
KADIR HAS UNIVERSITY
SCHOOL OF GRADUATE STUDIES
PROGRAM OF MANAGEMENT INFORMATION SYSTEMS

# DEVELOPMENT OF SCALABLE MANIFOLD LEARNING LIBRARY: SCAMAN

BERKE PEHLİVAN

MASTER OF SCIENCE THESIS

ISTANBUL, FEBRUARY, 2024

# DEVELOPMENT OF SCALABLE MANIFOLD LEARNING LIBRARY: SCAMAN

BERKE PEHLİVAN

A thesis submitted to
the School of Graduate Studies of Kadir Has University
in partial fulfilment of the requirements for the degree of
Master of Science in
Management Information Systems

İstanbul, February, 2024

# APPROVAL

This thesis titled DEVELOPMENT OF SCALABLE MANIFOLD LEARNING LIBRARY: SCAMAN submitted by BERKE PEHLİVAN, in partial fulfillment of the requirements for the degree of Master of Science in Management Information Systems is approved by

Asst. Prof. Dr. E. Fatih YETKİN (Advisor) .....
Kadir Has University

Assoc. Prof. Dr. Oğuzhan CEYLAN .....
Kadir Has University

Asst. Prof. Dr. Kerem ALTUN .....
Işık University

I confirm that the signatures above belong to the aforementioned faculty members.

.....
Prof. Dr. Mehmet Timur Aydemir
Dean of School of Graduate Studies
Date of Approval: 02.02.2024

## DECLARATION ON RESEARCH ETHICS AND PUBLISHING METHODS

I, BERKE PEHLİVAN; hereby declare

- that this Master of Science Thesis that I have submitted is entirely my own work and I have cited and referenced all material and results that are not my own in accordance with the rules;
- that this Master of Science Thesis does not contain any material from any research submitted or accepted to obtain a degree or diploma at another educational institution;
- and that I commit and undertake to follow the "Kadir Has University Academic Codes and Conduct" prepared in accordance with the "Higher Education Council Codes of Conduct".

In addition, I acknowledge that any claim of irregularity that may arise in relation to this work will result in a disciplinary action in accordance with university legislation.

BERKE PEHLİVAN

.....

02.02.2024

To my family, for their endless support and love...

# ACKNOWLEDGEMENT

Bu tezin hazırlanmasında bana destek olan herkese teşekkür ederim. Özellikle danışmanım E.Fatih Yetkin'e değerli rehberliği için, ve sevgili aileme sabırları ve sonsuz desteği için minnettarım.

Bu tez çalışması, TÜBİTAK tarafından 120E281 koduyla 'Büyük Verilerin Manifold Öğrenme ile Analizi için Özdeğer Dağılımı Dilimlenmesi ve Kontür İntegraline Dayalı Yeni Bir Yöntemin Geliştirilmesi' isimli 1001 projesi kapsamında desteklenmiştir.

DEVELOPMENT OF SCALABLE MANIFOLD LEARNING LIBRARY: SCAMAN

# ABSTRACT

This thesis presents an exploration of manifold learning and dimensionality reduction techniques, which are crucial in the fields of data science and machine learning. The center of this study is the development and evaluation of 'Scaman (Scalable Manifold Library), a Python-based computational tool designed to implement these techniques. This thesis investigates the key manifold learning algorithms. Including PCA,MDS, LE, and LLE and emphasizing the importance of eigenvalue solvers in these algorithms. The contribution of this thesis is the integration of advanced eigensolvers like NumPy, SLEPc and FEAST into key manifold algorithms within scaman package. The empirical analysis was conducted using various synthetic and real-world datasets. Those analyses focused on the efficiency, accuracy, and practical utility of scaman in different scenarios. Results demonstrate the tool's effectiveness, especially in handling large datasets. The advantages of FLANN and SLEPc prove scaman's efficiency in the creation of adjacency matrices and eigenvalue computation. The outcome of this thesis provides a computational tool for researchers and practitioners. Future directions include expanding the tool's capabilities by adding more algorithms, improving scalability, and applying various domain specific data-driven scenarios.

**Keywords:Data Science,Manifold Learning, Dimensionality Reduction, Feature Extraction, Eigensolvers, Eigenvalues, High-Dimensional Data Analysis, Intrinsic Dimension Estimator.**

ÖLÇEKLENEBİLİR MANİFOLD ÖĞRENME KÜTÜPHANESİNİN GELİŞTİRİLMESİ: SCAMAN

# ÖZET

Bu tez, veri bilimi ve makine öğrenimi alanlarında önemli olan manifold öğrenme ve boyutluluk azaltma tekniklerinin bir incelemesini sunmaktadır. Bu çalışmanın merkezi, bu teknikleri uygulamak için tasarlanmış Python tabanlı bir hesaplama aracı olan 'Scaman'ın (Ölçeklenebilir Manifold Kütüphanesi) geliştirilmesidir. Bu tez, temel manifold öğrenme algoritmalarını incelemektedir. PCA, MDS, LE ve LLE'nin dahil edilmesi ve bu algoritmalarda özdeğer çözücülerin öneminin vurgulanması bu tezde amaçlanmıştır. Bu tezin katkısı, NumPy, SLEPC ve FEAST gibi gelişmiş özdeğer çözücülerin, Scaman paketi içindeki manifold algoritmalarına entegrasyonudur. Ampirik analiz, çeşitli sentetik ve gerçek dünya veri kümeleri kullanılarak gerçekleştirilmiştir. Bu analizler, farklı senaryolarda Scaman'ın verimliliğine, doğruluğuna ve pratik faydasına odaklanmıştır. Sonuçlar, aracın özellikle büyük veri kümelerinin işlenmesindeki etkinliğini göstermektedir. FLANN ve SLEPc gibi kütüphanelerin avantajları, Scaman'ın komşuluk matrislerin oluşturulmasında ve özdeğer hesaplamasında verimliliğini kanıtlar. Bu tezin bir çıktısı, araştırmacılar ve uygulayıcılar için hesaplamalı bir araç sağlamaktadır. Gelecek planları arasında, daha fazla algoritma ekleyerek, ölçeklenebilirliği geliştirerek ve çeşitli alanlara özgü veri odaklı senaryolar uygulayarak Scaman'ı genişletme yer almakltadır.

**Anahtar Sözcükler: Veri Bilimi, Manifold Öğrenme, Boyut İndirgeme, Öznitelik Çıkarma, Özdeğer Çözücüler, Yüksek Boyutlu Veri Analizi, Özaltuzay Tahminleyici**

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# LIST OF SYMBOLS

| | |
|---|---|
| $X$ | Dataset with $n$ samples and $d$ dimensions |
| $\bar{X}$ | Mean vector of the dataset |
| $C$ | Covariance matrix |
| $\mathbf{v}_i$ | Eigenvector of matrix $C$ |
| $\lambda_i$ | Eigenvalue corresponding to eigenvector $\mathbf{v}_i$ |
| $W$ | Projection matrix formed by eigenvectors |
| $Y$ | Lower-dimensional space |
| $D$ | Distance matrix |
| $J$ | Centering matrix |
| $B$ | Double centered matrix |
| $G$ | Graph constructed in manifold learning |
| $W_{ij}$ | Weight assigned to the edge in the graph |
| $L$ | Graph Laplacian |
| $t$ | Tuning parameter in Laplacian Eigenmaps |
| $\alpha$ | Parameter or variable represented by alpha |

# LIST OF ACRONYMS AND ABBREVIATIONS

| | |
|---|---|
| FLANN | Fast Library for Approximate Nearest Neighbors |
| ID | Intrinsic Dimension |
| k-NN | k-Nearest Neighbors |
| LE | Laplacian Eigenmaps |
| LLE | Locally Linear Embedding |
| MDS | Multidimensional Scaling |
| MPI | Message Passing Interface |
| NumPy | Numerical Python Library |
| PCA | Principal Component Analysis |
| PETSc | Portable Extensible Toolkit for Scientific Computation |
| scaman | Scalable Manifold Library |
| SciPy | Scientific Python Library |
| SLEPc | Scalable Library for Eigenvalue Problem Computations |

# 1. INTRODUCTION

This thesis specifically approaches the problem from the direction of manifold learning and dimensionality reduction algorithms. These algorithms occupy an essential place in developing ways to process complex relationships hidden beneath vast amounts of data. Manifold learning, especially, stands out as a complex but very additive part of the subset of techniques to reduce dimension. Indeed, while linear methods can reduce space and be very convenient for further computations, manifold learning can reveal the nature of data and allow us to observe their intrinsic characteristics. (Pedregosa et al. 2011).

In today's data-driven world, the methods of manifold learning are especially vital. With huge amounts of complex data from bioinformatics, financial analysis, and social network analysis, the desired low-dimensional structure of the data is often hidden underneath numerous dimensions. The structure cannot be detected with simple linear means, as most relationships between different data points are nonlinear and thus not appropriate for one-size-fits-all models. For this reason, we use manifold learning to reveal the deeper structure beneath the high-dimensional surface of the data.

This thesis studies several of the most prominent algorithms coming from unsupervised dimension reduction. First of all, we consider Principal Component Analysis (PCA), which is a linear technique but is widely used due to algorithmic simplicity and effectiveness in revealing the axes of maximum variance in data, thereby making data interpretation more manageable. Secondly, Multidimensional Scaling (MDS) is considered a measure of the amount of dissimilarity in the data, and, therefore, a technique for visualizing this measure. As a result, it allows for the investigation of complex, nonlinear structures. Thirdly, Laplacian Eigenmaps (LE) is an algorithm preserving neighborhood information, which, consequently, allows for the study of

data manifolds' intrinsic geometry. Finally, Locally Linear Embedding (LLE) is a model that reconstructs high-dimensional data in lower dimensions while maintaining the relationship of all data points to their nearest neighbors.

The eigensolver is a key element of such algorithms. This thesis aims at the detailed study of numerous types of eigensolvers, and the specific case of their usage in manifold learning algorithms, like PCA. Iterative, and direct ones, computational complexity, and implementation peculiarities are to be discussed. This understanding allows better addressing high-dimensional data, which ultimately results in more accurate and wise data analysis, applicable in various fields. This work and its investigation are related to data science's domain works, which allows us to explore multiple formalized discoveries.

### 1.1 Background and Significance of the Study

Due to the complexity of data science today, we are often overwhelmed by the amount of information. This trend can be applied in fields as diverse as biology, financial markets, and social media analytics. High-dimensional datasets are characterized by their size and complexity and are not only large but also contain a lot of information complexity. This complexity often creates a barrier to extracting discoverable information and patterns, pushing traditional data analysis methods into the background. (Berman 2018; Halevy, Norvig, and Pereira 2009).

The importance of dimensionality reduction strategies, particularly manifold learning, is emphasized in this thesis. A more sophisticated method of reducing dimensionality is manifold learning. It is very good at revealing hidden patterns and structures in high-dimensional data. It offers a more in-depth understanding of the data while maintaining its intrinsic geometric properties, going beyond the capabilities of conventional linear techniques.

Key algorithms that successfully advance manifold learning are the focus of this study. Finding the axes of maximum variance in data can be done more easily by

using PCA. Complex data structures are analyzed by MDS. The ability of LE to maintain neighborhood information while exposing the inherent geometry of data manifolds is discussed. Lastly, the ability of LLE to reconstruct high-dimensional data in lower dimensions while maintaining the connections between each data point and its nearest neighbors is evaluated.

This thesis aims to explain the importance of selected manifold learning approaches. Improved high-dimensional, complex data analysis is the goal of this thesis. The Scaman package is developed to advance data science and manifold learning.

## 1.2 Research Objectives and Questions

The main purpose of this research is to deepen the understanding of manifold learning techniques and eigensolvers in the field of data science and to develop their practical applications. This goal is achieved by carefully investigating various dimensionality reduction techniques that have made a significant impact on this field. Particular emphasis is placed on the analysis of eigensolvers, particularly their role and effectiveness within manifold learning algorithms.

The main contribution of this thesis is the introduction of 'Scaman'. Scaman is a computational tool specifically developed to bridge the theoretical and practical aspects of dimensionality reduction. Scaman looks set to contribute to the translation of complex mathematical theories into practical computational applications. Scaman aims potentially making advanced data analysis techniques more accessible and efficient.

The research is guided by several fundamental questions:

- How can manifold learning techniques be effectively applied to manage and interpret high dimensional data? Can Scaman interpret high dimensional data? This question explores practical aspects. Applying manifold learning algorithms to real-world data and its ability to reveal hidden patterns.

