TÉCNICO LISBOA

Maximum Variance Unfolding on Disjoint Manifolds

João André Roque Costa

Thesis to obtain the Master of Science Degree in

Computer Science and Engineering

Supervisors: Francisco Saraiva de Melo
João Tomás Brazão Caldeira

October/November 2025

Declaration
I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Acknowledgments

I would like to thank my parents for their support, guidance and encouragement throughout my life. Without them, nothing would have been possible. For all the sacrifices they made, for the openness and love they have always shown, I am eternally grateful. my family also, my grandparents, aunts, uncles and cousins for their understanding and support throughout all these years.

I would also like to thank my grandparents, aunts, uncles and cousins for their understanding and support throughout all these years.

I would also like to acknowledge my dissertation supervisors Prof. Francisco Melo and Prof. João Caldeira for their insight, support and sharing of knowledge that has made this Thesis possible.

Last but not least, to all my friends and colleagues that helped me grow as a person and were always there for me during the good and bad times in my life.

Thank you all.

Abstract

Dimensionality reduction is a fundamental problem in machine learning and data analysis, particularly when dealing with high-dimensional data that often lies on lower-dimensional manifolds. While linear methods such as Principal Component Analysis (PCA) are widely used, they fail to capture the complex nonlinear relationships present in many real-world datasets. Maximum Variance Unfolding (MVU) is a well-established semidefinite programming-based approach for nonlinear dimensionality reduction that preserves local isometric properties by maintaining neighborhood distances.

However, MVU faces significant computational limitations with large datasets and struggles when data lie on multiple disconnected manifolds—where natural clustering or sampling irregularities create disjoint components in the neighborhood graph. These disconnections render standard MVU infeasible, as the resulting optimization problem becomes computationally intractable or fails to converge.

This thesis presents Maximum Variance Unfolding on Disjoint Manifolds (MVU-DM), a novel extension that addresses these limitations. The proposed method first identifies disjoint components within the dataset using neighborhood graph analysis, then independently applies MVU to each component to obtain local embeddings. The key innovation lies in the global stage, where representative points from each component are selected and inter-component connections are established to create a global embedding that preserves the intrinsic structure of individual manifolds. This decomposition enables parallel processing and makes MVU applicable to larger datasets that were previously computationally infeasible.

Extensive experimental evaluation on both artificial datasets and natural datasets demonstrates that MVU-DM, as the standard MVU, outperform related methods. Furthermore, the results show that the extension not only handles disconnected manifolds effectively but also improves the quality of embeddings by processing each component independently, leading to better preservation of local geometric structures. This work establishes MVU-DM as a practical and scalable solution for nonlinear dimension-

ality reduction in scenarios involving multiple disjoint manifolds.

Keywords

Dimensionality Reduction, Manifold Learning, Optimization problem, Eigenvalue decomposition, Embedding, Maximum Variance Unfolding, Neighborhood Graph

Resumo

A redução de dimensionalidade é um problema fundamental em aprendizagem automática e análise de dados, particularmente quando se lida com dados de alta dimensionalidade que frequentemente se encontram em volumes de menor dimensão. Embora métodos lineares como a Principal Component Analysis (PCA) sejam amplamente utilizados, falham em capturar complexas relações não lineares, presentes em muitos datasets. Maximum Variance Unfolding (MVU) é uma abordagem bem estabelecida baseada em programação semidefinida para redução não linear de dimensionalidade, que preserva propriedades isométricas locais, mantendo as distâncias entre pontos vizinhos.

No entanto, MVU apresenta limitações computacionais quando aplicado em grandes datasets, e principalmente quando os dados se encontram em múltiplos volumes desconexos — onde o agrupamento natural ou irregularidades na amostragem criam conjuntos disjuntos no grafo de vizinhança. Essas desconexões tornam o MVU inviável, visto que o problema de otimização resultante é incapaz de convergir nestas situações.

Esta dissertação apresenta uma nova extensão que aborda estas limitações, o Maximum Variance Unfolding on Disjoint Manifolds (MVU-DM). A metodologia proposta identifica primeiramente os componentes disjuntos do dataset dado, através do grafo de vizinhança, e aplica o MVU, em separado, a cada componente para obter as representações locais. O elemento diferenciador desta extensão é a etapa global, onde são selecionados pontos representativos de cada componente, e estabelecidas conexões inter-componentes para linearizar a globalidade do dataset, preservando a estrutura intrínseca dos volumes individuais. Esta decomposição permite o processamento paralelo e torna o MVU aplicável a datasets maiores do que previamente.

