega(f, \frac{1}{n}), & \text{if} \quad f \in C([0,1]), \\ \frac{1}{2n}\|f'\|_\infty, & \text{if} \quad f \in C^1([0,1]), \\ \frac{1}{8n^2}\|f''\|_\infty, & \text{if} \quad f \in C^2([0,1]). \end{cases} \tag{5.5}$$

Remark 5.1.2. *Note that even if we can further increase the smoothness of function f, the approximation rate is not getting better. In order to achieve a better approximation rate for those f with higher order smoothness, one has to use a higher degree splines. Therefore, for linear spline approximation, $\mathcal{O}(\frac{1}{n^2})$ is the optimal approximation rate.*

For $f \in C([0,1]^d)$, we would like to apply Lemma 5.1.1 for approximating the K-outer function g, and hence approximating f via the representation formula (4.2). For this purpose, let us define the linear KB-splines of f as

$$KB(f)_n(x_1, \cdots, x_d) := \sum_{q=0}^{2d} L_g \left(\sum_{i=1}^{d} \lambda_i \phi_q(x_i) \right), \tag{5.6}$$

where L_g is chosen to be a linear spline interpolation of the K-outer function $g \in C([0,d])$ with uniform partition of nd knots, i.e., $|\triangle| = \frac{1}{n}$. Then by Theorem 4.1.2 and Lemma 5.1.1, we have

Theorem 5.1.3. *Suppose $f \in C([0,1]^d)$. Then*

$$\|f - KB(f)_n\|_\infty \leq \begin{cases} (2d+1)\omega(g, \frac{1}{n}), & \text{if} \quad g \in C([0,d]), \\ \frac{2d+1}{2n}\|g'\|_\infty, & \text{if} \quad g \in C^1([0,d]), \\ \frac{2d+1}{8n^2}\|g''\|_\infty, & \text{if} \quad g \in C^2([0,d]). \end{cases} \tag{5.7}$$

Proof. We only show the proof for the case $g \in C^2([0,d])$, and the other two cases are similar. For any $\mathbf{x} = (x_1, \cdots, x_d)$, we have

$$\begin{aligned}\left|f(\mathbf{x}) - KB(f)_n(\mathbf{x})\right| &= \left|\sum_{q=0}^{2d} g\left(\sum_{i=1}^{d}\lambda_i\phi_q(x_i)\right) - \sum_{q=0}^{2d} L_g\left(\sum_{i=1}^{d}\lambda_i\phi_q(x_i)\right)\right| \\ &\leq \sum_{q=0}^{2d}\left|g\left(\sum_{i=1}^{d}\lambda_i\phi_q(x_i)\right) - L_g\left(\sum_{i=1}^{d}\lambda_i\phi_q(x_i)\right)\right| \leq \frac{2d+1}{8n^2}\|g''\|_\infty.\end{aligned}$$

This completes the proof. □

Theorem 5.1.3 immediately shows linear KB-splines are dense in $C([0,1]^d)$. More importantly, the approximation rate of linear KB-splines is independent of dimension d while the approximation constant is linearly dependent on d. Therefore we conclude that the approximation of high dimensional continuous function f does not suffer from the curse of dimensionality for a subclass of $C([0,1]^d)$, i.e., those f whose K-outer function $g \in C^1([0,d])$ or $g \in C^2([0,d])$. Such a subclass is dense because $C^1([0,d])$ and $C^2([0,d])$ are dense in $C([0,d])$. In fact, there are enormous choices of such g. For example, we can choose g to be polynomial functions, trigonometric functions, exponential functions, etc,.

Furthermore, let us recall the linear KB-splines basis functions defined in (4.25). Let $\triangle_n = \{0 = t_1 < t_2 < \cdots < t_{dn} < d\}$ be a uniform partition of interval $[0,d]$, and $b_{n,i}(t), i = 1, \cdots, dn$ be the standard univariate linear B-splines, we define the linear KB-spline (basis) functions as

$$KB_{n,j}(x_1, \cdots, x_d) := \sum_{q=0}^{2d} b_{n,j}\left(\sum_{i=1}^{d}\lambda_i\phi_q(x_i)\right), j = 1, \cdots, dn.$$

In Chapter 4, we have seen that these $KB_{n,j}$ have several nice properties, e.g. the partition of unity, linear independence, and denseness in $C([0,1]^d)$. Therefore we can treat $KB_{n,j}$ as basis functions for approximating $f \in C([0,1]^d)$. However, since there are only $\mathcal{O}(dn)$ number of $KB_{n,j}$ basis functions, the amount of information each basis $KB_{n,j}$ carries is much more than the information carried by each of the $\mathcal{O}(n^d)$ basis in the usual tensor product approximation or polynomial approximation. The basis $KB_{n,j}$ can behave very wildly, for

example, they are not differentiable, and hence can not be directly used for approximating f. For $d = 2$ and $d = 3$, we apply a spline denoising technique as introduced in the previous chapter to smooth the KB-splines and get the corresponding LKB-splines. For dimension $d \geq 4$, we need to apply tensor product of such denoising technique as introduced in the next section.

5.2 Tensor Product Approximation and Denoising

Let us first recall the approximation based on tensor product of Bernstein polynomial, which is well-known in the literature. We review them in order to explain the computation of tensor product splines for denoising in the later subsection.

5.2.1 Tensor product approximation of Bernstein polynomial

Suppose $f \in C([0,1])$, we define the Bernstein operator of degree n on f as

$$B_n f(x) := \sum_{i=0}^{n} f\left(\frac{i}{n}\right) B_{n,i}(x) \tag{5.8}$$

where $B_{n,i} = \binom{n}{i} x^i (1-x)^{n-i}$ is the Bernstein basis polynomial. From standard approximation theory (c.f. [111]), we know

Lemma 5.2.1. *Suppose $f \in C^2([0,1])$. Then*

$$\|f - B_n f\|_\infty \leq \frac{1}{8n} \|f''\|_\infty. \tag{5.9}$$

In general, for $f \in C([0,1]^d)$, we can define

$$B_{n_1,\cdots,n_d} f(x_1, \cdots, x_d) := \sum_{i_1=0}^{n_1} \cdots \sum_{i_d=0}^{n_d} f\left(\frac{i_1}{n_1}, \cdots, \frac{i_d}{n_d}\right) B_{n_1,i_1}(x_1) \cdots B_{n_d,i_d}(x_d). \tag{5.10}$$

By applying Lemma 5.2.1 and and a chain of triangle inequalities argument, it is not hard to establish the following result. We leave its proof to the interested readers.

Lemma 5.2.2. *Suppose $f \in C^2([0,1]^d)$ for integer $d \geq 1$. Then*

$$\|f - B_{n,\cdots,n} f\|_\infty \leq \frac{d}{8n} |f''|_{2,\infty}, \tag{5.11}$$

where $|f|_{2,\infty} = \max_{i_1+\cdots+i_d=2} \|D_{x_1}^{i_1} \cdots D_{x_d}^{i_d} f\|_\infty$.

5.2.2 Tensor product of spline denoising

As mentioned before, the linear KB-splines obtained via (4.25) can behave wildly, therefore may not be directly useful for approximation. We would like to smooth/denoise them so that they will be useful. For self-containedness, let us introduce the ideas of spline denoising and tensor product spline denoising. For convenience, we base our discussion on the bivariate splines. Let us first recall bivariate spline spaces. For a triangulation $\triangle$ of $[0,1]^2$, for any degree $d \geq 1$ and smoothness $r \geq 1$ with $r < d$, let

$$S_d^r(\triangle) = \{s \in C^r([0,1]^2) : s|_T \in \mathbb{P}_d, T \in \triangle\} \tag{5.12}$$

be the spline space of degree d and smoothness r with $d > r$. We refer to [73] for a theoretical detail and [6] and [72] for a computational detail of multivariate splines. For convenience, we can write a spline function

$$s(x,y) = \sum_{i=1}^{m} c_i b_i(x,y) \in S_d^r(\triangle), \tag{5.13}$$

where c_i's are the spline coefficients, $b_i(\cdot,\cdot)$ are bivariate basis splines with degree d and smoothness r, and m is the dimension of the bivariate spline space. Note that the computation of $b_i(x,y)$ is not easy at all. We adopt the approach in [6].