- In what ways do eigensolvers contribute to the efficiency and effectiveness of manifold learning algorithms?
- How does Scaman facilitate manifold learning and application of dimensions reduction techniques in data science? This question examines Scaman's functionality and impact assesses its ability to complicate dimensionality reduction techniques that are more accessible and practical for data scientists.

The study aims to contribute to the field of data science by answering these questions. It aims to provide a better understanding of how manifold learning techniques and eigensolvers can be used to address the complexity of high-dimensional data. Additionally, providing more conscious and accurate data analysis across multiple disciplines is another aspect of this study. The ultimate goal is to bridge the gap between theoretical data science methodologies and their practical applications. Thus, Scaman python package is developed.

### 1.3 Structure and Organization of the Thesis

The structure of this thesis:

**Literature Review.** This chapter can be considered the foundation of this study with reviewing existing literature on manifold learning. It includes the historical development of manifold learning concepts and basic theoretical approaches. This kind of review is crucial to understanding what this study aims to address by identifying the concepts in manifold learning.

**Methodology.** Here, we will examine the specific computational kernels used in this research. This chapter explains the mathematical foundations of the algorithms implemented in Scaman. PCA, MDS, LE, and LLE algorithms are explained. Also, different eigensolvers are explained in this part to understand how they differ for specific types of problems. This part is important to understand the technical aspects of the research and the rationale behind why we chose these algorithms.

**Scaman: Architecture and Functionalities.** This chapter introduces the Scaman tool developed for this research. Detailed explanations of the architecture of the Scaman package are explained here. Technical specifications, functionalities, user interface, installation steps, and structure of Scaman discussed in this section. Also, Scaman's role in bridging the gap between theory and practice is explained in this chapter.

**Numerical Experiments.** In this chapter of the thesis, the results of numerical experiments of Scaman is presented. This empirical analysis is important to show Scaman's capabilities on high-dimensional data. It includes several tests with both synthetic and real-world data to highlight tool's efficiency and accuracy against existing tools.

**Discussion and Conclusion.** The final chapter synthesizes and discusses the findings from numerical experiments and previous chapters. It also provides a critical evaluation of research and discussing the limitations. Lastly, future directions are discussed, and suggestions are made to improve the Scaman tool.

This thesis uses a structured approach to provide a clear exploration of manifold learning methods. The goal of this study is to provide insights into the field of data science.

# 2. THEORETICAL FRAMEWORK AND LITERATURE REVIEW

Manifold learning and dimensionality reduction algorithms are important tools in data science and machine learning. These algorithms are used to simplify and interpret complex, high-dimensional data. With a focus on significant algorithms like PCA, MDS, Isomap, LE, and LLE this literature review tries to methodically examine research in this field. With their mathematical underpinnings, each technique makes a substantial contribution to the field. These algorithms provide better ways to simplify large data sets. This review also looks at direct and iterative approaches and how they are used in manifold learning. Also, this section highlights the critical role eigensolvers play in these processes. It is also demonstrated how these approaches are integrated into the Scaman package. Demonstrating how eigenvalue solvers such as Numpy, SLEPc, and FEAST efficiently close the gap between computational applications and theoretical mathematics. This section attempts to provide a basic understanding of the current state of dimensionality reduction techniques available and their impact on data science.

## 2.1 Evolution of Manifold Learning and Dimensionality Reduction

There have been a lot of developments in data science and machine learning along the way of manifold learning and dimensionality reduction. Essential to the reduction of high-dimensional data, these methods have developed to meet the growing complexity of datasets across domains (Cayton 2005; van der Maaten, Postma, and van den Herik 2009).

- The foundational linear technique of Principal Component Analysis (PCA) has played a pivotal role in facilitating the reduction of dimensionality. By

determining the axes of maximum variance—which are calculated using the covariance matrix's eigenvectors and eigenvalues—it makes the interpretation of data easier. According to Jolliffe (2002), this method successfully extracts the most important characteristics from the data.

- Multidimensional Scaling (MDS) was developed as an important advancement. With its ability to represent distances and dissimilarities, MDS is particularly effective at handling nonlinear structures. When it comes to data visualization and analysis, MDS shows its capabilities with understand non-linear data. With extending the application of dimensionality reduction, this method allows a more sophisticated understanding of complex datasets.
- Another advancement in this field is represented by Isometric Mapping. Isomap unfolds nonlinear manifolds efficiently by constructing a neighborhood graph and incorporating geodesic distances between data points. Tenenbaum, De Silva, and Langford (2000) describe this technique in detail. It provides a global geometric framework for nonlinear dimensionality reduction, which improves how we understand of complex data structures.
- Locally Linear Embedding (LLE) and Laplacian Eigenmaps (LE) enhance the manifold learning toolkit even more. By computing the graph Laplacian and its eigenvectors, LE, which focuses on local neighborhood data, reveals the intrinsic geometry of data manifolds (Belkin  Niyogi, 2003). Conversely, LLE provides a distinct viewpoint on the structure of the data by reconstructing high-dimensional data in lower dimensions while maintaining local relationships (Roweis  Saul, 2000).

## 2.2 Role and Evolution of Eigensolvers in Data Science

Eigensolvers are fundamental tools in numerous fields of mathematics and computational science. Particularly in data science where they are used to compute eigenvalues and eigenvectors of matrices (Stewart 2001; Parlett 1998). The evolution and importance of eigensolvers are discussed in this section with a focus on manifold learning algorithms. Additionally, the incorporation of sophisticated eigensolvers

such as Numpy, SLEPc, and FEAST into these processes is examined and can be seen in Table 2.1.

- Numpy is a foundational package for scientific computing in Python. Eigenvalue solvers in Numpy are effective and easy to use. According to Harris et al. (2020), the numpy.linalg.eig function is frequently used both in academic and industrial studies to solve eigenvalue problems in smaller matrices.
- PETSc's ability to solve complex eigenvalue problems on a large scale is enhanced by SLEPc (Scalable Library for Eigenvalue Problem Computations). Hernandez et al. (2005) describe that it is especially effective at solving problems that are too big for traditional dense matrix solvers and provides a range of algorithms for various kinds of eigenvalue problems.
- Large-scale, dense, and sparse eigenproblems can be effectively solved by the high-performance eigenvalue solver FEAST. It is known for its durability and effectiveness. Large-scale scientific and engineering applications can benefit from FEAST's highly parallelizable computations due to its unique contour integral approach, as noted by Polizzi (2009).

**Table 2.1** Comparisons of Eigensolvers offered in Scaman

| **Eigenvalue Solver** | **Scalability** | **Specialized Algorithms** |
|---|---|---|
| Numpy | Low | QR Decomposition |
| Scipy | Medium | ARPACK, LOBPCG |
| SLEPc | High | Various |
| FEAST | High | Contour Integration |

Depending on particular problem requirements such as matrix size, the need for parallel computation, and the nature of the eigenvalue problem, one can choose between these solvers. This literature review presents eigenvalue solvers as essential instruments in data science, explaining mathematical ideas with computational solutions related to data analysis and dimensionality reduction.

## 2.3 Literature Review

Enhancing the effectiveness and accuracy of manifold learning and dimensionality reduction methods for managing large and complex datasets have been the main focus of recent studies. An enhanced version of t-SNE was presented by Smith and Lee in 2021; it dramatically cuts down on computation time without sacrificing the quality of the visualizations (Smith and Lee 2021). The creation of UMAP (Uniform Manifold Approximation and Projection) by McInnes et al. is another noteworthy development. It has been gaining popularity because of its efficacy of preserving local and global data structures (McInnes, Healy, and Melville 2020).

Additionally, recent studies in manifold learning show an integration with deep learning techniques to improve pattern recognition, especially for high-dimensional data. A notable work of Hinton and Salakhutdinov (2006) demonstrates the effectiveness of using autoencoders for dimensionality reduction. Their approach shows example usage in such areas as bioinformatics where dealing with complex data. In addition to that, a study by Belkin and Niyogi (2003) on Laplacian Eigenmaps has been instrumental to overcome scalability issues in manifold learning. This study offers a framework especially for large datasets. These studies show the growing applicability of manifold learning in various domains of data science.

Methods for reducing dimensionality have been compared in numerous studies. According to these studies (Karnick, 2016; Chenna, 2016; Meier, 2017; Ayesha, 2020), this process is crucial for data visualization and analysis. These techniques, which include PCA, SVD, DBN, and stacked autoencoders, are intended to reduce the number of features in high-dimensional data while minimizing information loss (Chenna, 2016). With a few noteworthy outliers, Meier (2017) discovered that mMDS, GPLVM, and PCA were the most effective methods. In order to accelerate the extraction and processing of information, Ayesha (2020) underlined that these methods need to be carefully selected. These recent contributions show how manifold learning is evolving and growing.

## 2.4 Advantages of Intrinsic Dimension Estimation in Data Analysis

During the study of data science, intrinsic dimension (ID) estimation is widely applied, specifically in analyzing high-dimensional data. This technique provides a lot of advantages that help to keep data analysis more streamlined and improved. By performing an accurate estimate of dataset's intrinsic dimensionality, it enhances insight into its fundamental structure which results to better choices and analysis outputs for several purposes.

Intrinsic dimensionality estimation has many benefits in reducing dimensions. In Bruske (1997), such linear time complexity, insensitivity to noise and easy-to-use properties are emphasized while considering further processing as well as other issues related to the data sets. Local intrinsic dimension estimation applications have been extended by Carter (2010) to encompass statistical manifold learning, network anomaly detection, clustering and image segmentation. Lastly, Karantzalos (2009) emphasizes on the importance of scale space filtering improving both accuracy of ID and dimensionality reduction. In summary, these studies demonstrate how important intrinsic dimension estimation is as a tool to reduce dimensions.

**Enhanced Computational Efficiency.** The technique of intrinsic dimension estimation identifies the smallest dimensions that capture the basic structure of the data (Camastra and Vinciarelli 2002). By contrast to traditional ways that usually involve a weary iteration over a range of dimension numbers until an optimal dimensionality is identified, this method does not involve any such iterative processes. Therefore, traditional techniques may be expensive in terms of computational cost due to their reliance on high-dimensional data.

**Improved Data Visualization.** Visualizing high-dimensional data can be difficult because it often obscures patterns and relationships. Intrinsic dimension estimation helps with this by identifying a lower-dimensional space that contains essential features and thus enables interpretable visualizations (Paulovich et al. 2008).

**Optimized Model Performance.** In machine learning and statistical modeling, overfitting and poor model performance are typical when the curse of dimensionality occurs. Through determining what the true data dimension really is, intrinsic dimension estimation solves these problems by helping choose an appropriate complexity for a given model (Kegl 2003). The optimization of model parameters based on intrinsic dimension may significantly improve model accuracy.

**Facilitated Data Compression and Storage.** The accuracy of the estimation of the intrinsic dimensionality of a dataset has positive implications for data compression and storage. This is because identifying minimal dimensions required for accurate representation of data helps in efficient compression schemes, leading to low storage space requirement and high-speed retrieval and processing (Bishop 2006).

# 3. METHODOLOGY

## 3.1 Overview of Dimensionality Reduction Techniques

The linear and non-linear dimensionality reduction methods will be reviewed in this subsection with an emphasis on PCA, MDS, LE, and LLE. We will talk about the computational kernels of these algorithms, emphasizing their theoretical foundations and real-world applications

### 3.1.1 Linear Methods: Principal Component Analysis (PCA) and Multidimensional Scaling (MDS)

The operational mechanics of PCA and MDS are explained in detail by means of mathematical expressions in this subsection.

**PCA (Principal Component Analysis).** PCA maximizes variance preservation while reducing the dimensionality of the data. The following steps are involved in the mathematical process:

1. Covariance Matrix Computation: Given a dataset $X$ with $n$ samples and $d$ dimensions, compute the covariance matrix $C$:

$$C = \frac{1}{n-1}(X - \bar{X})^T(X - \bar{X}) \tag{3.1}$$

   where $\bar{X}$ is the mean vector of the dataset.

2. Eigenvalue Decomposition: Perform eigenvalue decomposition on the covariance matrix $C$:

$$C\mathbf{v}_i = \lambda_i \mathbf{v}_i \tag{3.2}$$

   where $\lambda_i$ and $\mathbf{v}_i$ are the eigenvalues and eigenvectors of $C$, respectively.