Após uma avaliação experimental em datasets artificiais e naturais, ambos o MVU-DM como o MVU padrão superaram outros métodos comparáveis. Para além disso, os resultados mostram que a extensão não só lida eficazmente com volumes desconexos, como também melhora a qualidade das suas representações ao processar cada componente independentemente, levando a uma melhor preservação das estruturas geométricas locais. Este trabalho estabelece o MVU-DM como uma solução prática e escalável para redução não linear de dimensionalidade em cenários envolvendo múltiplos

volumes disjuntos.

Palavras Chave

Redução de Dimensionalidade, Aprendizagem de Variedades, Problema de otimização, Decomposição de valores próprios, Embedding, Maximum Variance Unfolding, Grafo de Vizinhança

Contents

Acronyms

EVD	Eigenvalue Decomposition
LP	Linear Programming
QP	Quadratic Programming
SOCP	Second-Order Cone Programming
SDP	Semidefinite Programming
PCA	Principal Component Analysis
MDS	Multidimensional Scaling
Isomap	Isometric Mapping
LE	Laplacian Eigenmaps
LLE	Locally Linear Embedding
HLLE	Hessian-based Locally Linear Embedding
KPCA	Kernel Principal Component Analysis
LTSA	Local Tangent Space Alignment
t-SNE	t-distributed Stochastic Neighbor Embedding
MVU	Maximum Variance Unfolding
C-MVU	Colored Maximum Variance Unfolding
L-MVU	Landmark Maximum Variance Unfolding
ENG	Enhanced Neighborhood Graph
MVU-DM	Maximum Variance Unfolding on Disjoint Manifolds
BSC	Broken S-Curve
SR1	Arbitrarily positioned Swiss Rolls
SR2	Parallel Swiss Rolls
FM	Four Moons

COIL20 Columbia Object Image Library

ORL Olivetti Research Laboratory

MIT-CBCL Massachusetts Institute of Technology Center for Biological and Computational Learning

1

Introduction

Contents

1.1 Context and Motivation

In an era where data is starting to get collected from any conceivable source, high-dimensional datasets have become ubiquitous across diverse domains. While the abundance of features in these datasets can provide rich information, it also presents the appearance of the "curse of dimensionality". With the increased number of variables, data points become increasingly sparse, distances lose their discriminative power, and computational complexity grows, making these studies ineffective or computationally prohibitive.

Dimensionality reduction techniques aim to find lower-dimensional representations while preserving the structure of the original data. Linear methods such as Principal Component Analysis (PCA) [1] and Multidimensional Scaling (MDS) [2] have been widely adopted due to their simplicity and theoretical foundations. However, these methods assume that the data lies on a linear subspace, an assumption that is not frequently respected in real-world examples, where data exhibit complex nonlinear relationships.

To address these limitations, nonlinear dimensionality reduction methods have emerged as powerful alternatives. Techniques such as Isometric Mapping (Isomap) [3], Locally Linear Embedding (LLE) [4], and Laplacian Eigenmaps (LE) [5] attempt to capture the intrinsic geometry of nonlinear manifolds by exploiting local neighborhood relationships. Among these methods, Maximum Variance Unfolding (MVU) [6] stands out for its unique approach and theoretical properties.

1.2 Maximum Variance Unfolding: Strengths and Limitations

MVU, also known as Semidefinite Embedding (SDE), formulates dimensionality reduction as a semidefinite programming problem. The method seeks to "unfold" a manifold by maximizing the variance (or equivalently, the sum of squared pairwise distances) of the embedded points while preserving local distances defined by a neighborhood graph. This approach offers several distinctive advantages:

- **Strong theoretical foundation**: MVU is based on convex optimization, ensuring global optimality of the solution within its formulation.
- **Local isometry preservation**: The method maintains distances between neighboring points, preserving the local geometric structure of the manifold.
- **Parameter robustness**: Unlike some alternatives, MVU requires only the specification of the neighborhood size k, making it relatively parameter-insensitive.
- **Metric preservation**: The resulting embeddings maintain meaningful distance relationships, facilitating subsequent analysis and interpretation.