For a given data set $\{(x_i, y_i, z_i), i = 1, \cdots, N\}$ with data locations $(x_i, y_i) \in [0,1]^2$ and noisy data values z_i, the penalized least squares method (cf. [71] and [74]) of bivariate spline denoising is to find

$$\min_{s \in S_d^r(\triangle)} \sum_{i=1,\cdots,N} \left| s(x_i, y_i) - z_i \right|^2 + \lambda \mathcal{E}_2(s) \tag{5.14}$$

for some fixed constant $\lambda \approx 1$, where $\mathcal{E}_2(s)$ is the thin-plate energy functional defined as

$$\mathcal{E}_2(s) := \int_{[0,1]^2} \left| \frac{\partial^2}{\partial x^2} s \right|^2 + 2 \left| \frac{\partial^2}{\partial x \partial y} s \right|^2 + \left| \frac{\partial^2}{\partial y^2} s \right|^2. \tag{5.15}$$

It is well-known that this approach can be used for smoothing noisy data. This is the denoising technique we have used in Chapter 4 to generate linear LKB-splines.

Now let us explain the idea of tensor product spline based denoising method for smoothing noisy KB-splines. For convenience, let us consider the case $d = 4$, the similar arguements can be applied to a general d. See [23] for the general case for tensor product splines for data interpolation. For the rest of the discussion, we shall consider the tensor product the bivariate spline space $\mathcal{S} := S_d^r(\triangle) \times S_d^r(\triangle)$.

For a given data set $\{(x_i, y_i, u_j, v_j, z_{i,j}), i, j = 1, \cdots, N\}$ with data locations $(x_i, y_i, u_j, v_j) \in [0,1]^2 \times [0,1]^2$ and noisy data values $z_{i,j}$, we can write a spline function

$$s(x, y, u, v) = \sum_{i=1}^{m_1} \sum_{j=1}^{m_2} c_{ij} b_i(x, y) b_j(u, v) \in \mathcal{S}, \tag{5.16}$$

where c_{ij}'s are the spline coefficients, $b_i(\cdot, \cdot)$ are bivariate splines with degree d and smoothness r, and m_1, m_2 are the dimensions of the bivariate spline spaces. The penalized least squares method of tensor product bivariate spline denoising is to find the spline coefficients c_{ij} which solves

$$\min_{s \in \mathcal{S}} \sum_{i,j=1,\cdots,N} |s(x_i, y_i, u_j, v_j) - z_{ij}|^2 + \lambda \mathcal{E}_{2\times 2}(s) \tag{5.17}$$

with $r > 0, d > 0, \lambda \approx 1$ are hyperparameters, and $\mathcal{E}_{2\times 2}(s)$ is defined as

$$\begin{aligned} \mathcal{E}_{2\times 2}(s) := & \int_{[0,1]^2} \left(\int_{[0,1]^2} \left| \frac{\partial^2}{\partial x^2} s \right|^2 + 2 \left| \frac{\partial^2}{\partial x \partial y} s \right|^2 + \left| \frac{\partial^2}{\partial y^2} s \right|^2 dxdy \right) dudv \\ & + \int_{[0,1]^2} \left(\int_{[0,1]^2} \left| \frac{\partial^2}{\partial u^2} s \right|^2 + 2 \left| \frac{\partial^2}{\partial u \partial v} s \right|^2 + \left| \frac{\partial^2}{\partial v^2} s \right|^2 dudv \right) dxdy. \end{aligned} \tag{5.18}$$

Let us explain next the computational procedure for finding the spline coefficients c_{ij} based on a two-stage bivariate spline denoising scheme. Recall that tensor product splines for data interpolation were explained in [23]. We extend its ideas to data denoising. For a given data set $\{(x_i, y_i, u_j, v_j, z_{i,j}), i, j = 1, \cdots, N\}$ with data locations $(x_i, y_i) \in [0,1]^2$ and $(u_j, v_j) \in [0,1]^2$ and noised data values $z_{i,j}$, $i, j = 1, \cdots, N$, we can write

$$s(x, y, u, v) = \sum_{i=1}^{m_1} \sum_{j=1}^{m_2} c_{ij} b_j(u, v) b_i(x, y). \tag{5.19}$$

Suppose our data is equally-spaced over $[0,1]^2 \times [0,1]^2$, i.e., $N = m_1 = m_2$. Let us denote $d_i(u,v) = \sum_{j=1}^{m_2} c_{ij} b_j(u,v)$, then we can write equation (5.19) as $s(x,y,u,v) = \sum_{i=1}^{m_1} d_i(u,v) b_i(x,y)$. For fixed (u_k, v_k), $k = 1, \cdots, N$, write $d_i(u_k, v_k) = d_{ik}$ for all $i = 1, \cdots, m_1$. For each fixed k, we can find the intermediate spline coefficients d_{ik} via (5.14) by letting

$$s(x_\ell, y_\ell)_k := s(x_\ell, y_\ell, u_k, v_k) = \sum_{i=1}^{m_1} d_{ik} b_i(x_\ell, y_\ell) \tag{5.20}$$

for $\ell = 1, \cdots, N$. Once we have d_{ik}, then for each fixed i, we can find the spline coefficients c_{ij} via (5.14) by letting

$$s(u_k, v_k) := d_{ik} = \sum_{j=1}^{m_2} c_{ij} b_j(u_k, v_k) \tag{5.21}$$

for $k = 1, \cdots, N$.

The advantage of tensor product spline denoising is its computational efficiency. If we directly solve the penalized least squares problem (5.17) for the coefficients c_{ij} without using this tensor product approach, then the matrix size associated in (5.19) is $N^2 \times m_1 m_2$. Hence, solving it directly requires the computation complexity $\mathcal{O}(m_1^2 m_2^2 N^2)$. However, if we solve it by using tensor product via (5.20) and (5.21), then we only need to solve N systems whose matrix size is of $N \times m_1$ and another N systems whose matrix size is of $N \times m_2$. Therefore the computational complexity for solving them directly requires $\mathcal{O}(N m_1^2 N + N m_2^2 N) = \mathcal{O}((m_1^2 + m_2^2) N^2)$. If we use large degree d and high smoothness $r \geq 1$ for denoising, then $m_1^2 + m_2^2 \ll m_1^2 m_2^2$. Therefore, the computational cost for the two-stage tensor product denoising technique is much less than the direct denosing technique. This is why we adopt the tensor product spline denosing method. For the general case $d > 4$, we can easily extend this idea to have a multi-stage denoising scheme. We leave out the details here.

For each high dimensional linear KB-spline obtained via (4.25), we can apply such a computational scheme to solve (5.17) and obtain the corresponding high dimensional linear LKB-spline, which is useful for approximation. We will use these linear LKB-splines as basis for high dimensional function approximation. Numerical results will be discussed in the following sections.

5.3 Numerical Results for LKB-splines based Approximation in 4D and 6D

Let us present the numerical results for function approximation in $\mathbb{R}^d$ with $d = 4$ and $d = 6$ based on the linear LKB-splines obtained via the computation procedure described in the previous section.