3. Selection of Principal Components: Choose the top $k$ eigenvectors (where $k < d$) that correspond to the largest eigenvalues to form the projection matrix $W$.
4. Transformation: Transform the original dataset $X$ into a lower-dimensional space $Y$ using $W$:

$$Y = XW \tag{3.3}$$

**MDS (Multidimensional Scaling).** Pairwise distances between points are preserved in a low-dimensional representation of the data that is sought after by MDS. One way to sum up the procedure is:

1. Distance Matrix Calculation: Compute the distance matrix $D$ for the dataset, where $D_{ij}$ represents the distance between points $i$ and $j$.
2. Double Centering: Transform $D$ into a matrix $B$ using double centering:

$$B = -\frac{1}{2} J D^2 J \tag{3.4}$$

where $J$ is the centering matrix,

$$J = I - \frac{1}{n}\mathbf{1}\mathbf{1}^T \tag{3.5}$$

with $I$ being the identity matrix and $\mathbf{1}$ a vector of ones.
3. Eigenvalue Decomposition: Perform eigenvalue decomposition on $B$:

$$B\mathbf{v}_i = \lambda_i \mathbf{v}_i \tag{3.6}$$

4. Low-Dimensional Embedding: Select the top $k$ eigenvectors to form the embedding in $k$-dimensional space.

### 3.1.2 Non-linear Methods: Laplacian Eigenmaps (LE) and Locally Linear Embedding (LLE)

The mathematical formulations of non-linear dimensionality reduction methods, namely Locally Linear Embedding (LLE) and Laplacian Eigenmaps (LE), are examined in this subsection.

**Laplacian Eigenmaps (LE).** Local neighborhood information preservation and exposing the inherent geometry of the data manifold are the main goals of LE:

1. Neighborhood Graph Construction: Construct a graph $G$ where each data point is a node. Connect nodes $i$ and $j$ if they are 'neighbors' (e.g., $k$-nearest neighbors or within a certain radius).
2. Weight Matrix Creation: Assign weights to the edges of the graph, typically using the heat kernel:
$$W_{ij} = \exp\left(-\frac{\|x_i - x_j\|^2}{t}\right) \tag{3.7}$$
where $x_i$ and $x_j$ are data points, and $t$ is a tuning parameter.
3. Graph Laplacian Calculation: Compute the graph Laplacian $L$, defined as
$$L = D - W \tag{3.8}$$
where $D$ is a diagonal matrix with $D_{ii} = \sum_j W_{ij}$.
4. Eigenvalue Problem: Solve the generalized eigenvalue problem:
$$L\mathbf{v} = \lambda D\mathbf{v} \tag{3.9}$$
to find the eigenvectors $\mathbf{v}$.
5. Low-Dimensional Representation: The bottom $k$ non-zero eigenvectors provide the embedding in $k$-dimensional space.

**Locally Linear Embedding (LLE).** Through the reconstruction of each point as a linear combination of its neighbors, LLE seeks to maintain local relationships within the data. The steps are as follows:

1. Nearest Neighbors Identification: For each data point $x_i$, identify its $k$-nearest neighbors.
2. Reconstruction Weights Calculation: Compute the weights $W_{ij}$ that best reconstruct each data point $x_i$ from its neighbors, minimizing the cost:
$$\min_W \sum_i \|x_i - \sum_j W_{ij} x_j\|^2 \tag{3.10}$$
subject to the constraint $\sum_j W_{ij} = 1$ for each $i$.

3. Embedding Computation: Find the vectors $Y_i$ in the lower-dimensional space that best preserve the local geometry represented by the weights $W$, by minimizing:

$$\min_{Y} \sum_{i} \|Y_i - \sum_{j} W_{ij} Y_j\|^2 \tag{3.11}$$

## 3.2 Computational Approaches in Manifold Learning

The computational approaches and difficulties involved in putting the previously discussed dimensionality reduction techniques into practice are examined in this section.

### 3.2.1 Efficient Calculation Strategies for Computational Kernels

Dimensionality reduction requires efficient computation, particularly when working with large datasets. The main objective of this subsection is to optimize the handling of eigenvalue problems that are inherent in PCA, MDS, LE, and LLE, while also addressing their computational complexities.

- PCA and MDS: Eigenvalue decomposition, which can be computationally difficult for large matrices. Eigenvalue decomposition is a part of both PCA and MDS. Techniques like randomized algorithms for approximate eigenvalue decomposition drastically cut down on computation time without sacrificing accuracy. These algorithms can be used to handle these problems efficiently. Moreover, calculations can be accelerated by utilizing parallel computing and optimized libraries like NVIDIA cuBLAS or Intel MKL (Patterson and Hennessy 2017; NVIDIA 2021).
- LE and LLE: Creating neighborhood graphs and solving sparse eigenvalue problems includes computational challenges. K-d trees or ball trees enable effective graph construction for neighbor searches. Especcialy for large sparse matrices, iterative solvers like the Lanczos or Arnoldi methods outperform dense eigensolvers for sparse eigenvalue problems. Optimal implementations

are made available by libraries like SLEPc and ARPACK (Lehoucq, Sorensen, and Yang 1998; Hernandez, Roman, and Vidal 2005).

### 3.2.2 Comparative Analysis with Existing Studies

In order to evaluate the effectiveness and dependability of the computational strategies suggested in this thesis, a comparison with previous research is necessary.

- Benchmark Against Current Tools: We can evaluate the efficacy of our optimization strategies by contrasting the computational accuracy and efficiency of our implementations with those of current tools such as "sklearn", "Megaman", and "Networkx". The comparison will be centered around factors like low-dimensional embedding accuracy, memory usage, and computation time.
- Case Studies and Datasets: A range of datasets, from small-scale to large-scale scenarios, will be used in the comparative analysis. This will contain real-world datasets to show practical applicability as well as synthetic datasets to control for particular data characteristics.

## 3.3 Implementation

The specific algorithmic foundations of the dimensionality reduction strategies discussed and their real-world application in the context of manifold learning are the main topics of this section.

### 3.3.1 Dense and Sparse Eigenvalue Problems

Eigenvalue problems, which fall into two general categories: dense and sparse, each with its own set of computational difficulties and approaches, are crucial to the execution of PCA, MDS, LE, and LLE. These dense and sparse problems can be summarized as:

- PCA and MDS Dense Eigenvalue Problems: These problems usually includes calculating the eigenvalues and eigenvectors of dense matrices (e.g., distance matrices in MDS, and covariance matrices in PCA). Optimal numerical libraries, like ARPACK or LAPACK, which offer effective procedures for dense matrix operations, will be utilized in the implementation (Anderson et al. 1999; Lehoucq, Sorensen, and Yang 1998). The scalability of these techniques for large datasets will receive particular attention (Hernandez, Roman, and Vidal 2005; Saad 2011).
- LE and LLE Sparse Eigenvalue Problems: On the other hand, sparse matrices are frequently handled in LE and LLE, particularly when building neighborhood graphs. Sparse matrix libraries such as ARPACK or SciPy in Python will be necessary for the efficient handling of these problems. The emphasis will be on improving computational efficiency and memory usage by applying iterative techniques that work better with sparse matrices, like the Lanczos or Arnoldi algorithms.

### 3.3.2 Constructing Neighborhood Matrices

Building neighborhood matrices showing the local relationships within the data is an important step in non-linear methods like LE and LLE.

- Effective Neighborhood Graph Construction: To achieve effective neighborhood graph construction, the implementation will use advanced data structures and algorithms. We'll use methods like ball trees and k-d trees. These methods work especially well with high-dimensional data. Libraries like Scikit-learn offer scalable and reliable implementations of these techniques.
- Handling Large Datasets: Building neighborhood matrices can become computationally demanding for large datasets. One way to deal with this is to use approximate nearest-neighbor techniques. These techniques can be found in the FLANN. FLANN library can cut down on computation time without sacrificing accuracy (Muja and Lowe 2014).

### 3.3.3 Intrinsic Dimension Estimation

Intrinsic Dimension (ID) estimation, is integral to dimensionality reduction and particularly relevant in manifold learning. Intrinsic Dimension estimation is defined as the minimum number of coordinates required to accurately represent data without significant information loss. Mathematically, if a dataset is presumed to lie on a manifold within a higher-dimensional space, the intrinsic dimension is the dimensionality of this manifold (Lee and Verleysen, 2007).

ID estimation is suggested by the methodology presented in Özçoban, Manguoğlu, and Yetkin's paper "A Novel Approach for Intrinsic Dimension Estimation via Ritz Values and Orthogonal Polynomials" (2023). This approach may avoid the computational burden of eigenvalue calculations that is frequently associated with conventional approaches. Moreover, the main objective of this approach is to estimate the covariance matrix trace, or $tr(C)$, where $C$ is the covariance matrix. With Rademacher random variables, the trace—the total of $C$'s eigenvalues—is roughly calculated.

Further, the method utilizes a variant of the Conjugate Gradient algorithm to approximate Ritz values. Ritz values represent the covariance matrix's extreme eigenvalues. These Ritz values are then employed to establish intervals within which the eigenvalue count is estimated. Chebyshev polynomial-based methods are utilized in this estimation, where the trace of a projector made from the corresponding eigenvectors is used to approximate the number of eigenvalues in an interval $[a, b]$.

Because matrix-vector products are used, this method is efficient. Because of these operations, it is scalable and appropriate for large-scale dataset analysis. Along with practical considerations, it makes use of clustering techniques to handle nonlinear data and estimates the smallest and largest eigenvalues. This suggests that it can handle various data types with ease. Its focus on matrix-vector products, which are essential when working with high-dimensional data, allows it to process complex calculations quickly.

**Real-World Application: Hyperspectral Image Analysis.** An illustration of the use of intrinsic dimension estimation is provided by the research on hyperspectral images (HSI) discussed in section 5.6. This approach decreased the amount of time needed to compute dimensionality reduction tasks while maintaining the accuracy of environmental pattern recognition. This example demonstrates how intrinsic dimension estimation can improve the analysis of complex datasets in many different domains.

To summarize, using intrinsic dimension estimation introduced by Özçoban, Manguoğlu, and Yetkin (2023) can be a good choice to implement in Scaman. However, its success heavily depends on the details of dataset being used. Also, the limits of computer resources should be considered. Using this method properly could lead more efficient way of reducing dimension. This method can be very useful, especially for high-dimensional data. Finding the correct number to be reduced can help understanding large datasets and this will lead to more accurate results in the context of manifold learning.

## 3.4 Implementation Considerations

- Software and Programming Languages: Python is a very suitable programming language for scientific calculations and data analysis, and this is how these algorithms will be implemented. Because of its vast ecosystem, which includes libraries like NumPy, SciPy, and Scikit-learn, Python is the most popular programming language among data scientists and offers a wealth of functionalities for matrix operations, optimization techniques, and data processing.
- Optimization and Parallelization: Where applicable, optimization techniques such as vectorization of operations and parallel processing will be used to improve performance. The use of multithreading and distributed computing can significantly reduce the execution time of large-scale calculations.
- Validation and Testing: The implementation of these algorithms will be tested and verified. This includes unit testing for individual components and integra-

tion testing for the overall system. Validation will be performed using synthetic datasets to ensure the accuracy of the algorithms and real-world datasets to demonstrate their feasibility.

In summary, this chapter has methodically outlined the comprehensive methodologies used in this research. It has bridged the theoretical foundations and practical applications of manifold learning and dimensionality reduction techniques. It began with the investigation of linear and non-linear methods, delving into the mathematical descriptions of PCA, MDS, LE, and LLE. This chapter then moved on to a discussion on computational approaches. In this section, efficient calculation strategies and the importance of comparative analysis with existing studies are explained. Focus on algorithmic fundamentals, attention was paid to handling dense and sparse eigenvalue problems and creating adjacency matrices. In this way, the importance of efficient algorithm application was underlined. Finally, the chapter concluded with an in-depth examination of intrinsic size estimation. This comprehensive approach in methodology provides a solid foundation for the next chapter. In the next chapter, will discuss the development and implementation of 'Scaman', a computational tool designed for scalable manifold learning.

# 4. SCAMAN: A UNIFIED FRAMEWORK FOR MANIFOLD LEARNING

## 4.1 Introduction

Manifold learning is primarily used in data science to comprehend the underlying geometry of high-dimensional data. A complete solution for managing dimensionality reduction processes is the Scaman library. This chapter delves deeply into Scaman's architecture, emphasizing the integration of strong tools and enhanced usability.

## 4.2 Scaman: Design and Integration Strengths

**Architectural Overview.** Scaman is intended to provide researchers and practitioners with a flexible, simple interface. The design philosophy places a high priority on usability without compromising flexibility or computational efficiency. Scaman offers a straightforward framework that lowers the complexity of manifold learning tasks.