Despite these strengths, MVU faces significant practical limitations that restrict its applicability to real-world datasets:

Computational Complexity: The algorithm's computational complexity of $O((nk)^3)$ [7] makes it prohibitively expensive for large datasets. The semidefinite programming formulation requires solving optimization problems over $n \times n$ matrices, where n is the number of data points, leading to memory requirements that scale quadratically with dataset size.

Connectivity Requirements: MVU assumes that the neighborhood graph constructed from the data is connected. This assumption is frequently violated in practice, particularly when: (i) data exhibit natural

clustering or multimodal distributions; (ii) sampling is irregular or sparse in certain regions; (iii) noise or outliers create artificial disconnections; or (iv) the intrinsic structure consists of multiple separate manifolds.

When the neighborhood graph is disconnected, the resulting semidefinite program becomes ill-conditioned or infeasible, preventing the application of standard MVU.

1.3 The Challenge of Disjoint Manifolds

Real-world datasets frequently exhibit disconnected structure for various reasons. In computer vision, images of different object categories may form separate clusters in feature space. In bioinformatics, gene expression data from different cell types or conditions may lie on distinct manifolds. In social networks, different communities may be weakly connected or entirely separated. Traditional approaches to handle such disconnections include:

- **Preprocessing techniques**: Methods like the Enhanced Neighborhood Graph (ENG) [8] attempt to connect disjoint components by adding edges, but this can distort the intrinsic geometry and increase computational complexity.
- **Separate processing**: Processing each component independently loses global relationships and makes unified analysis difficult.
- **Alternative methods**: Switching to other dimensionality reduction techniques that can handle disconnections, but potentially losing MVU's desirable properties.

None of these approaches fully address the fundamental challenge: how to leverage MVU's strengths while efficiently handling datasets with multiple disjoint manifolds.

1.4 Research Objectives and Contributions

This thesis addresses the limitations of MVU when applied to datasets containing multiple disjoint manifolds. Our primary objective is to develop a method that preserves the theoretical and practical advantages of MVU while making it applicable to disconnected datasets and computationally tractable for large-scale problems.

The main contributions of this work are:

1. **Algorithm Development**: We propose Maximum Variance Unfolding on Disjoint Manifolds (MVU-DM) (Maximum Variance Unfolding on Disjoint Manifolds), a novel extension that decomposes the dimensionality reduction problem into local and global stages, enabling efficient processing of disconnected datasets.

2. **Computational Efficiency**: The proposed method reduces computational complexity from $O(n^3)$ to $\sum_{p=1}^{C} O(n_p^3)$, where C is the number of components and n_p is the size of component p. This decomposition enables parallel processing and significantly improves scalability.

3. **Global Structure Preservation**: We develop strategies for selecting representative points and establishing inter-component connections that preserve global relationships while maintaining local geometric fidelity.

4. **Comprehensive Evaluation**: We provide extensive experimental validation on both artificial and natural datasets, demonstrating superior performance compared to standard MVU and other dimensionality reduction methods across multiple evaluation metrics.

5. **Practical Implementation**: We present algorithmic details and implementation considerations that make the method practically applicable to real-world scenarios.

1.5 Thesis Organization

This thesis is organized as follows:

Chapter 2 provides a comprehensive literature review covering convex optimization foundations, dimensionality reduction techniques (both linear and nonlinear), and existing approaches for handling disconnected manifolds. We examine the theoretical background of MVU and related methods in detail.

Chapter 3 discusses related work, focusing on out-of-sample extensions, the Enhanced Neighborhood Graph method, and other approaches for addressing connectivity issues in manifold learning.

Chapter 4 presents our proposed MVU-DM algorithm, including detailed descriptions of the local and global stages, representative point selection strategies, inter-component connection methods, and computational complexity analysis.

Chapter 5 provides comprehensive experimental evaluation, including dataset descriptions, evaluation metrics, comparative results, and analysis of computational performance. We demonstrate the effectiveness of MVU-DM across various scenarios and discuss the implications of our findings.

Chapter 6 concludes the thesis with a summary of contributions, discussion of limitations, and directions for future research.

This work advances the state-of-the-art in nonlinear dimensionality reduction by making MVU applicable to a broader class of real-world problems while maintaining its desirable theoretical properties and improving its computational tractability.