In 4D, we sampled 11^4 equally-spaced data across $[0, 1]^4$ and fit discrete least squares (DLS) approximation of a continuous function f with 4D linear LKB-spline as basis, and we check the RMSEs based on 26^4 equally-spaced data across $[0, 1]^4$. The following 10 testing functions across different families of continuous functions are used to check the approximation accuracy.

$$
\begin{aligned}
f_1 &= (1 + 2x + 3y + 4u + 5v)/15; \\
f_2 &= (x^2 + y^2 + u^2 + v^2)/4; \\
f_3 &= (x^4 + y^4 + u^4 + v^4)/4; \\
f_4 &= (\sin(x)\exp(y) + \cos(x)\exp(u) + \sin(x)\exp(v))/(3\exp(1)); \\
f_5 &= 1/(1 + x^2 + y^2 + u^2 + v^2); \\
f_6 &= \sin(\pi x)\sin(\pi y)\sin(\pi u)\sin(\pi v); \\
f_7 &= (\sin(\pi(x^2 + y^2 + u^2 + v^2)) + 1)/2; \\
f_8 &= \exp(-x^2 - y^2 - u^2 - v^2); \\
f_9 &= \max(x - 0.5)\max(y - 0.5)\max(u - 0.5)\max(v - 0.5); \\
f_{10} &= \max(x + y + u + v - 2, 0);
\end{aligned}
$$

In 6D, we sampled 6^6 equally-spaced data across $[0, 1]^6$ and fit DLS approximation of a continuous function f with 6D linear LKB-splines, and we check the RMSEs based on 11^6 equally-spaced data across $[0, 1]^6$. The following 10 testing functions across different families of continuous functions are used to check the approximation accuracy.

$$
f_1 = (1 + 2x + 3y + 4z + 5u + 6v + 7w)/28;
$$

$$
\begin{aligned}
f_2 &= (x^2+y^2+z^2+u^2+v^2+w^2)/6;\\
f_3 &= (x^3y^3+x^3z^3+y^3z^3+x^3u^3+u^3v^3+v^3w^3)/6;\\
f_4 &= (\sin(x)e^y+\cos(x)e^z+\sin(x)e^u+\cos(y)e^v+\sin(x)e^w)/(5e);\\
f_5 &= 1/(1+x^2+y^2+z^2+u^2+v^2+w^2);\\
f_6 &= \sin(\pi x)\sin(\pi y)\sin(\pi z)\sin(\pi u)\sin(\pi v)\sin(\pi w);\\
f_7 &= (\sin(\pi(x^2+y^2+z^2+u^2+v^2+w^2))+1)/2;\\
f_8 &= \exp(-x^2-y^2-z^2-u^2-v^2-w^2);\\
f_9 &= \max(x-0.5)\max(y-0.5)\max(z-0.5)\max(u-0.5)\max(v-0.5)\max(w-0.5);\\
f_{10} &= \max(x+y+z+u+v+w-3,0);
\end{aligned}
$$

Table 5.1: RMSEs (computed based on 26^4 equally-spaced locations) of the DLS fitting based 11^4 equally-space sampled data and pivotal location in 4D.

	$n=100$		$n=300$		$n=1000$	
# Sampled Data	11^4	128	11^4	241	11^4	531
f_1	3.06e-04	8.90e-04	6.02e-05	4.24e-04	2.79e-06	9.86e-06
f_2	9.70e-04	2.75e-03	4.35e-04	1.63e-03	2.66e-04	5.85e-04
f_3	4.00e-03	1.13e-02	1.87e-03	6.88e-03	1.12e-03	2.32e-03
f_4	5.86e-04	1.88e-03	3.23e-04	1.45e-03	1.62e-04	4.31e-04
f_5	1.39e-03	3.63e-03	4.76e-04	1.80e-03	2.67e-04	7.07e-04
f_6	3.40e-02	1.07e-01	1.33e-02	7.80e-02	3.96e-03	2.24e-02
f_7	9.75e-02	3.07e-01	4.13e-02	1.96e-01	1.57e-02	5.40e-02
f_8	1.54e-03	3.78e-03	6.28e-04	2.55e-03	3.58e-04	8.85e-04
f_9	3.51e-04	1.29e-03	1.80e-04	1.56e-03	1.03e-04	5.59e-04
f_{10}	2.53e-02	5.32e-02	1.96e-02	8.25e-02	1.40e-02	3.45e-02

Table 5.2: RMSEs (computed based on 11^6 equally-spaced locations) of the DLS fitting based 6^6 equally-space sampled data and pivotal location in 6D.

	$n = 20$		$n = 40$		$n = 120$	
# Sampled Data	6^6	13	6^6	24	6^6	70
f_1	5.09e-02	7.81e-02	3.03e-02	5.52e-02	7.78e-03	3.61e-02
f_2	4.56e-02	1.31e-01	4.09e-02	8.30e-02	1.73e-02	5.31e-02
f_3	9.70e-02	1.29e-01	5.26e-02	8.39e-02	2.85e-02	5.09e-02
f_4	7.50e-02	2.50e-01	7.27e-02	1.50e-01	3.59e-02	1.23e-01
f_5	5.44e-02	1.88e-01	4.98e-02	1.22e-01	2.72e-02	7.25e-02
f_6	3.95e-02	8.07e-02	3.55e-02	6.14e-02	1.64e-02	4.45e-02
f_7	2.50e-02	8.71e-02	2.40e-02	6.46e-02	9.08e-03	3.94e-02
f_8	8.84e-02	9.39e-02	6.83e-02	7.39e-02	4.62e-02	5.40e-02
f_9	3.47e-01	3.55e-01	2.30e-01	2.56e-01	1.04e-01	1.86e-01
f_{10}	3.12e-02	7.66e-02	2.37e-02	5.65e-02	1.25e-02	3.94e-02

In addition, we noticed that the linear system associated with the DLS approximation has many zero or near zero columns due to the structure K-inner functions. As discussed in Chapter 4, we adopt the matrix cross approximation in [4] to find the pivotal point set. Based on the function values at the pivotal points in $[0, 1]^d$, we can simply solve the subsystem with much smaller size to find the approximation of f. Similar RMSEs are computed and present in Table 5.1 and 5.2 side by side to show that the approximation of f for both approaches works well and the approximation based on the function values over the pivotal points is similar to the approximation based on the 11^4 function values.

We plot the approximation error based on the pivotal location against n on the log-log scale, hence the exponent of n in the approximation rate is associated with the slope of the fitted line via linear regression. The results are shown in Figure 5.3. The fitted line with

slope less than -1 or -2 indicates that the approximation rate is at least $\mathcal{O}(1/n)$ or $\mathcal{O}(1/n^2)$, which indicates the corresponding $g_f \in C^1([0,d])$ or $g_f \in C^2([0,d])$.

We also plot the number of pivotal locations needed to achieve those approximation errors, the results are shown in Figure 5.2. It shows that we only need fewer than $\mathcal{O}(nd)$ function values of f to achieve the convergence rate $\mathcal{O}(1/n^\beta)$, $\mathcal{O}(1/n)$, or $\mathcal{O}(1/n^2)$ based on the smoothness of K-outer function g_f.