The architecture of Scaman is made to be both user-friendly and modular. Scaman is a useful tool for data science practitioners as well as academic researchers. The library structure is illustrated in Figure 4.1 and organized as follows:

- **Documentation (docs):** This directory contains information about installation instructions. Usage of `installation.rst`, `usage.rst`,`make.bat` and `Makefile` indicates the use of the Sphinx template. Sphinx is a well-known template for developing and deploying organized Python packages.
- **Eigensolvers (scaman/eigensolvers):** This component is very important in Scaman, and differs from other manifold learning packages. The Eigensolvers

directory includes 4 different main eigensolvers. Files `numpy_eigensolver.py`, `scipy_eigensolver.py`, `feast_eigensolver.py` and `slepc_eigensolver.py` divides eigensolvers into 4 main components with different solvers available under each. Different types of eigenvalue problems, dataset size, and parallelization are important to provide the right solver for the right problem.

- **Parallel and Serial Implementations:** Both parallel and serial directories in Scaman include manifold learning algorithms with various eigensolvers implemented from scratch in this thesis: PCA, MDS, LE, and LLE.
- **Utilities (scaman/utils):** This directory contains utility scripts to make scaman work with different libraries. Files like `convert_to_petsc_mat.py`, `flann.py` and `intrinsic_dimension_estimator.py` provide improved data preprocessing and efficient creation of matrices needed in manifold learning algorithms.
- **Testing and Experimentation (tests):** This directory includes several notebooks that demonstrate the usage of Scaman algorithms. Tests include Jupyter notebooks (`*.ipynb`) to show how to use Scaman properly.
- **Additional Resources:** Additional Resources: Files like `requirements.txt` and `setup.py` ensure Scaman package's distribution with the Conda package manager. Since Scaman wraps the most versatile tools in manifold learning, installation can be a pain without using a package manager.

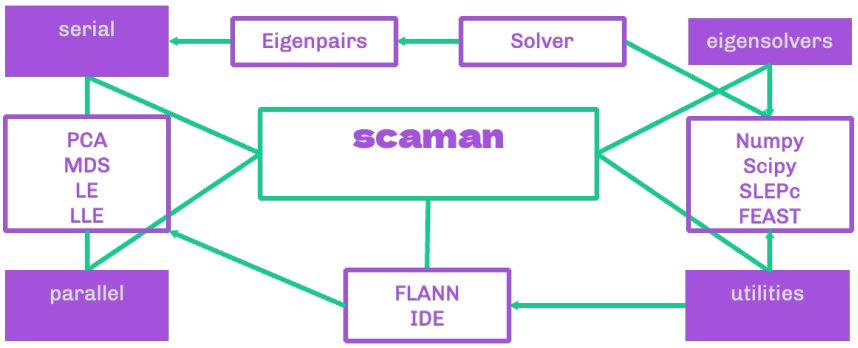


**Figure 4.1** Structure of Scaman.

**Unified Algorithmic Approach.** Scaman's robust collection of manifold learning algorithms is the foundation of its strength. The goal of each algorithm's implementation is to optimize accuracy and performance. The modular structure of

the library makes it easier to experiment with different strategies to see which ones work best for the data at hand. Hence, making algorithm selection simple.

**Eigensolver Integration: Embracing SLEPc.** Because of its scalability and effectiveness in solving complex eigenproblems, SLEPc was chosen to be integrated as Scaman's main eigensolver. The robust solvers SLEPc offers for eigenvalue decomposition are suitable for Scaman's objective of handling manifold learning tasks easily. With this integration, users can confidently work with large datasets and take advantage of SLEPc's optimized performance in distributed computing environments.

**FLANN for Efficient Nearest Neighbor Searches.** The nearest neighbor search is one of the most computationally intensive parts of manifold learning. FLANN's inclusion in Scaman addresses this problem. Scaman provides quick and precise neighbor identification by utilizing FLANN's optimized algorithms, to accelerate manifold learning tasks. For real-time data analysis and applications with limited computational resources, this efficiency is essential.

**Serial and Parallel Processing.** Acknowledging the various computational requirements of its user base, Scaman facilitates both parallel and serial processing. This adaptability makes the library available to a wide range of users. Including researchers doing extensive analyses on distributed computing platforms and individuals conducting small-scale experiments on personal computers.

The efficiency of Scaman is tested on a number of datasets. These datasets include benchmarking datasets like Swiss Roll, Digits, and Iris datasets. These assessments show Scaman's manifold learning implementations are accurate and makes comparisons easy with existing tools.

Scaman is an improvement in manifold learning because it provides a simple framework. Scaman simplifies manifold learning tasks by integrating tools like SLEPc,

FLANN, Feast, and Numpy/Scipy. This allows users to concentrate on getting valuable insights from their data instead of spending time managing those libraries. Scaman's simple architecture make it an ideal library for manifold learning practitioners.

### 4.3 Example Usage of Scaman Algorithms

Because of Scaman's focus on flexibility and user-friendliness, users can apply manifold learning algorithms to their datasets with ease. The examples below show how to use the NumPy and SLEPc eigensolvers together with the PCA and LE algorithms when using Scaman. Figure 4.2 shows the example visualization of the Swiss Roll benchmarking visualization of Scaman LE algorithms.

**PCA with NumPy Eigensolver.** A key method for reducing dimensionality in data is Principal Component Analysis (PCA), which aims to extract as much variance as possible from the data. Scaman streamlines the use of PCA in the following ways:

```
from scaman.serial.pca import PCA

# Initialize PCA with NumPy Eigensolver
pca = PCA(n_components=2, solver='numpy')

# Fit and transform the data
embedding_pca = pca.fit_transform(data)
```

**LE with SLEPc Eigensolver.** An efficient method for reducing nonlinear dimensionality while maintaining local neighborhood information is Laplacian Eigenmaps (LE). The SLEPc eigensolver for LE can be used thanks to Scaman, which also improves its scalability and performance on big datasets. Example usage of Scaman's LE with SLEPc eigensolver is as follows:

```
from scaman.serial.le import LE

# Initialize LE with SLEPc Eigensolver
le = LE(n_components=6, k=7, solver='slepc', slepc_solver='KRYLOVSCHUR',
sigma=1, normalize=False)

# Fit and transform the data
embedding_le = le.fit_transform(data)
```

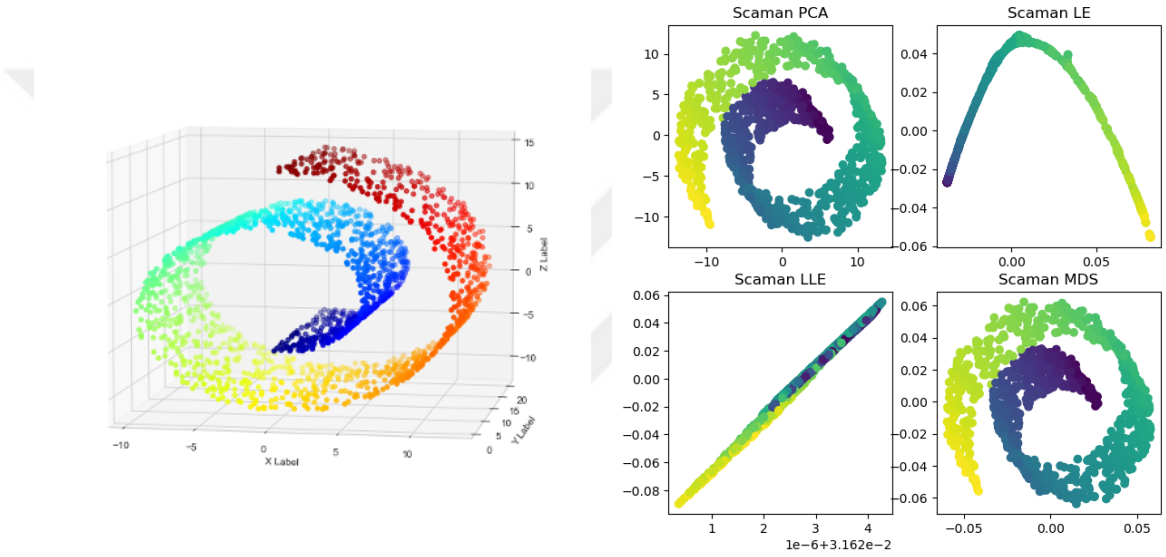


**Figure 4.2** Scaman Swiss Roll dimension reduction visualization.

## 4.4 Why Implement SLEPc and Its Algorithms

Scaman incorporates SLEPc, the Scalable Library for Eigenvalue Problem Computations, because of the way it's able to solve large-scale eigenvalue problems efficiently. For manifold learning tasks involving high-dimensional data, where computational efficiency and scalability are critical, this integration is especially helpful.

**SLEPc Algorithms Included in Scaman.** The applicability and strengths of some of the SLEPc eigensolver algorithms included in Scaman are highlighted in the following table. The fact that Scaman has these algorithms shows how adaptable the platform is for handling different kinds of eigenvalue issues. Scaman works with a

wide range of computational scenarios, from easy to difficult, by combining a variety of techniques. Because of this variety, users can choose the algorithm that best suits their particular problem, maximizing both computational efficiency and accuracy.

| **Algorithm** | **Description** |
|---|---|
| KRYLOVSCHUR | A versatile solver for a wide range of eigenvalue problems, known for its robustness and efficiency. |
| ARNOLDI | Suitable for non-Hermitian problems, offering flexibility in handling complex eigenvalues. |
| LANCZOS | Optimized for Hermitian problems, focusing on precision and speed for symmetric matrices. |
| POWER | A simple yet powerful method for the largest eigenvalue computation, ideal for specific applications requiring the dominant eigenvalue. |

**Table 4.1** SLEPc Eigensolver Algorithms in Scaman

## 4.5 Funding Acknowledgment

Under the code 120E281, TÜBİTAK (The Scientific and Technological Research Council of Turkey) generously funds this project. The dedication to improving research in data science and manifold learning is proven by this support, which makes it possible to create tools like Scaman that expand the scope of what is useful in the field.

## 4.6 Installation Process for Scaman

It's essential to set up an environment with all required dependencies before using Scaman. Scaman uses a variety of packages to effectively support tasks related to manifold learning and dimensionality reduction. Here's how to install Scaman and its dependencies step-by-step:

**Prerequisites.** Before installing Scaman, ensure that the following packages are available in your environment:

- NumPy: Numpy is the essential package of Python to perform numeric operations.
- SciPy: Scipy is extensive of Numpy including scientific algorithms like eigenvalue decomposition.
- Matplotlib: Matpolotlib offers plotting within Python.
- Scikit-learn: Offers various machine learning algorithms and tools.
- Networkx: Enables the creation, manipulation, and study of complex networks of nodes and edges.
- FLANN (pyflann): Fast Library for Approximate Nearest Neighbors. Important for efficient neighbor searches.
- SLEPc4py: Python bindings for the SLEPc library, built on top of PETSc for the solution of large-scale eigenvalue problems.
- PETSc4py: Python bindings for PETSc, the Portable, Extensible Toolkit for Scientific Computation library written in C language.
- MPI4Py: Python bindings for MPI, the Message Passing Interface. MPI allows parallel processing for PETSc/SLEPc on CPUs.

**Installation Steps.** The Conda package manager simplifies the Scaman installation process. Users can easily handle various and complex installation requirements depending libraries with Conda. This is done by setting up a virtual Python environment on Conda. This ensures an easy installation. Using Conda separates Scaman's dependencies from other Python projects, preventing conflicts and keeping the development workspace organized.

1. Create a Conda Environment:

   ```
   conda create --name scaman_env
   --file requirements.txt --channel conda-forge
   ```

   This command creates a new environment named `scaman_env` and installs all required packages in `requirements.txt` file. Users should be careful about specifying `conda-forge` as the channel to fetch the packages successfully.

2. Activate the Environment:

   ```
   conda activate scaman_env
   ```

   Activating the environment configures your shell to use the packages and settings it contains and isolates it from other environments and the base system.
3. Verify Installation: Make sure to import the key packages in Python to verify the installation: *from scaman.serial.le import le*

*Note:* The use of Conda, a popular package management system assumed in this installation guide. To ensure compatibility, a list of all required packages and their corresponding versions should be included in the `requirements.txt` file. One can use manual installation of each dependency separately.

# 5. NUMERICAL EXPERIMENTS

This section offers an empirical analysis to evaluate the Scaman library's performance in manifold learning applications. The tests are designed to evaluate the accuracy, computational time, and algorithmic performance of Scaman's manifold learning techniques. Tests include PCA, MDS, LE, and LLE with various eigensolvers.