2

Literature Review

Contents

2.1 Convex Optimization

2.1.1 Convex Sets

Given a set S, it is considered convex if and only if for any pair of points, all points in the straight line between them are contained in the set. This is a simplification of the following condition:

$$(1-\alpha)x + \alpha y \in S, \quad \forall \alpha \in [0,1], \forall x, y \in S. \tag{2.1}$$

2.1.2 Convex Functions

By definition, a function f is convex if, for any pair of values x and y in a convex set, the value of the function in between them is never higher than the line segment between these two points. This can be written as follows:

$$f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha)f(y), \quad \forall \alpha \in [0,1], \forall x, y \in \text{dom}(f). \tag{2.2}$$

There are cases where a function, and thus an optimization problem, might be convex but have no global minimum, as is the case with the exponential function. On the other hand, a constant function would take infinite global minima.

2.1.3 Optimization Problem

An optimization problem consists of finding the optimal values of a set of variables x, possibly within a subset $\Omega \in D$ of their domain (which is represented by constraints), such that some function of interest f is either maximized or minimized. When the purpose of the problem is to pursue a maximization, this optimized function shall be denominated a utility function, while when it is meant to be minimized, it is called a cost function. We call "variables of interest" the variables to be changed to reach the optimal function value. The general form of an optimization problem is represented as:

$$\min_{x \in D} \quad f(x) \tag{2.3}$$

$$\text{s.t.} \quad x \in \Omega, \text{ where } \Omega \subseteq D. \tag{2.4}$$

Each optimization problem can be assigned to distinct classes. If a problem has any kind of constraint, it is declared as a Constrained Problem. If either the objective function is not convex or the constraints define a non-convex set, then the whole problem is denoted as non-convex. For a problem to be convex, all the functions to optimize and the constraints have to be convex.

According to the differences in optimization problems and types of constraints, the optimization problems can be classified as:

- **Linear Programming (LP)** problems take an affine function as an objective function and linear constraints. They are computationally the simplest to solve, usually solvable using the simplex algorithm or the interior point method.

- **Quadratic Programming (QP)** problems involve the optimization of a quadratic function, while also being limited to linear constraints. Depending on the convexity of the problem, a simple gradient method may reach the global minimum if it is an unconstrained problem. While in cases where

the problem presents constraints, the Karush–Kuhn–Tucker conditions is a possible approach, in non-convex cases, when either the objective function or the constraint set is non-convex, a gradient method would need the help of a global search extension to then be able to find the global minima.

- **Second-Order Cone Programming (SOCP)** problems consist of the optimization problems constrained by constraints that form a second order cone, these are formulated as $\|\boldsymbol{A}x+b\|_2 \leq c^\top x+d$.

- **Semidefinite Programming (SDP)** problems represent the optimization problems that are constrained by a linear matrix inequality of the form $x_1\boldsymbol{F}_1 + \ldots + x_n\boldsymbol{F}_n + \boldsymbol{G} \preceq 0$.

2.2 Dimensionality Reduction

Every dataset is set to have an intrinsic dimensionality, which is the minimum number of parameters needed to represent the data without losing information. This intrinsic dimensionality can be lower than the actual number of features or dimensions in the dataset, especially if the data lies on a lower-dimensional manifold within the higher-dimensional space.

The goal of dimensionality reduction is to simplify a dataset while preserving its essential structure and relationships. This is particularly important in high-dimensional spaces, where the curse of dimensionality can make analysis and visualization challenging.

2.2.1 Linear Methods

2.2.1.A Principal Component Analysis

The most commonly used method for dimensionality reduction is PCA [1], which, by comparing the linear relation between each pair of variables, creates a new set of variables in a linear subspace that retains as much variance as possible.

Let's consider a dataset, with n records, represented by D dimensions ($\boldsymbol{X} \in \mathbb{R}^{n\times D}$), which we want to reduce to d variables ($\boldsymbol{Y} \in \mathbb{R}^{n\times d}$), where $d << D$.