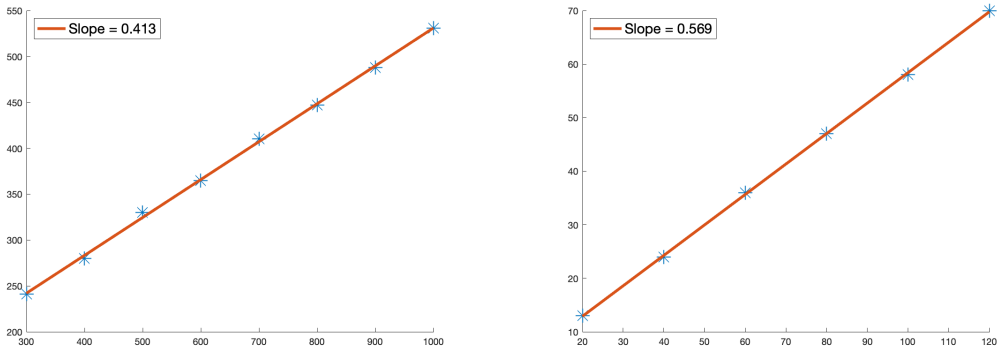
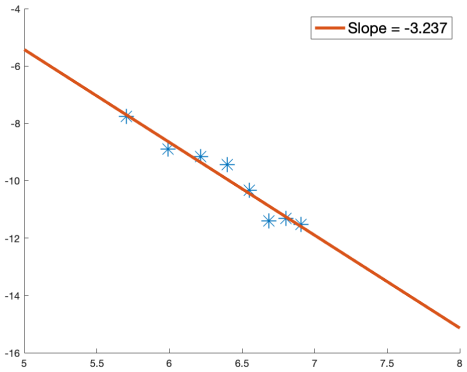
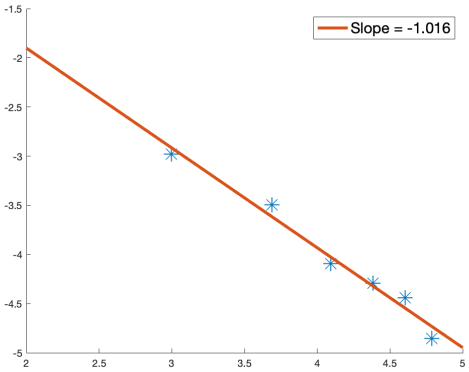


Figure 5.2: Number of pivotal locations (vertical axis) against n (horizontal axis) in 4D (left) and in 6D (right).

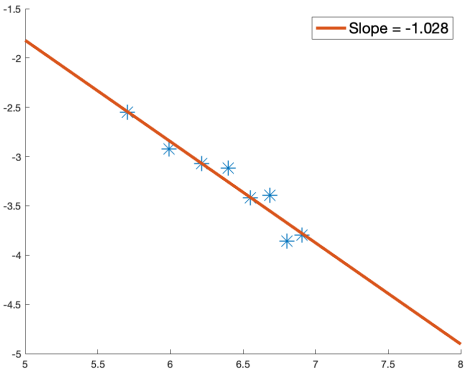
$f = (1 + 2x + 3y + 4u + 5v)/15$



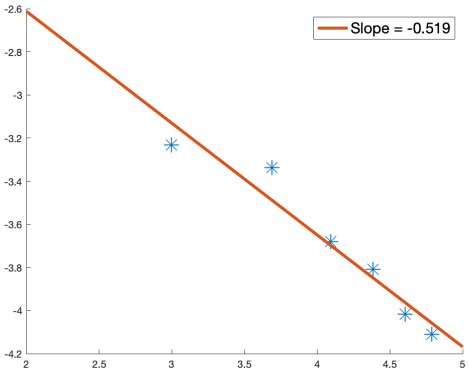
$f = (1 + 2x + 3y + 4z + 5u + 6v + 7w)/28$



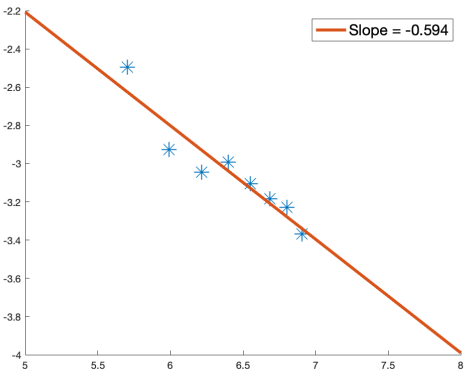
$f = \sin(\pi x)\sin(\pi y)\sin(\pi u)\sin(\pi v)$



$f = \sin(\pi x)\sin(\pi y)\sin(\pi z)\sin(\pi u)\sin(\pi v)\sin(\pi w)$



$f = \max(x + y + u + v - 2, 0)$



$f = \max(x + y + z + u + v + w - 3, 0)$

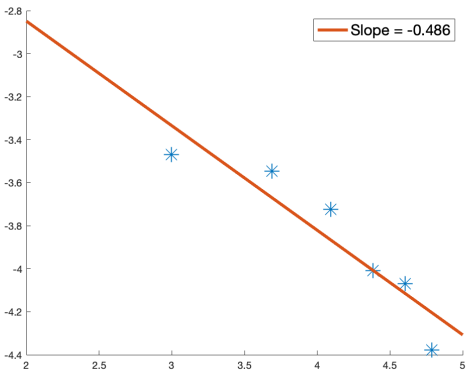


Figure 5.3: Plots of convergence rate on the Log-log scale in 4D and 6D based on pivotal dataset.

5.4 Application to Numerical Solution of Poisson Equation

One of the powerful aspects of the LKB-splines based approximation scheme is that we can use it to solve the partial differential equations. To start with, let us consider the Poisson equation:

$$\begin{cases} \Delta u &= f, \mathbf{x} \in \Omega \\ u &= 0, \mathbf{x} \in \partial\Omega. \end{cases} \tag{5.22}$$

For simplicity, let us consider the 2D case where $\Omega = [0,1]^2$. Let us use the $f_i = LKB_i, i = 1, \cdots, 2n$ as the right-hand side of (5.22). We can use bivariate spline function of degree $d = 8$ and $r = 2$ to solve (5.22) by using the spline collocation method as proposed in [72] to obtain the solution u_i $i = 1, \cdots, 2n$. These form a set of basic Poisson solutions. Then for any f, we use $LKB_i, i = 1, \cdots, 2n$ to approximate f. As discussed in the previous section, we can use a small number of LKB_i to approximate f very well. Letting $f = \sum_{i=1}^{2n} c_i(f) LKB_i$ be a good approximation of f, the solution u of the Poisson equation (5.22) can be well approximated by

$$u_n = \sum_{i=1}^{2n} c_i(f) u_i. \tag{5.23}$$

To show $u - u_n$ goes to 0 when $n \to \infty$, we consider $\|\Delta(u - u_n)\|_{L^2(\Omega)}$ which is $\|f - \sum_{i=1}^{n} c_i(f) LKB_i\|_{L^2(\Omega)}$. Let us recall a basic result from [72]. Define a new norm $\|u\|_L$ on $H^2(\Omega) \cap H_0^1(\Omega)$ for the Poisson equation as follows.

$$\|u\|_L = \|\Delta u\|_{L^2(\Omega)}. \tag{5.24}$$

Lai and Lee in [72] showed that the new norm is equivalent to the standard norm on Banach space $H^2(\Omega) \cap H_0^1(\Omega)$. That is,

Theorem 5.4.1. *Suppose $\Omega \subset \mathbb{R}^d$ be a bounded domain and the closure of Ω is of uniformly positive reach $r_\Omega > 0$. Then there exist two positive constants A and B such that*

$$A\|u\|_{H^2(\Omega)} \leq \|u\|_L \leq B\|u\|_{H^2}, \quad \forall u \in H^2(\Omega) \cap H_0^1(\Omega), \tag{5.25}$$

where $\|\cdot\|_{H^2(\Omega)}$ is the standard H^2 norm for the Sobolev space $H^2(\Omega)$.

For convenience, let $\epsilon_1 = \|f - \sum_{i=1}^{n} c_i(f) LKB_i\|_{L^2(\Omega)}$, where $\Omega = [0,1]^2$ which is convex and hence, has an uniformly positive reach. We refer the interested reader to [72] for the definition of positive reach.