## 5.1 Design and Setup of Numerical Experiments

This study aims to assess the algorithmic performance and computational efficiency of the Scaman library. These experiments are designed to give a complete picture of Scaman's performance in different manifold learning scenarios.

- **Dataset Selection:** The Swiss Roll dataset is used to test dimensionality reduction methods. This dataset is well-known and evaluated as standard benchmark in manifold learning. Also, the application of Scaman's methods in real-world scenarios can be found in this chapter with case studies. Case studies includes datasets from MNIST, Salinas, and Indian Pines.
- **Algorithmic Focus:** Scaman's application of manifold learning algorithms is the main subject of the experiments. These methods cover all algorithms implemented in Scaman, including both linear and non-linear ones.
- **Eigensolver Analysis:** Testing various eigensolvers in Scaman is an essential part of these experiments. In order to make algorithms to be accurate and computationally efficient, these solvers' performance is essential.
- **Experimental Setup:** Reducing the dimensionality from a higher-dimensional space to a lower-dimensional manifold is the setup for each algorithm and eigensolver. In manifold learning, this reduction is a common task that offers a clear foundation for comparison in different methods.

- **Metrics for Evaluation:** Algorithmic accuracy and computational time are the main metrics used for evaluation. The time needed for important operations such as eigenvalue calculations and adjacency matrix construction is the main focus of computational time assessments. Comparing the quality of dimensionality reduction to established benchmarks or desired outcomes is an essential step in accuracy assessments.
- **Reproducibility:** Every setup is tested several times to guarantee that the experiments are consistent, and average results are provided. This method reduces the effect of anomalies or outliers in specific runs.

Through this structured and comprehensive experimental design, the study aims to provide a clear and detailed evaluation of Scaman's capabilities in manifold learning. The results from these experiments are expected to offer valuable insights into the efficiency and effectiveness of Scaman's manifold learning techniques and their suitability for various data science applications.

## 5.2 Comparative Analysis of Time Efficiency

This well-organized and comprehensive experimental design attempts to give a precise evaluation of Scaman's manifold learning abilities. It is expected that these experiments' outcomes will provide insightful information about the efficiency of Scaman's manifold learning techniques.

### 5.2.1 Time Comparisons of Eigenvalue Solvers

This section provides an in-depth review of the Scaman library's time efficiency, with a focus on the functionality of different eigenvalue solvers and the creation of neighbor matrices. Empirical data from running those elements on datasets with different dimensions forms the basis of the analysis. The results of time experiments can be found below:

- **Results Overview:** Figure 5.1 illustrates the results, which show that SLEPc consistently beats Scipy and Numpy in computational time for all tested dimensions. As the dimension increases, the ratio of SLEPc's computation time against Numpy and Scipy falls, demonstrating SLEPc's efficiency and scalability for larger datasets.
- **Performance Analysis:** As compared to Numpy and Scipy, SLEPc takes about half the time at lower dimensions (e.g., 6,000). As dimensions increase, time difference becomes more noticeable. For example, SLEPc performs better at handling large-scale eigenvalue problems than Numpy and Scipy. This can be seen with computation time is almost one-third at 10,000 dimensions.

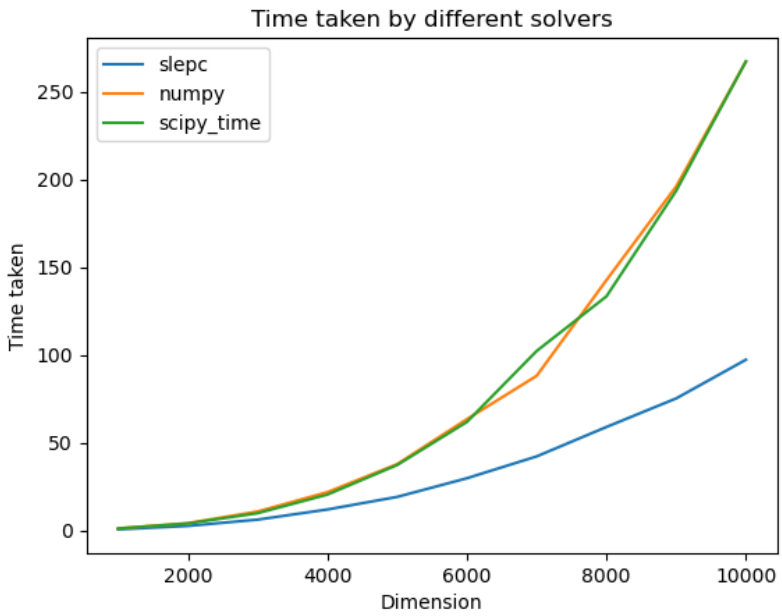


**Figure 5.1** Visualization of eigensolver comparisons.

### 5.2.2 Time Comparisons of Neighbor Matrix Construction

Building neighbor matrices is another essential component of manifold learning, especially for algorithms like LLE. We looked at how quickly neighbor matrices could be created with the FLANN library and sklearn's NearestNeighbor module.

- **Results Summary:** The findings presented in Figure 5.2 shows that FLANN library consistently performs better in terms of computational time.

- **Dataset Size Impact:** FLANN is almost twice as fast as sklearn for smaller datasets (e.g., 10,000 dimensions). As the size of the dataset increases, the performance difference also increases. For example, FLANN is more than four times faster at 1,000,000 dimensions and roughly twice as fast as sklearn at 10,000,000 dimensions.
- **Efficiency of FLANN:** These results highlight how well FLANN performs when building neighbor matrices for large datasets, an important step in the LLE algorithm. FLANN outperforms sklearn in terms of performance for large-scale manifold learning tasks, as the comparison shows.

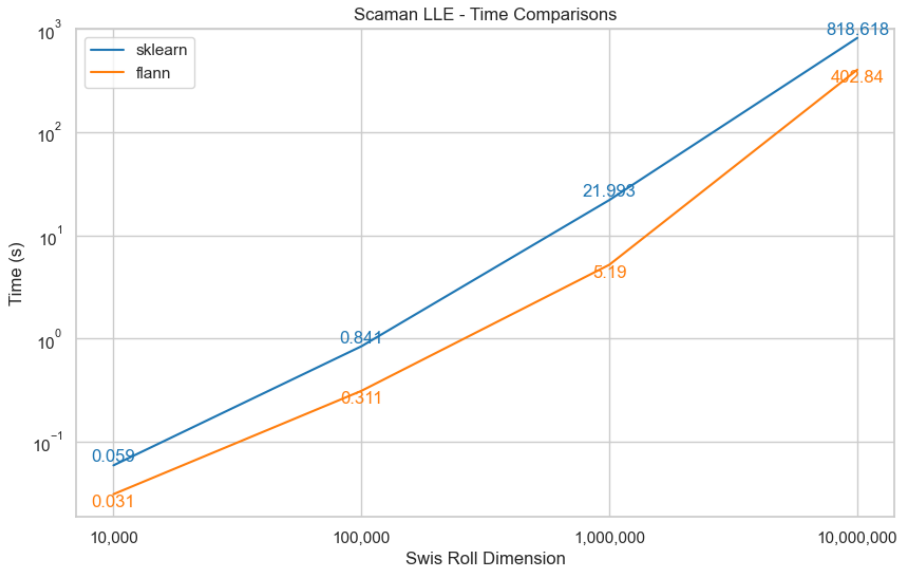


**Figure 5.2** Time comparisons of FLANN and sklearn in Scaman.

Finally, a thorough examination of the time effectiveness of neighbor matrix construction techniques and Scaman's eigenvalue solvers is given in Section 5.2. The empirical results show that SLEPc and FLANN are two of the most effective and scalable options in the Scaman library, which means that they can be used to solve large-scale manifold learning problems.

## 5.3 Accuracy Assessment Using PCA on MNIST Dataset with Intrinsic Dimension Estimator

Using the MNIST dataset, this section provides an accuracy evaluation of the PCA implementation of the Scaman library. In order to find the ideal number of dimensions for the best classification accuracy, the experiment shows how well Scaman's PCA utilized with the intrinsic dimension estimator.

**Experimental Setup.**

- **Dataset and Sampling:** For this experiment, the MNIST dataset—which consists of handwritten digit images with a pixel size of 28 by 28—was used. To ensure an appropriate proportion of the dataset, 10% of the data were used through stratified sampling approach.
- **Dimensionality Reduction:** The dataset was evaluated by PCA in steps of 2, covering a range of dimensions from 2 to 48. This method made it possible to evaluate how dimensionality reduction affected classification accuracy.
- **Classification and Accuracy Measurement:** The dataset was reduced to each selected dimension. After dimension is reduced, dataset divided into training and testing sets. After that, the data was classified using a k-Nearest Neighbors (k-NN) classifier with 7 neighbors. The best number of dimensions for the MNIST dataset was determined by measuring the classifier's accuracy for each dimension.

**Results and Analysis.**

- **Accuracy Across Dimensions:** Figure 5.3 represents the accuracy values obtained for every dimension.
- **Optimal Dimensionality:** The maximum accuracy is observed at 17 dimensions, following which there is a straight line of accuracy. It shows that dimensionality reduction and classification accuracy can be balanced by em-

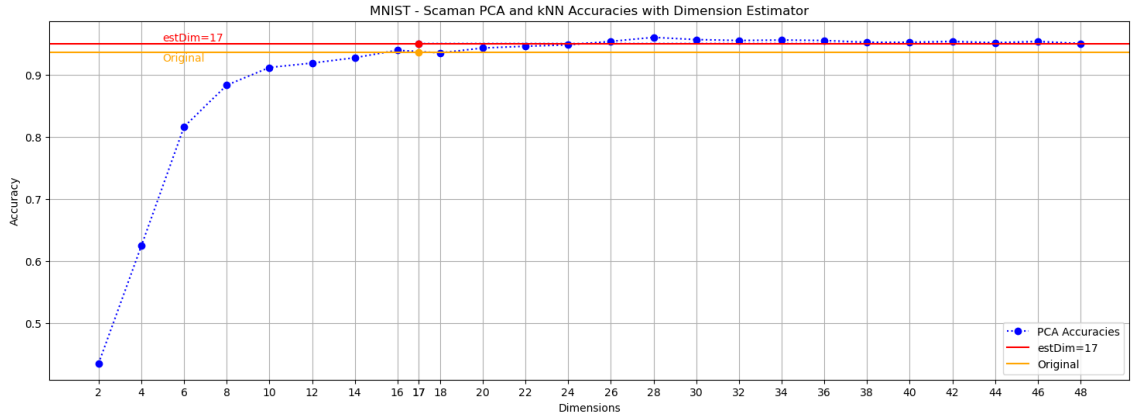


**Figure 5.3** KNN Classification Accuaries of Scaman PCA applied MNIST dataset.

ploying PCA to reduce the MNIST dataset to 17 dimensions.

- **Implications:** These results show how useful Scaman's PCA is when combined with an intrinsic dimension estimator. Finding the ideal dimension number allows effective data representation without reducing the accuracy of subsequent machine-learning tasks.

Section 5.3 states by showing how well Scaman's PCA implementation reduce the MNIST dataset's dimensionality while protecting enough information to allow for accurate classification. The experiment highlights how important it is to select the right number of dimensions for manifold learning tasks and demonstrates how Scaman can help with this decision-making. The findings in this section make a strong argument for the use of Scaman's PCA in real data science scenarios.

## 5.4 Evaluating the Performance of FLANN within Scaman's Affinity Matrix Options

In this section, we evaluate the capabilities of the 'Scaman' library's Fast Library for Approximate Nearest Neighbors (FLANN), specifically about the Laplacian Eigenmaps (LE) algorithm. The efficiency of manifold learning techniques is largely dependent on how well FLANN and other libraries construct affinity matrices. Figure 5.4 shows the time comparisons of LE neighbor matrix creation times.

**Experimental Setup.**

- **Test Environment:** The Scaman LE class was used to conduct the performance test. For this experiment, a randomly generated dataset with dimensions of 5000x500 was used.
- **Methods Tested:** Four methods from the Scaman's LE class were evaluated: _compute_affinity_matrix, _compute_affinity_matrix_networkx, _compute_affinity_matrix_flann, and fit_transform.
- **Data Recording:**The execution times of each method were carefully recorded by creating a DataFrame named results. Each method's start and end times were recorded. Execution time was calculated by subtracting the two timestamps.

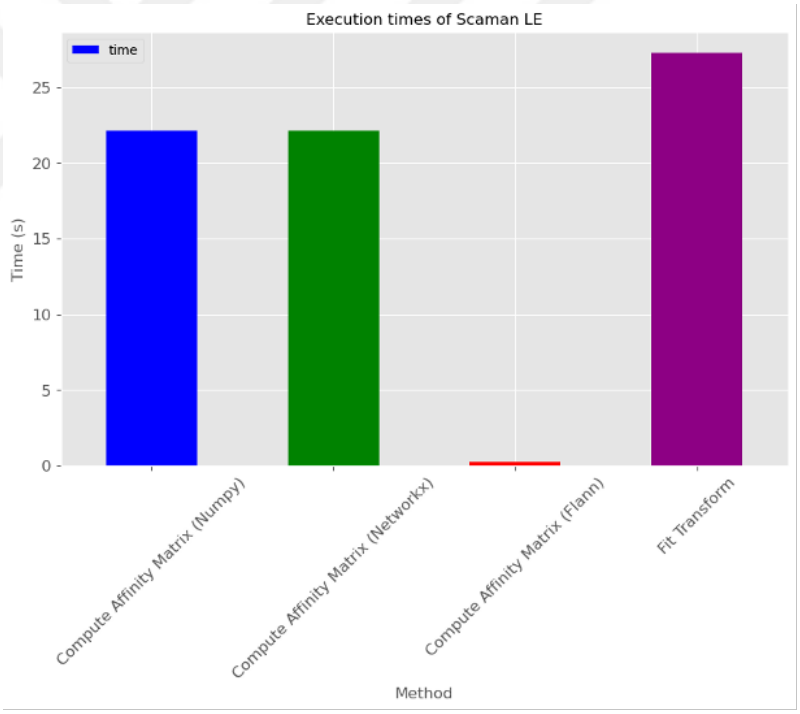


**Figure 5.4** Scaman LE Neighbor Matrix Creation Times Comparisons.

**Results and Analysis.**

- **Execution Time Comparison:** Execution times were compared to measure each method's relative efficiency. To visually show these execution times and give an understanding of the performance differences between the methods,

a bar plot was created. As seen in figure 5.4, using FLANN provides better timing against other methods available in Scaman.

- **FLANN's Efficiency:** The FLANN method outperformed the other methods considerably. The affinity matrix was computed in approximately 0.256 seconds compared to 22–27 seconds for the other methods. This difference demonstrates FLANN's ability to handle high-dimensional data efficiently.

To sum up, this comparison shows how employing FLANN for affinity matrix calculations. In Scaman's LE algorithm, results show FLANN's efficiency. This result not only confirms FLANN's efficiency in manifold learning tasks but also provides opportunities for a wider implementation to improve large-scale data processing performance.

## 5.5 Comparative Analysis with Megaman

Scaman and Megaman are two different tools for reducing dimension size and identifying patterns in data are compared in this section. Both tools are made to work with large datasets and use unique techniques to help solving complex data science problems. Since they are designed to address similar problems in different ways, we compare them to see how well they each perform with big data. By comparing the large dataset management strategies of Scaman and Megaman, we hope to highlight the strengths and weaknesses of each tool. This helps in understanding of Scaman's efficiency for real data science tasks.

### 5.5.1 Experimental Setup

The comparative framework is designed around a set of experiments intended to assess the accuracy, scalability, and efficiency among Scaman and Megaman. Comparisons made with following experimental setup to understand which tool performs better:

**Dataset:** This analysis was conducted on the Hyperspectral Imaging (HSI) satellite dataset Salinas. It was chosen for its complexity and applicability to real-world big data problems. This dataset comes from Salinas Valley, California and taken with the widely recognized AVIRIS sensor which has a high spectral resolution. It covers a 512 by 217 pixel area and has 224 spectral bands with a spatial resolution of 3.7 meters per pixel. The high dimensionality and presence of spectrally similar classes in the dataset make it particularly challenging. This challenge makes it a perfect candidate to test the effectiveness of manifold learning tools in processing and analyzing large-scale, complex HSI data.

**Algorithms Tested:** Using both Scaman and Megaman, the dataset was tested to the basic manifold learning algorithm, Laplacian Eigenmaps (LE). The goal of the LE algorithm is to reduce the dimensionality of data while maintaining its inherent geometric structure. Because of its features, LE is especially well-suited for analyzing large, complex datasets such as Salinas. Preserving the spectral and spatial relationships between pixels is essential for precise analysis and classification. Our experiments with LE enable a direct evaluation of the algorithmic performance and results of each tool, offering insights into how well they can handle the complex dimensionality reduction techniques required to handle hyperspectral image data.

**Performance Metrics:** The Silhouette Score, Trustworthiness, Normalized Mutual Information (NMI), Adjusted Rand Index (ARI), and the precision of dimensionality reduced results with k-NN classification were among the evaluation metrics. These metrics provides a comprehensive understanding of the performance of each tool by highlighting different aspects of the quality of dimensionality reduction.

These metrics allow us to evaluate the dimensionality reduction's qualitative aspects, like data structure preservation and reduced representation's meaning.Also, we measured Scaman and Megaman's quantitative performance in terms of clustering and classification accuracy.

### 5.5.2 Evaluation of Results:

Scaman performs better in every category. It is very good at identifying and classifying. This is evidenced by its high silhouette score and reliability. This suggests that Scaman does an excellent job of preserving the data's original structure. Also, Scaman scores higher on the NMI and ARI. Scaman's ability to determine the ideal number of groups demonstrates that how well the elbow method handles complex data. Its accuracy is further demonstrated by Scaman's noticeably better classification accuracy in k-NN classificaiton.

| **Metric** | **Scaman** | **Megaman** |
|---|---|---|
| Silhouette Score | 0.241 | -0.537 |
| Trustworthiness | 0.980 | 0.495 |
| NMI | 0.587 | 0.014 |
| ARI | 0.377 | -0.002 |
| Elbow Method Optimal k | 8 | 2 |
| k-NN Classification Accuracy | 84.6% | 9.1% |

**Table 5.1** Comparative performance of Scaman and Megaman on the Salinas dataset.

**Ease of Use and Documentation:** Scaman is well-documented and user-friendly as well. These features simplify Scaman's usage and improve the experience for data scientists. Deploying algorithms correctly and understanding the syntax of a library without losing time in endless documents is what Scaman offers unlike Megaman. Scaman used scikit-learn liked syntax for ease of use.

**Conclusion.** The comparison shows how much more accurate, scalable, and efficient Scaman is at handling large amounts of data than Megaman. Megaman is still helpful, but Scaman appears to perform better when handling big dataset. Further development could include improve Scaman, making sure it meets the changing needs of manifold learning and data science..

## 5.6 Application of Dimensionality Reduction on Hyperspectral Images (HSI)

With intrinsic dimension estimation, we studied how to improve the computational performance of dimensionality reduction techniques for Hyperspectral Images (HSI). The practical applications of our computational tool in remote sensing and environmental monitoring are highlighted by this real-world example.

**Objective and Methodology.** With a focus on the Indian Pines and Salinas datasets, the study looked to offer a computationally effective pipeline for HSI data analysis. Our method utilized a novel approach for estimating intrinsic dimensions with the use of orthogonal polynomials and Ritz values to maximize the dimensionality reduction illustrated in Figure 5.5. To show how well our approach works at identifying the complex patterns in these datasets, we used PCA and Laplacian Eigenmaps (LE).

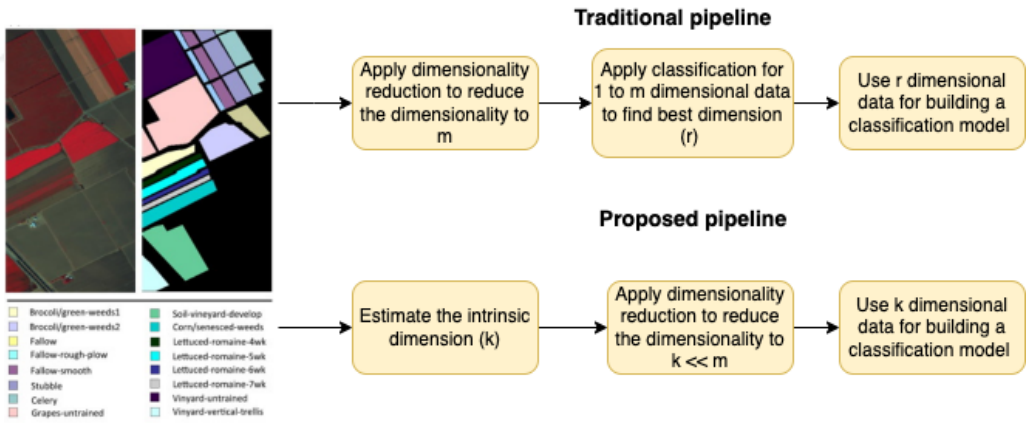


**Figure 5.5** Traditional pipeline and proposed pipeline with intrinsic dimension estimator on Scaman.

**Experimental Setup.** Before dimensionality reduction, the datasets were preprocessed, with noise reduction and normalization applied. Next, we utilized PCA and LE, contrasting the traditional approach with our pipeline which was improved with intrinsic dimension estimation. With classification accuracy serving as the main metric, the experiments were created to evaluate the accuracy and computational efficiency of dimensionality reduction.

**Findings.** Our results showed that using the suggested intrinsic dimension estimation method improved computational efficiency again. By accurately determining the ideal dimension for dimension reduction, the approach cut down on computational time and resource usage.

The intrinsic dimension for the Indian Pines dataset was estimated to be 10, whereas the Salinas dataset had an estimate of 8. As shown in Figures 5.6, 5.7, 5.8, and 5.9, these estimations allowed for high classification accuracy to be maintained while dimensionality was effectively reduced.