The results in Theorem 5.4.1 show that

$$\|u - u_n\|_{H^2(\Omega)} \leq \epsilon_1 / A.$$

As in the previous section, $\epsilon_1 \to 0$ if f is K-Lipschitz continuous, we can further show the following results by using the arguments in [72]:

Theorem 5.4.2. *Suppose $\Omega \subset \mathbb{R}^d$ is a bounded domain and the closure of Ω is of uniformly positive reach $r_\Omega > 0$. Suppose that $u \in H^3(\Omega)$. Then we have the following inequalities:*

$$\|u - u_n\|_{L^2(\Omega)} \leq C|\triangle|^2 \epsilon_1 \text{ and } \|\nabla(u - u_n)\|_{L^2(\Omega)} \leq C|\triangle|\epsilon_1$$

for a positive constant $C = 1/A$, where A is one of the constants in Theorem 5.4.1 and $|\triangle|$ is the size of the underlying triangulation $\triangle$.

5.4.1 Numerical results

For numerical experiments, we will use the following six functions as testing functions for the solution of (5.22). Their right-hand-side f can be easily computed.

$$\begin{aligned}
u_1 &= x(1-x)y(1-y)/4; \\
u_2 &= \sin(\pi x)\sin(\pi y); \\
u_3 &= \sin(x)(1-x)(1-y)\sin(y); \\
u_4 &= (x(1-x)y(1-y)/4)^2; \\
u_5 &= (\sin(\pi x)\sin(\pi y))^2; \\
u_6 &= (\sin(x)(1-x)(1-y)\sin(y))^2;
\end{aligned}$$

We first use the linear LKB-splines to approximate the right-hand-side f associated with $u_1, \cdots, u_6$ over $[0,1]^2$. We sampled 101^2 equally-spaced data across $[0,1]^2$ and fit the discrete

least squares (DLS) approximation of a continuous function f with LKB-splines as basis, and we check the RMSEs based on 1001^2 equally-spaced data across $[0,1]^2$. See Table 5.3 for the numerical results.

Table 5.3: RMSEs (computed based on 1001^2 equally-spaced locations) of the approximation for the right-hand-side function $f = \Delta u$ based on equally-space sampled data and pivotal locations.

	$n = 100$		$n = 300$		$n = 1000$	
# Sampled Data	101^2	59	101^2	76	101^2	136
Δu_1	4.90e-04	9.67e-04	2.46e-04	5.39e-04	1.01e-04	2.91e-04
Δu_2	3.04e-02	4.35e-02	1.31e-02	2.22e-02	3.90e-03	6.98e-03
Δu_3	2.00e-03	3.80e-03	1.00e-03	2.30e-03	3.77e-04	1.10e-03
Δu_4	9.05e-05	1.32e-04	3.85e-05	8.59e-05	6.98e-06	1.86e-05
Δu_5	2.38e-01	4.26e-01	1.13e-01	1.53e-01	2.83e-02	6.06e-02
Δu_6	1.50e-03	2.20e-03	4.90e-04	9.49e-04	1.20e-04	3.17e-04

Next we compute the spline solution of the Poisson equation for each LKB-spline as the right-hand side of the PDE (5.22) to obtain u_i's. As explained above, we use the coefficients of linear LKB-spline approximation of each right-hand-side function f to form the solution of the Poisson equation. We apply the method described in (5.23) to approximate the solution of the Poisson equation and the numerical results are shown in Table 5.4. We can see that our method works very well.

So far we only explained how to use LKB-splines for approximating the solution of the Poisson equation with zero boundary conditions. The underlying domain of interest is $[0,1]^2$. This is the most simple PDE. We are currently studying how to use the LKB-splines for the numerical solution of the Poisson equation over arbitrary polygons with nonzero Dirichlet boundary conditions.

Table 5.4: RMSEs (computed based on 1001^2 equally-spaced locations) of the approximation for the true solution u based on equally-space sampled data and pivotal locations.

	$n = 100$		$n = 300$		$n = 1000$	
# Sampled Data	101^2	59	101^2	76	101^2	136
u_1	1.04e-06	1.57e-05	4.02e-07	5.97e-06	8.09e-08	1.92e-06
u_2	1.82e-04	1.20e-03	5.09e-05	6.95e-04	9.53e-06	1.05e-04
u_3	4.56e-06	6.30e-05	1.82e-06	2.38e-05	3.23e-07	6.47e-06
u_4	2.11e-07	6.91e-07	6.71e-08	1.07e-06	8.34e-09	1.05e-07
u_5	1.80e-03	8.40e-03	5.98e-04	1.90e-03	1.26e-04	1.01e-03
u_6	4.30e-06	1.15e-05	9.51e-07	9.51e-06	1.50e-07	1.77e-06

The advantage of this approach is that the basic solutions u_i's of the Poisson equation can be solved beforehand and stored and one only needs to approximate the right-hand-side function. Note that the right-hand-side function f can be easily approximated based on the pivotal point locations without using a large amount of the function values if f is K-Hölder continuous. This approach provides an efficient method for solving PDE numerically. As we have seen, when f is K-Lipschitz or K-Hölider continuous, the LKB-splines approach gives a nice approximation of the right-hand side, hence the accurate approximation of the solution to the original Poisson equation.

So far we have seen that KB-splines can approximate a general continuous function (polynomial, exponential, rational, etc,.) very well without many function evaluations. However, the current scheme with KB-splines does not work very well for approximating trigonometric functions with high frequency, e.g., $f(x, y) = \sin(100x + 100y)$. We can instead using Fourier basis to replace $b_{n,j}$ and define

$$KF_{n,j}(x_1, \cdots, x_d) := \sum_{q=0}^{2d} \Big(p_j(\sum_{i=1}^{d} \lambda_i \phi_q(x_i)) + q_j(\sum_{i=1}^{d} \lambda_i \phi_q(x_i)) \Big), j = 1, \cdots, n, \tag{5.26}$$

where $p_j(t) = \cos(2\pi jt)$ and $q_j(t) = \sin(2\pi jt)$. We can also increase n to achieve a better approximation accuracy. In this way, it may have the capacity to approximate high frequency trigonometric functions or some other highly oscillated functions effectively.

Some other potential research directions are how to apply the KST representation for approximating discontinuous functions, e.g., images, and how to extend the current approximation scheme for general domains in $\mathbb{R}^n$. We leave these investigation to future work.

Bibliography

[1] Emmanuel Abbe. Community detection and stochastic block models: Recent developments. *Journal of Machine Learning Research*, 18(177):1–86, 2018.

[2] Emmanuel Abbe and Clément Sandon. Recovering communities in the general stochastic block model without knowing the parameters. In *Advances in Neural Information Processing Systems*, pages 676–684, 2015.

[3] Lada A. Adamic and Natalie Glance. The political blogosphere and the 2004 us election: Divided they blog. In *Proceedings of the 3rd International Workshop on Link Discovery*, pages 36–43, 2005.

[4] Kenneth Allen, Ming-Jun Lai, and Zhaiming Shen. Maximal volume matrix cross approximation for image compression and least squares solution. *arXiv preprint arXiv:2309.17403*, 2023.

[5] Reid Andersen, Fan Chung, and Kevin Lang. Local graph partitioning using pagerank vectors. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 475–486. IEEE, 2006.

[6] Gerard Awanou, Ming-Jun Lai, and Paul Wenston. The multivariate spline method for scattered data fitting and numerical solutions of partial differential equations. In *Wavelets and Splines: Athens*, pages 24–74. 2005.

[7] Francis Bach. Breaking the curse of dimensionality with convex neural networks. *The Journal of Machine Learning Research*, 18(1):629–681, 2017.

[8] Andrew R. Barron. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information Theory*, 39(3):930–945, 1993.

[9] Sugato Basu, Arindam Banerjee, and Raymond J. Mooney. Active semi-supervision for pairwise constrained clustering. In *Proceedings of the 2004 SIAM International Conference on Data Mining*, pages 333–344. Society for Industrial and Applied Mathematics, 2004.