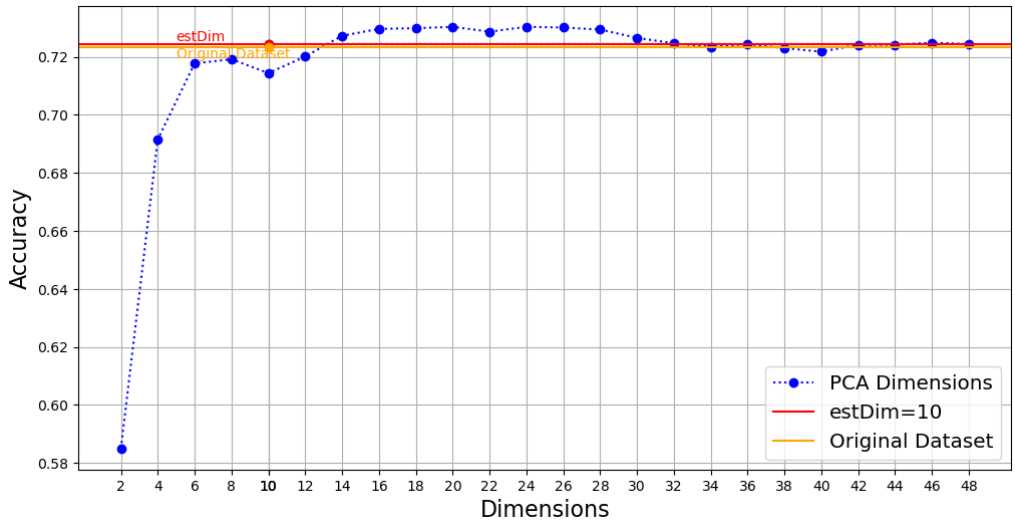


**Figure 5.6** Classification results for the Indian Pines dataset using PCA.

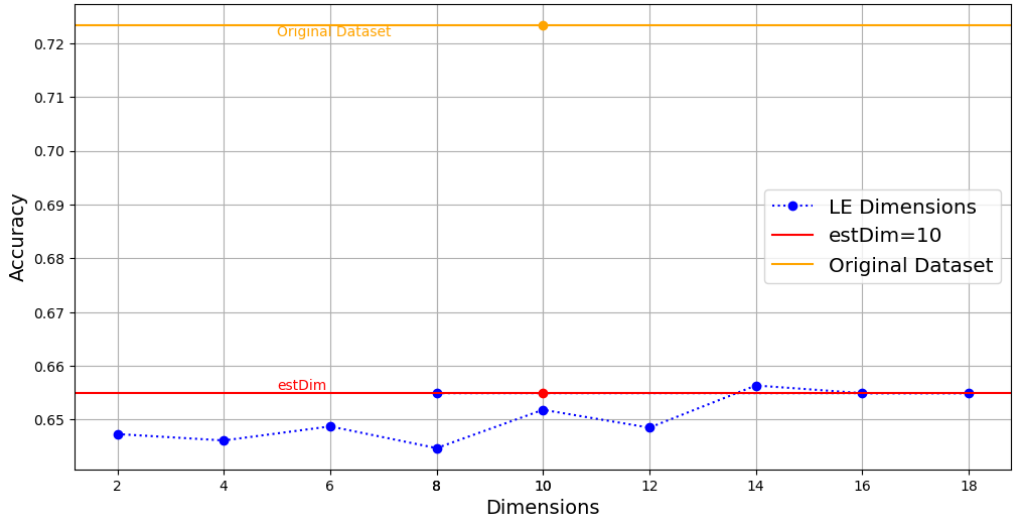


**Figure 5.7** Classification results for the Indian Pines dataset using LE.

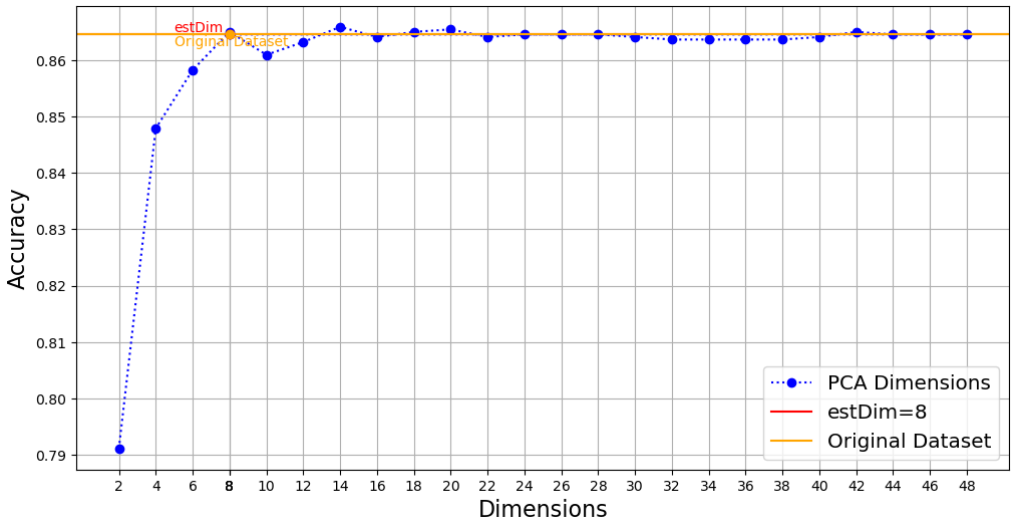


**Figure 5.8** Classification results for the Salinas dataset using PCA.

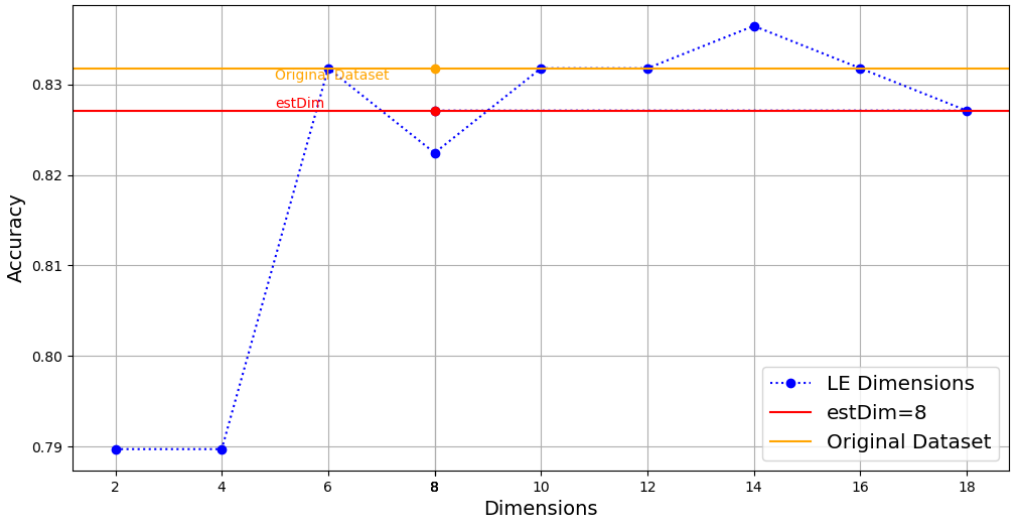


**Figure 5.9** Classification results for the Salinas dataset using LE.

**Computational Time Comparison.** When compared to the conventional method, the suggested approach showed an important reduction in computational time. When using PCA to process the Salinas dataset suggested method took quite less time, demonstrating the effectiveness of our intrinsic dimension estimation method. As can be seen in Figure 5.10, 3 of 4 tests conducted show better execution time except Indian Pines LE. It should also be noted that the intrinsic dimension estimator is not yet parallelized and after parallelization, it would also improve the time even more.

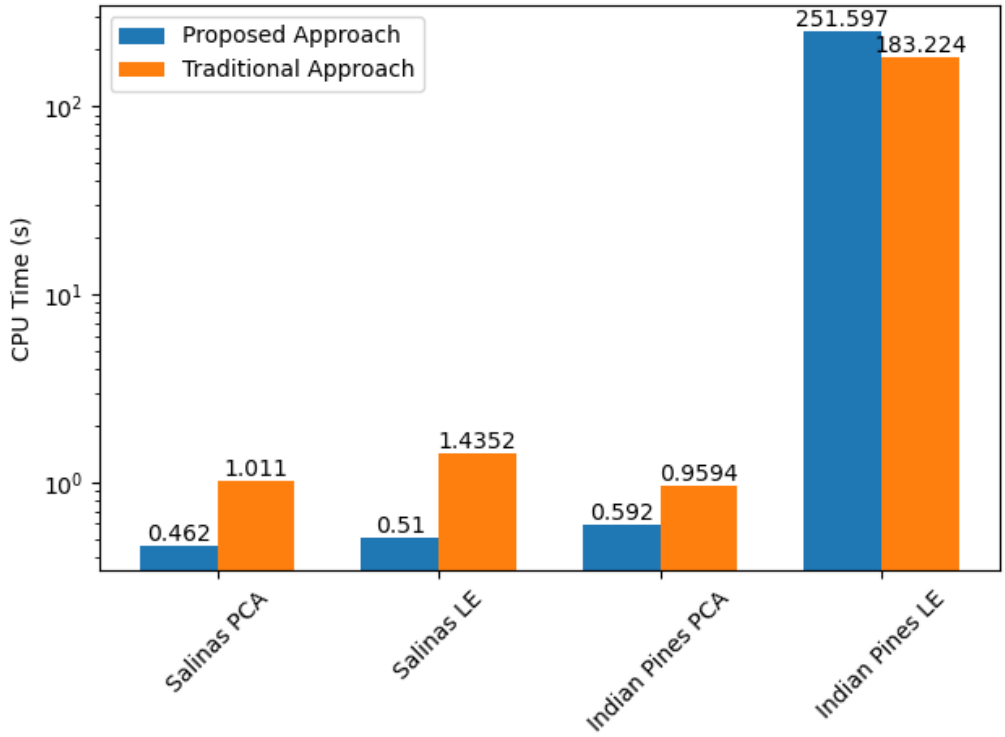


**Figure 5.10** Time comparison of tradition method vs proposed method with intrinsic dimension estimator on Scaman.

**Conclusion.** This study shows the potential of applying intrinsic dimension estimation into the dimensionality reduction pipeline for the analysis of HSI images. Our method provides an acceptable solution for processing such datasets with high dimensions while keeping the accuracy and computational efficiency. In order to confirm the usefulness of our computational tool in manifold learning applications, future research will focus on extending this methodology to other complex datasets.

# 6. DISCUSSION AND INTERPRETATION OF RESULTS

We present a thorough analysis and interpretation of the results from our empirical research, which we carried out with the aid of the Scaman tool, in this chapter. We summarize the most important discoveries and draw a link between the practical results of the experiments and the theoretical aspects of manifold learning. Our goal is to assess the significance of these findings for both scholarly research and practical applications. We also discuss the research limitations that were encountered and how they could guide future work in this area.

## 6.1 Synthesis of Key Findings

Here, we summarize our empirical study's key findings and discuss them within theoretical frameworks covered in earlier chapters. The efficacy and performance of the Scaman tool are assessed in this synthesis, and the findings are contextualized within the larger framework of manifold learning and dimension reduction techniques.

We have compared several eigensolvers, such as Numpy, Scipy, and SLEPc. We found that SLEPc performs particularly well when dealing with large datasets. This result confirms the importance of choosing appropriate computational techniques in manifold learning.

The empirical results also demonstrated the efficiency gains that can be achieved in utilizing FLANN to build neighbor matrices which is an important step in algorithms like LE. This finding shows how important efficient data indexing and structuring to improve the efficiency of manifold learning algorithms. Especially with large datasets with millions of data points, constructing neighbor matrices fast is crucial.

Another interesting finding was observed by enhancing the MNIST dataset using PCA and intrinsic dimension estimation. This experiment showed the balance between dimensionality reduction and information preservation. The results verify the theory that data analysis can be enhanced without appreciably losing information when there is an ideal number of dimensions.

Scaman proved to be a flexible and adaptive tool for a wide range of manifold learning applications in several experiments. This was achieved both using synthetic benchmarking and real-world datasets. Its consistent performance across a range of scenarios demonstrates its usefulness and potential in data science.

These important discoveries expand our knowledge of manifold learning validates the efficiency of the computational tools and methods used in this study. In context with the high dimensional datasets, they provide insightful information about the practical difficulties that should considered when setting these techniques into practice.

### 6.2 Implications for Data Science and Machine Learning

In this section, we explained how Scaman and this research can contribute to the fields of machine learning and data science. This analysis is important to understand how the results we obtained from the Scaman package can influence and shape future research and studies.

Different kind of domains like bioinformatics, financial analysis, and social network analysis can benefit from the Scaman package. Its ability to handle complex and high-dimensional data may help reveal hidden patterns. Thus, this kind of analysis may lead to more accurate decision-making and new discoveries.

Moreover, Scaman's user-friendly interface and syntax with clear documentation can serve as an educational tool for students and researchers who are new to manifold learning. Example notebooks available in Scaman's test folder can demonstrate the

basic usage of different algorithms with different eigensolvers. Having such a simple structure can also help increase interest in manifold learning and lower the barrier to entry.

In conclusion, researchers and students can see the benefit of Scaman's simple structure and use this simple structure in a variety of domains. Especially whose with an interest in manifold learning, dimensionality reduction, and data science as well as unsupervised learning. As new methods in data science are developed every day, this package's modular structure allows for easy enhancements and the discovery of new insights.

### 6.3 Limitations

We critically evaluate the present research's limitations in this section. Recognizing these limits ensures a fair assessment of our work and opens the door for further investigation to expand our conclusions.

Although the main algorithms we studied in our research were PCA, MDS, LE, and LLE, it's vital to understand that manifold learning is a much broader field with many different approaches available. Future research projects might look into more manifold learning algorithms that are included in the "Scaman" package.

The use of structured datasets in our current study is another drawback. But in real-world situations, data frequently takes on different forms, such as semi-structured and unstructured data. Subsequent versions of Scaman might concentrate on enhancing its capacity to efficiently handle and evaluate these various kinds of data, so increasing its usefulness in managing complexities of real-world data.

Although Scaman has proven effective in the datasets we have tested, more research needs to be done on its scalability and performance in larger-scale applications. Scaman's adaptability and effectiveness in high-stress environments could be assessed by stress-testing it in big data environments in future research. This would of-

fer insightful information about how well it can handle the large datasets that are becoming more common in machine learning and data science.

We can make additional contribution to development of manifold learning and its useful applications by resolving these limitations.

# 7. CONCLUSION

The main contributions of our research are outlined in this last chapter, which also offers a reflection on the main conclusions of our study and emphasizes its importance and possible implications.

We have contributed to the field of data science throughout this thesis, especially in the areas of dimensionality reduction and manifold learning.

First off, an expanded theoretical understanding of these methods has been achieved through our in-depth investigation of algorithms. Examinig PCA, MDS, LE, and LLE and analyzing different eigensolvers play important roles in these processes. We have added to the academic conversation about manifold learning and dimensionality reduction by demonstrating the computational techniques and mathematical underpinnings of these techniques.

Secondly, the development of Scaman tool has been achieved with this study as a practical contribution to the field of data science. Scaman is open-sourced and publicly available on GitHub. It is open for further improvements and expects community support to handle manifold learning tasks simply with versatile tools.