[10] Thomas Blumensath and Mike E Davies. Iterative hard thresholding for compressed sensing. *Applied and computational harmonic analysis*, 27(3):265–274, 2009.

[11] Jürgen Braun. *An Application of Kolmogorov's Superposition Theorem to Function Reconstruction in Higher Dimensions*. PhD thesis, University of Bonn, 2009.

[12] Jürgen Braun and Michael Griebel. On a constructive proof of kolmogorov's superposition theorem. *Constructive Approximation*, 30(3):653–675, 2009.

[13] D. W. Bryant. *Analysis of Kolmogorov's Superposition Theorem and Its Implementation in Applications with Low and High Dimensional Data*. PhD thesis, University of Central Florida, 2008.

[14] T. Tony Cai and Anru Zhang. Sharp rip bound for sparse signal and low-rank matrix recovery. *Applied and Computational Harmonic Analysis*, 35(1):74–93, 2013.

[15] Emmanuel J Candès and Terence Tao. Decoding by linear programming. *IEEE transactions on information theory*, 51(12):4203–4215, 2005.

[16] Emmanuel J. Candès, Justin Romberg, and Terence Tao. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on Information Theory*, 52(2):489–509, 2006.

[17] Scott Shaobing Chen, David L. Donoho, and Michael A. Saunders. Atomic decomposition by basis pursuit. *SIAM Review*, 43(1):129–159, 2001.

[18] Fan Chung. The heat kernel as the pagerank of a graph. *Proceedings of the National Academy of Sciences*, 104(50):19735–19740, 2007.

[19] Fan Chung and Linyuan Lu. *Complex Graphs and Networks*, volume 107. American Mathematical Society, 2006.

[20] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2(4):303–314, 1989.

[21] Wei Dai and Olgica Milenkovic. Subspace pursuit for compressive sensing signal reconstruction. *IEEE Transactions on Information Theory*, 55(5):2230–2249, May 2009.

[22] Ingrid Daubechies, Ronald DeVore, Simon Foucart, Boris Hanin, and Guergana Petrova. Nonlinear approximation and (deep) relu networks. *Constructive Approximation*, 55(1):127–172, 2022.

[23] Carl De Boor. Efficient computer manipulation of tensor products. *ACM Transactions on Mathematical Software (TOMS)*, 5(2):173–182, 1979.

[24] Stephen Demko. A superposition theorem for bounded continuous functions. *Proceedings of the American Mathematical Society*, 66(1):75–78, 1977.

[25] Ronald DeVore, Boris Hanin, and Guergana Petrova. Neural network approximation. *Acta Numerica*, 30:327–444, 2021.

[26] Ronald A. DeVore, Ralph Howard, and Charles Micchelli. Optimal nonlinear approximation. *Manuscripta Mathematica*, 63:469–478, 1989.

[27] Inderjit S. Dhillon, Yuqiang Guan, and Brian Kulis. Kernel k-means: Spectral clustering and normalized cuts. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2004.

[28] Chris HQ Ding, Xiaofeng He, Hongyuan Zha, Ming Gu, and Horst D. Simon. A min-max cut algorithm for graph partitioning and data clustering. In *Proceedings of the 2001 IEEE International Conference on Data Mining*, pages 107–114, 2001.

[29] David L. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4):1289–1306, 2006.

[30] Raouf Doss. A superposition theorem for unbounded continuous functions. *Transactions of the American Mathematical Society*, 233:197–203, 1977.

[31] Weinan E. Machine learning and computational mathematics. *arXiv preprint*, 2020.

[32] Weinan E, Chao Ma, and Lei Wu. The barron space and the flow-induced function spaces for neural network models. *Constructive Approximation*, 55(1):369–406, 2022.

[33] Weinan E and Stephan Wojtowytsch. On the banach spaces associated with multi-layer relu networks: Function representation, approximation theory, and gradient descent dynamics. *arXiv preprint*, 2020.

[34] Weinan E and Stephan Wojtowytsch. Representation formulas and pointwise properties for barron functions. *Calculus of Variations and Partial Differential Equations*, 61(2):1–37, 2022.

[35] P. Erdős and A. Rényi. On random graphs i. *Publ. Math. Debrecen*, 6:290–297, 1959.

[36] Daniele Fakhoury, Emanuele Fakhoury, and Hendrik Speleers. Exsplinet: An interpretable and expressive spline-based neural network. *Neural Networks*, 152:332–346, 2022.

[37] Renzhong Feng, Aitong Huang, Ming-Jun Lai, and Zhaiming Shen. Reconstruction of sparse polynomials via quasi-orthogonal matching pursuit method. *J. Comput. Math*, 2021.

[38] Zhiyuan Feng. *Hilbert's 13th Problem*. PhD thesis, University of Pittsburgh, 2010.

[39] Miroslav Fiedler. Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, 23(2):298–305, 1973.

[40] Simon Foucart. A note on guaranteed sparse recovery via ℓ_1-minimization. *Applied and Computational Harmonic Analysis*, 29(1):97–103, 2010.

[41] Simon Foucart and Ming-Jun Lai. Sparsest solutions of underdetermined linear systems via ℓ_q-minimization for $0 < q \leq 1$. *Applied and Computational Harmonic Analysis*, 26(3):395–407, 2009.

[42] Kimon Fountoulakis, David F. Gleich, and Michael W. Mahoney. A short introduction to local graph clustering methods and software. *arXiv preprint arXiv:1810.07324*, 2018.

[43] Kimon Fountoulakis, Farbod Roosta-Khorasani, Julian Shun, Xiang Cheng, and Michael W. Mahoney. Variational perspective on local graph clustering. *Mathematical Programming*, 174:553–573, 2019.

[44] B. Fridman. Improvement in the smoothness of functions in the kolmogorov superposition theorem. *Doklady Akademii Nauk SSSR*, 177(5), 1967.

[45] Irina Georgieva and Clemens Hofreither. On best uniform approximation by low-rank matrices. *Linear Algebra and its Applications*, 518:159–176, 2017.

[46] Federico Girosi and Tomaso Poggio. Representation properties of networks: Kolmogorov's theorem is irrelevant. *Neural Computation*, 1(4):465–469, 1989.

[47] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[48] Sergei A. Goreinov, Ivan V. Oseledets, Dimitry V. Savostyanov, Eugene E. Tyrtyshnikov, and Nikolay L. Zamarashkin. How to find a good submatrix. In *Matrix Methods:*

Theory, Algorithms And Applications: Dedicated to the Memory of Gene Golub, pages 247–256, 2010.

[49] Sergei A. Goreinov and Eugene E. Tyrtyshnikov. The maximal-volume concept in approximation by low-rank matrices. *Contemporary Mathematics*, 280:47–52, 2001.

[50] Xifeng Guo, Long Gao, Xinwang Liu, and Jianping Yin. Improved deep embedded clustering with local structure preservation. In *IJCAI*, volume 17, pages 1753–1759, 2017.

[51] Kun He, Yiwei Sun, David Bindel, John Hopcroft, and Yixuan Li. Detecting overlapping communities from local spectral subspaces. In *2015 IEEE international conference on data mining*, pages 769–774. IEEE, 2015.

[52] Robert Hecht-Nielsen. Kolmogorov's mapping neural network existence theorem. In *Proceedings of the International Conference on Neural Networks*, volume 3, pages 11–14, New York, NY, USA, 1987. IEEE Press.

[53] Gennadi Markovich Henkin. Linear superpositions of continuously differentiable functions. *Doklady Akademii Nauk*, 157(2):288–290, 1964.

[54] Matthew A. Herman and Thomas Strohmer. General deviants: An analysis of perturbations in compressed sensing. *IEEE Journal of Selected topics in signal processing*, 4(2):342–349, 2010.

[55] Paul W Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. Stochastic blockmodels: First steps. *Social Networks*, 5(2):109–137, 1983.

[56] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257, 1991.

[57] Hsin-Chien Huang, Yung-Yu Chuang, and Chu-Song Chen. Affinity aggregation for spectral clustering. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[58] Jin Huang, Feiping Nie, and Heng Huang. A new simplex sparse learning model to measure data similarity for clustering. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[59] Binyuan Hui, Pengfei Zhu, and Qinghua Hu. Collaborative graph convolutional networks: Unsupervised learning meets semi-supervised learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 4215–4222, 2020.

[60] Boris Igelnik and Neel Parikh. Kolmogorov's spline network. *IEEE Transactions on Neural Networks*, 14(4):725–733, 2003.

[61] Matt Jacobs, Ekaterina Merkurjev, and Selim Esedoḡlu. Auction dynamics: A volume constrained mbo scheme. *Journal of Computational Physics*, 354:288–310, 2018.

[62] Zhao Kang, Chong Peng, Qiang Cheng, Xinwang Liu, Xi Peng, Zenglin Xu, and Ling Tian. Structured graph learning for clustering and semi-supervised classification. *Pattern Recognition*, 110:107627, 2021.

[63] Kyle Kloster and David F Gleich. Heat kernel based community detection. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1386–1395, 2014.

[64] Jason M. Klusowski and Andrew R. Barron. Approximation by combinations of relu and squared relu ridge functions with ℓ^1 and ℓ^0 controls. *IEEE Transactions on Information Theory*, 64(12):7649–7656, 2018.

[65] Andrei Nikolaevich Kolmogorov. On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition. *Doklady Akademii Nauk*, 114(5):953–956, 1957.

[66] Brian Kulis, Sugato Basu, Inderjit Dhillon, and Raymond Mooney. Semi-supervised graph clustering: A kernel approach. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 457–464, 2005.

[67] N. Kishore Kumar and Jan Schneider. Literature survey on low rank approximation of matrices. *Linear and Multilinear Algebra*, 65(11):2212–2244, 2017.

[68] Věra Kůrková. Kolmogorov's theorem is relevant. *Neural Computation*, 3(4):617–622, 1991.

[69] Věra Kůrková. Kolmogorov's theorem and multilayer neural networks. *Neural Networks*, 5(3):501–506, 1992.

[70] Ming-Jui Lai and David McKenzie. Compressive sensing approach to cut improvement and local clustering. *SIAM Journal on Mathematics of Data Science*, 2:368–395, 2020.

[71] Ming-Jun Lai. Multivariate splines for data fitting and approximation. In M. Neamtu and Larry L. Schumaker, editors, *Approximation Theory XII: San Antonio*, pages 210–228. Nashboro Press, 2008.

[72] Ming-Jun Lai and Jinsil Lee. A multivariate spline based collocation method for numerical solution of partial differential equations. *SIAM Journal on Numerical Analysis*, 60(5):2405–2434, 2022.

[73] Ming-Jun Lai and Larry L. Schumaker. *Spline Functions on Triangulations*. Number 110. Cambridge University Press, 2007.

[74] Ming-Jun Lai and Larry L. Schumaker. A domain decomposition method for computing bivariate spline fits of scattered data. *SIAM Journal on Numerical Analysis*, 47(2):911–928, 2009.

[75] Ming-Jun Lai and Zhaiming Shen. A quasi-orthogonal matching pursuit algorithm for compressive sensing. *arXiv preprint arXiv:2007.09534*, 2020.

[76] Ming-Jun Lai and Zhaiming Shen. The kolmogorov superposition theorem can break the curse of dimensionality when approximating high dimensional functions. *arXiv preprint*, 2021.

[77] Ming-Jun Lai and Zhaiming Shen. A compressed sensing based least squares approach to semi-supervised local cluster extraction. *Journal of Scientific Computing*, 94(3):63, 2023.

[78] Ming-Jun Lai and Zhaiming Shen. The optimal rate for linear kb-splines and lkb-splines approximation of high dimensional continuous functions and its application. *arXiv preprint*, 2024.

[79] Ming-Jun Lai and Li Wang. Bivariate penalized splines for regression. *Statistica Sinica*, pages 1399–1417, 2013.

[80] Ming-Jun Lai and Yang Wang. *Sparse Solutions of Underdetermined Linear Systems and Their Applications*. Society for Industrial and Applied Mathematics, 2021.

[81] Kevin Lang and Satish Rao. A flow-based method for improving the expansion or conductance of graph cuts. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 325–337. Springer Berlin Heidelberg, 2004.

[82] Haifeng Li. Improved analysis of sp and cosamp under total perturbations. *EURASIP Journal on Advances in Signal Processing*, 2016:1–6, 2016.

[83] Yixuan Li, Kun He, David Bindel, and John E Hopcroft. Uncovering the small community structure in large networks: A local spectral approach. In *Proceedings of the 24th international conference on world wide web*, pages 658–668, 2015.

[84] Yixuan Li, Kun He, Kyle Kloster, David Bindel, and John Hopcroft. Local spectral clustering for overlapping community detection. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 12(2):1–27, 2018.

[85] Ji-Nan Lin and Rolf Unbehauen. On the realization of a kolmogorov network. *Neural Computation*, 5(1):18–20, 1993.

[86] Guangcan Liu, Zhouchen Lin, Shuicheng Yan, Ju Sun, Yong Yu, and Yi Ma. Robust recovery of subspace structures by low-rank representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):171–184, 2012.

[87] G. G. Lorentz. Metric entropy, widths, and superpositions of functions. *The American Mathematical Monthly*, 69(6):469–485, 1962.

[88] G. G. Lorentz. *Approximation of Functions: Selected Topics in Mathematics*. 1966.

[89] Jianfeng Lu, Zuowei Shen, Haizhao Yang, and Shijun Zhang. Deep network approximation for smooth functions. *SIAM Journal on Mathematical Analysis*, 53(5):5465–5506, 2021.

[90] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

[91] James MacQueen. Classification and analysis of multivariate observations. In *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1967.

[92] Michael W Mahoney, Lorenzo Orecchia, and Nisheeth K Vishnoi. A local spectral method for graphs: With applications to improving graph partitions and exploring data graphs locally. *The Journal of Machine Learning Research*, 13(1):2339–2365, 2012.

[93] Vitaly E. Maiorov. On best approximation by ridge functions. *Journal of Approximation Theory*, 99(1):68–94, 1999.

[94] Stéphane G Mallat and Zhifeng Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing*, 1993(12):3397–3415, 1993.

[95] Stéphane Mallat. Understanding deep convolutional networks. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374(2065):20150203, 2016.

[96] Daniel McKenzie and Steven Damelin. Power weighted shortest paths for clustering euclidean data. *Foundations of Data Science*, 1(3):307–327, 2019.

[97] Hrushikesh N. Mhaskar and Charles A. Micchelli. Approximation by superposition of sigmoidal and radial basis functions. *Advances in Applied Mathematics*, 13(3):350–373, 1992.

[98] Aleksandr Mikhalev and Ivan V. Oseledets. Rectangular maximum-volume submatrices and their applications. *Linear Algebra and its Applications*, 538:187–211, 2018.

[99] Qun Mo and Yi Shen. A remark on the restricted isometry property in orthogonal matching pursuit. *IEEE Transactions on Information Theory*, 58(6):3654–3656, 2012.

[100] Hadrien Montanelli and Haizhao Yang. Error bounds for deep relu networks using the kolmogorov–arnold superposition theorem. *Neural Networks*, 129:1–6, 2020.

[101] Hadrien Montanelli, Haizhao Yang, and Qiang Du. Deep relu networks overcome the curse of dimensionality for bandlimited functions. *arXiv preprint arXiv:1903.00735*, 2019.