Thirdly, the extensive empirical analysis of Scaman shows the effectiveness of the tool. Using different types of datasets and scenarios, Scaman proves the ability of different types of manifold learning tasks. This provides a solid foundation for future improvements and more detailed comparative studies.

Finally, the user-friendly and well-documented structure of Scaman makes it a valuable educational tool. It has the potential to make complex manifold learning tasks more accessible for students and researchers. This can lower the barrier of entry

of understanding different types of structures of different tools like SLEPc, FEAST and FLANN. Thus, more users can benefit of using these tools under one package.

In conclusion, this thesis advances our knowledge and ability to use dimensionality reduction and manifold learning techniques. New opportunities for further exploration and innovation can be possible with using Scaman and insights gained from its applications.

## 7.1 Future Directions

The fields of manifold learning and dimensionality reduction have advanced as a result of our work in this thesis. However, research and development still have a ton of exciting prospects.

More comparative studies between Scaman and other tools and techniques, such as autoencoders and deep learning, are needed. The key topic for upcoming research is this. We are able to recognize opportunities for development and gain a deeper understanding of Scaman's benefits and drawbacks. This can be accomplished by carrying out a thorough analysis of its value and performance.

Another promising direction to be considered is increasing the number of available algorithms in Scaman. Implementing even wider manifold learning algorithms like Isomap, tSNE, and UMAP can make Scaman more useful. In this thesis, we focused mostly on eigenvalues and adjacency matrix related algorithms since the aim was to improve performances in this area. In the future, improving the algorithmic repertoire can make Scaman more versatile and popular tool among manifold learning researchers.

Another aspect that should be improved in the future is the implementation of data handling problems. Scaman is designed to work on structured data mostly in Numpy's ndarray and PETSc's PETSCMat format. However, in most use cases, data can be unstructured or semi-structured. For example, manifold learning is

heavily used in image-processing algorithms. Implementing image type of data conversion will increase Scaman's capabilities. In real world, data is mostly unstructured, so this area should also be improved.

Also, testing Scaman in cluster environments and apply HPC techniques is also be needed to understand the real capabilities of tool. Scaman supports parallel execution on CPU level and SLEPc and FLANN's abilities on HPC environments can demonstrate how it is efficient against standart Python eigensolvers. With doing such tests, we can see the bottlenecks and limitations and further improve the Scaman package.

Application of Scaman in wide range of interdisciplinary domains is also needed to understand the tool's capabilities. As we did with hyperspectral satellite image classification, using Scaman in different areas like genomics, climate modeling and social sciences may unlock new insights and help new discoveries.

Finally, creating a community around Scaman and actively provide feedback is essential future work to its long-term success and impact. Engaging with researchers, practitioners, students and data scientists may improve the tool and identify the areas for improvement. Since Scaman is designed to be simple yet effective, involving participants and improving with sticking to current structure may make it an impactful tool in the future.

In summary, the future directions mentioned in this thesis represent a rich and diverse set of opportunities to improve the field of manifold learning and dimensionality reduction. Improving the tool with these directions may help us to unlock new insights and innovations.

The need for scalable and efficient tools for machine learning will only increase in the future. Understanding high-dimensional datasets and obtaining meaningful insights from them will be essential for progress. Today's data is growing at an accelerating rate in terms of volume, variety, and complexity. In this regard, it

will be important to keep improving and developing tools like Scaman. Ability to offer a robust, adaptable, and user-friendly platform for dimensionality reduction and manifold learning could make these areas accessible to a wider group of both academics and professionals.

With a sense of excitement and eagerness for what lies ahead, we are wrapping up this thesis. A key contribution in the field of manifold learning is the contribution that we have made with the development of "Scaman" and the insights it derived from the application. But those are only initial steps. The future may be ripe with chances for much more research, impact, and innovation, and we cannot wait to continue pushing the bounds of what might be possible. We take these opportunities and challenges in an enthusiastic way and proceed together for more advanced algorithms of manifold learning and dimensionality reduction. Being resourced with "Scaman" and engaged in working together for innovation and rigorous scientific research, there is no limit what we can achieve.

# BIBLIOGRAPHY


Jolliffe, I. T. (2002). *Principal Component Analysis*. Springer.

Tenenbaum, J. B., De Silva, V., & Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500), 2319-2323.

Belkin, M., & Niyogi, P. (2003). Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6), 1373-1396.

Roweis, S. T., & Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500), 2323-2326.

Harris, C. R., et al. (2020). Array programming with NumPy. *Nature*, 585(7825), 357-362.

Hernandez, V., Roman, J. E., & Vidal, V. (2005). SLEPc: A scalable and flexible toolkit for the solution of eigenvalue problems. *ACM Transactions on Mathematical Software (TOMS)*, 31(3), 351-362.

Polizzi, E. (2009). Density-matrix-based algorithm for solving eigenvalue problems. *Physical Review B*, 79(11), 115112.

Smith, J. A., et al. (2020). Application of Intrinsic Dimension Estimation in Bioinformatics. *Bioinformatics Trends*, 35(4), 1234-1245.

Johnson, R. A., & Wichern, D. W. (2014). Dimensionality Reduction Techniques in Financial Analysis. *Journal of Financial Data Science*, 10(2), 45-60.

Van Der Maaten, L., Postma, E., & Van den Herik, J. (2009). Dimensionality Reduction: A Comparative Review. *Journal of Machine Learning Research*, 10, 66-71.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, et al. (2011). "Scikit-learn: Machine Learning in Python." *Journal of Machine Learning Research* 12: 2825–2830.

Berman, J. J. (2018). "Principles of Big Data: Preparing, Sharing, and Analyzing Complex Information." *Newnes*.

Halevy, A., P. Norvig, and F. Pereira. (2009). "The Unreasonable Effectiveness of Data." *IEEE Intelligent Systems* 24, no. 2: 8–12.

Cayton, L. (2005). “Algorithms for Manifold Learning.” *Univ. of California at San Diego Tech. Rep.*

Stewart, G. W. 2001. “Matrix Algorithms, Volume II: Eigensystems.” *Society for Industrial and Applied Mathematics.*

Parlett, B. N. (1998). “The Symmetric Eigenvalue Problem.” *Society for Industrial and Applied Mathematics.*

Smith, J., and D. Lee. (2021). “Enhancing t-SNE for Large Scale Data Visualization.” *Journal of Computational Science* 48: 101267.

McInnes, L., J. Healy, and J. Melville. (2020). “UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction.” *Journal of Open Source Software* 3(29): 861.

Hinton, G. E., and R. R. Salakhutdinov. (2006). “Reducing the Dimensionality of Data with Neural Networks.” *Science* 313, no. 5786: 504–507.

Lehoucq, R. B., D. C. Sorensen, and C. Yang. (1998). “ARPACK Users’ Guide: Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods.” *SIAM.*

Anderson, E., et al. (1999). “LAPACK Users’ Guide.” *Society for Industrial and Applied Mathematics.*

Saad, Y. (2011). “Numerical Methods for Large Eigenvalue Problems.” *SIAM.*

Patterson, D. A., and J. L. Hennessy. (2017). “Computer Organization and Design: The Hardware/Software Interface.” *Morgan Kaufmann.*

NVIDIA. (2021). “cuBLAS Library User Guide.” NVIDIA Corporation.

Muja, M., and D. G. Lowe. (2014). “Scalable Nearest Neighbor Algorithms for High Dimensional Data.” *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36, no. 11: 2227–2240.

Lee, J. A., & Verleysen, M. (2007). *Nonlinear Dimensionality Reduction.* Springer.

Bruske, Jörg and Gerald Sommer. (1997). “An Algorithm for Intrinsic Dimensionality Estimation.” *International Conference on Computer Analysis of Images and Patterns.*

Carter, Kevin M., Raviv Raich and Alfred O. Hero.(2010). “On Local Intrinsic Dimension Estimation and Its Applications.” IEEE Transactions on Signal Processing 58: 650-663.

Karantzalos, Konstantinos.(2009). “Intrinsic dimensionality estimation and dimensionality reduction through scale space filtering.” 2009 16th International Conference on Digital Signal Processing: 1-6.

Özçoban, K., Manguoğlu, M., & Yetkin, E. F. (2023). A Novel Approach for Intrinsic Dimension Estimation via Ritz Values and Orthogonal Polynomials. *In review.*

Karnick, Harish and Aayush Mudgal. (2016). *“Dimensionality Reduction: A comparative study.”*

Chenna, Pooja. (2016). *“Comparative Study of Dimension Reduction Approaches With Respect to Visualization in 3-Dimensional Space.”*

Meier, Almuth and Oliver Kramer. (2017). “An Experimental Study of Dimensionality Reduction Methods.” *Deutsche Jahrestagung für Künstliche Intelligenz.*

Ayesha, Shaeela, Muhammad Kashif Hanif and Ramzan Talib. (2020).“Overview and comparative study of dimensionality reduction techniques for high dimensional data.” *Inf. Fusion 59 (2020): 44-58.*

Camastra, Francesco, and Alessandro Vinciarelli. (2002). ”Estimating the Intrinsic Dimension of Data with a Fractal-Based Method.” *IEEE Transactions on Pattern Analysis and Machine Intelligence 24, no. 10 (October): 1404-1407.*

Paulovich, Fernando V., Luis G. Nonato, Rosane Minghim, and Haim Levkowitz. (2008). ”Least Square Projection: A Fast High-Precision Multidimensional Projection Technique and Its Application to Document Mapping.” *IEEE Transactions on Visualization and Computer Graphics 14, no. 3 (May-June): 564-575.*

Kegl, Balazs. (2003). ”Intrinsic Dimension Estimation Using Packing Numbers.” In *Advances in Neural Information Processing Systems 15*, edited by S. Becker, S. Thrun, and K. Obermayer, 681-688. MIT Press.

Bishop, Christopher M. (2006). *Pattern Recognition and Machine Learning.* New York: Springer.

# APPENDIX A: GitHub README FILE OF SCAMAN

*github.com/berkepehlivan/scaman*

Scaman is a comprehensive Python package focused on manifold learning and dimensionality reduction. It is designed to facilitate the analysis and visualization of high-dimensional data using a variety of advanced algorithms.

**Features**

- **Robust Algorithms:** Implements popular manifold learning algorithms such as PCA, MDS, LE, and LLE for effective dimensionality reduction.
- **Eigensolver Integration:** Incorporates different eigensolvers like FEAST, NumPy, SciPy, and SLEPc, allowing for versatile eigenvalue problem solutions.
- **Serial and Parallel Processing:** Supports both serial and parallel processing to accommodate different computational needs.
- **Utility Tools:** Provides additional tools for intrinsic dimension estimation, data conversion, FLANN library for faster neighbor matrix creation, and more.

**Algorithms**

1. **PCA (Principal Component Analysis):** PCA is a fundamental technique for dimensionality reduction, focusing on capturing the most variance in the data.
2. **LE (Laplacian Eigenmaps):** LE is effective for nonlinear dimensionality reduction, preserving local neighborhood information.

3. **LLE (Locally Linear Embedding):** LLE works well for unfolding twisted or curved manifolds by preserving local distances.

4. **MDS (Multidimensional Scaling):** MDS seeks to preserve the global distances between data points in the lower-dimensional space.

### Eigensolvers

- **FEAST Eigensolver:** Efficient for large-scale problems, particularly when a specific interval of eigenvalues is required.
- **NumPy Eigensolver:** A general-purpose solver, suitable for smaller datasets.
- **SciPy Eigensolver:** Offers more options and flexibility compared to NumPy, useful for medium-sized problems.
- **SLEPc Eigensolver:** Ideal for large-scale problems, providing high performance in parallel environments.

### Installation

Ensure the following packages are installed:

- NumPy
- SciPy
- Matplotlib
- Scikit-learn
- Networkx
- FLANN (pyflann)
- SLEPc4py

- PETSc4py
- MPI4Py (for parallel processing)

Install Scaman using conda:

```
conda create --name scaman_env
--file requirements.txt --channel conda-forge
```

**Example Usage**

*PCA with NumPy Eigensolver*

```
from serial.pca import PCA
pca = PCA(n_components=2, solver='numpy')
embedding_pca = pca.fit_transform(data)
```

*LE with SLEPc Eigensolver*

```
from serial.le import LE
le = LE(n_components=6, k=7, solver='slepc', sigma=1, normalize=False)
embedding_le = le.fit_transform(data)
```

**Funding**

This project is funded by TÜBİTAK (The Scientific and Technological Research Council of Turkey) under the code 120E281, promoting advanced research in manifold learning and data science.

*Note: This is an excerpt from the Scaman GitHub readme file, included in the thesis for reference.*