[102] Sidney Morris. Hilbert 13: Are there any genuine continuous multivariate real-valued functions? *Bulletin of the American Mathematical Society*, 58(1):107–118, 2021.

[103] Elchanan Mossel, Joe Neeman, and Allan Sly. A proof of the block model threshold conjecture. *Combinatorica*, 38(3):665–708, 2018.

[104] Deanna Needell and Joel A. Tropp. Cosamp: Iterative signal recovery from incomplete and inaccurate samples. *Applied and Computational Harmonic Analysis*, 26(3):301–321, 2009.

[105] Deanna Needell and Roman Vershynin. Signal recovery from incomplete and inaccurate measurements via regularized orthogonal matching pursuit. *IEEE Journal of selected topics in signal processing*, 4(2):310–316, 2010.

[106] Andrew Ng, Michael Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems*, volume 14, 2001.

[107] Phillip A. Ostrand. Dimension of metric spaces and hilbert's problem 13. pages 619–622, 1965.

[108] Pencho P. Petrushev. Approximation by ridge functions and neural networks. *SIAM Journal on Mathematical Analysis*, 30(1):155–189, 1998.

[109] Alexander Petukhov. Fast implementation of orthogonal greedy algorithm for tight wavelet frames. *Signal Processing*, 86(3):471–479, 2006.

[110] Allan Pinkus. Approximation theory of the mlp model in neural networks. *Acta Numerica*, 8:143–195, 1999.

[111] Michael James David Powell. *Approximation Theory and Methods*. Cambridge University Press, 1981.

[112] Holger Rauhut. Compressive sensing and structured random matrices. *Theoretical foundations and numerical methods for sparse recovery*, 9(1):92, 2010.

[113] Yazhou Ren, Kangrong Hu, Xinyi Dai, Lili Pan, Steven CH Hoi, and Zenglin Xu. Semi-supervised deep embedded clustering. *Neurocomputing*, 325:121–130, 2019.

[114] Fadil Santosa and William W Symes. Linear inversion of band-limited reflection seismograms. *SIAM Journal on Scientific and Statistical Computing*, 7(4):1307–1330, 1986.

[115] Johannes Schmidt-Hieber. The kolmogorov–arnold representation theorem revisited. *Neural Networks*, 137:119–126, 2021.

[116] Larry Schumaker. *Spline Functions: Basic Theory*. Cambridge University Press, 2007.

[117] Zhaiming Shen, Ming-Jun Lai, and Sheng Li. Graph-based semi-supervised local clustering with few labeled nodes. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, 2023.

[118] Zuowei Shen, Haizhao Yang, and Shijun Zhang. Neural network approximation: Three hidden layers are enough. *Neural Networks*, 141:160–173, 2021.

[119] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[120] Pan Shi, Kun He, David Bindel, and John E. Hopcroft. Locally-biased spectral approximation for community detection. *Knowledge-Based Systems*, 164:459–472, 2019.

[121] Jonathan W. Siegel and Jinchao Xu. Approximation rates for neural networks with general activation functions. *Neural Networks*, 128:313–321, 2020.

[122] Jonathan W. Siegel and Jinchao Xu. High-order approximation rates for shallow neural networks with cosine and $relu^k$ activation functions. *Applied and Computational Harmonic Analysis*, 58:1–26, 2022.

[123] Sho Sonoda and Noboru Murata. Neural network with unbounded activation functions is universal approximator. *Applied and Computational Harmonic Analysis*, 43(2):233–268, 2017.

[124] Daniel A Spielman and Shang-Hua Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 81–90, 2004.

[125] Daniel A Spielman and Shang-Hua Teng. A local clustering algorithm for massive graphs and its application to nearly linear time graph partitioning. *SIAM Journal on Computing*, 42(1):1–26, 2013.

[126] David Sprecher. *Ph.D. Dissertation*. PhD thesis, University of Maryland, 1963.

[127] David Sprecher. On the structure of continuous functions of several variables. *Transactions of the American Mathematical Society*, 115:340–355, 1965.

[128] David Sprecher. A representation theorem for continuous functions of several variables. *Proceedings of the American Mathematical Society*, 16:200–203, 1965.

[129] David Sprecher. On the structure of representation of continuous functions of several variables as finite sums of continuous functions of one variable. *Proceedings of the American Mathematical Society*, 17(1):98–105, 1966.

[130] David Sprecher. An improvement in the superposition theorem of kolmogorov. *Journal of Mathematical Analysis and Applications*, 38(1):208–213, 1972.

[131] Y. Sternfeld. Dimension, superposition of functions, and separation of points, in compact metric spaces. *Israel Journal of Mathematics*, 50:13–53, 1985.

[132] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58(1):267–288, 1996.

[133] Joel A Tropp. Greed is good: Algorithmic results for sparse approximation. *IEEE Transactions on Information theory*, 50(10):2231–2242, 2004.

[134] Anton Tsitsulin, John Palowitch, Bryan Perozzi, and Emmanuel Müller. Graph clustering with graph neural networks. *Journal of Machine Learning Research*, 24(127):1–21, 2023.

[135] Nate Veldt, David Gleich, and Michael Mahoney. A simple and strongly-local flow-based method for cut improvement. In *International Conference on Machine Learning*, pages 1938–1947. PMLR, 2016.

[136] A. G. Vitushkin. On the hilbert's thirteenth problem. *Soviet Mathematics Doklady*, 95:701–704, 1954.

[137] Anatoliy Georgievich Vitushkin. A proof of the existence of analytic functions of several variables not representable by linear superpositions of continuously differentiable functions of fewer variables. *Doklady Akademii Nauk*, 156(6):1258–1261, 1964.

[138] Manfred von Golitschek, Ming-Jun Lai, and Larry L. Schumaker. Error bounds for minimal energy bivariate polynomial splines. *Numerische Mathematik*, 93(2):315–331, 2002.

[139] Di Wang, Kimon Fountoulakis, Monika Henzinger, Michael W Mahoney, and Satish Rao. Capacity releasing diffusion for speed and locality. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 3598–3607, 2017.

[140] Jian Wang, Seokbeop Kwon, and Byonghyo Shim. Generalized orthogonal matching pursuit. *IEEE Transactions on Signal Processing*, 60(12):6202–6216, 2012.

[141] L. Wang, G. Wang, Ming-Jun Lai, and L. Gao. Efficient estimation of partially linear models for spatial data over complex domains. *Statistica Sinica*, 30:347–360, 2020.

[142] Jinming Wen, Zhengchun Zhou, Zilong Liu, Ming-Jun Lai, and Xiaohu Tang. Sharp sufficient conditions for stable recovery of block sparse signals by block orthogonal matching pursuit. *Applied and Computational Harmonic Analysis*, 47(3):948–974, 2019.

[143] James D. Wilson, Simi Wang, Peter J. Mucha, Shankar Bhamidi, and Andrew B. Nobel. A testing-based extraction algorithm for identifying significant communities in networks. *The Annals of Applied Statistics*, 8:1853–1891, 2014.

[144] Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *International Conference on Machine Learning*, pages 478–487. PMLR, 2016.

[145] Dmitry Yarotsky. Error bounds for approximations with deep relu networks. *Neural Networks*, 94:103–114, 2017.

[146] Dmitry Yarotsky and Anton Zhevnerchuk. The phase diagram of approximation rates for deep neural networks. In *Advances in Neural Information Processing Systems*, volume 33, pages 13005–13015, 2020.

[147] Lihi Zelnik-Manor and Pietro Perona. Self-tuning spectral clustering. In *Advances in Neural Information Processing Systems*, volume 17, 2004.

[148] Yunpeng Zhao, Elizaveta Levina, and Ji Zhu. Community extraction for social networks. *Proceedings of the National Academy of Sciences*, 108(18):7321–7326, 2011.