For PCA to compare and find the best d variables composed of the original ones, it relates them by computing a similarity matrix, directly from the covariance or correlation between each pair of variables:

$$\text{covariance}(\boldsymbol{X}_i, \boldsymbol{X}_j) = \frac{\sum_k (\boldsymbol{X}_{ik} - \overline{\boldsymbol{X}_i})(\boldsymbol{X}_{jk} - \overline{\boldsymbol{X}_j})}{n}, \quad \text{correlation}(\boldsymbol{X}_i, \boldsymbol{X}_j) = \frac{\text{covariance}(\boldsymbol{X}_i, \boldsymbol{X}_j)}{\sigma_{\boldsymbol{X}_i}\sigma_{\boldsymbol{X}_j}}. \tag{2.5}$$

It is also possible to compute this similarity matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{\mathbb{D}\times\mathbb{D}}$ matricially, following:

$$\boldsymbol{X'}_k = \boldsymbol{X}_k - \overline{\boldsymbol{X}_k} \qquad , \forall k \in \{1, \ldots, D\} \tag{2.6}$$

$$\boldsymbol{\Sigma} = \frac{\overbrace{\boldsymbol{X'}^\top}^{D\times n}\overbrace{\boldsymbol{X'}}^{n\times D}}{\boldsymbol{n}}, \tag{2.7}$$

where $\boldsymbol{X'}_k \in \mathbb{R}^n$ represents the $\boldsymbol{X}_k$ matrix centered by components.

Now that we have a similarity matrix $\boldsymbol{\Sigma}$, we look for a new orthogonal basis that maximizes the variance of the dataset over the minimum components possible. A matrix of such nature is guaranteed to be symmetric and positive semi-definite matrix. Meaning that performing an Eigenvalue Decomposition (EVD) over this matrix, it returns a set of orthogonal eigenvalues and subsequent eigenvectors, representing the orthogonal basis that maximizes the dataset's variance. These pairs of eigenvectors and eigenvalues, when analyzed individually, represent each of the new directions of the new eigenbasis and the scale of the variance of the dataset over that particular direction.

This approach can be applied using a Gram matrix $\boldsymbol{K} = \boldsymbol{X}\boldsymbol{X}^\top$ instead of the covariance matrix. If the input data $\boldsymbol{X}$ is not mean-centered beforehand, the Gram matrix $\boldsymbol{K}$ must be double-centered to ensure the projection maximizes variance.

Mathematically, this centering is achieved using the Centering Matrix $\boldsymbol{H}$:

$$\boldsymbol{H} = \mathbb{I}_n - \frac{\mathbf{1}_n\mathbf{1}_n^\top}{n}, \tag{2.8}$$

where $\mathbb{I}_n$ is the identity matrix of size n (a zero matrix with ones on the diagonal) and $\mathbf{1}_n\mathbf{1}_n^\top$ is a matrix of size $n \times n$ full of 1's.

The centered Gram matrix $\boldsymbol{K}_{\text{centered}}$ is then computed as:

$$\boldsymbol{K}_{\text{centered}} = \boldsymbol{H}\boldsymbol{K}\boldsymbol{H} \tag{2.9}$$

To decompose the similarity matrices, we perform an EVD:

$$\boldsymbol{\Sigma}\boldsymbol{V} = \boldsymbol{V}\boldsymbol{\lambda} \quad \iff \quad \boldsymbol{\Sigma} = \boldsymbol{V}\boldsymbol{\lambda}\boldsymbol{V}^\top, \tag{2.10}$$

where $\boldsymbol{\lambda}$ is the resulting diagonal matrix of eigenvalues, and $\boldsymbol{V}$ is the matrix of columnwise corresponding eigenvectors.

The eigenvalues are then sorted in descending order, and the d biggest ones are selected, along with their corresponding eigenvectors. This way, we ensure that the selected eigenvectors represent the directions of the new basis that retain the most variance of the original dataset. Meaning that $\boldsymbol{\lambda}$ is

reduced from $\mathbb{R}^{D\times D}$ to $\mathbb{R}^{d\times d}$, and $\boldsymbol{V}$ is reduced from $\mathbb{R}^{D\times D}$ to $\mathbb{R}^{D\times d}$.

Finally, to project the data points from the higher-dimensional space to the lower one, perform:

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{V}, \tag{2.11}$$

where $Y \in \mathbb{R}^{n\times d}$ represents the data points in the embedding space (i.e., the PCA projections) and $\boldsymbol{X} \in \mathbb{X}^{n\times D}$ is the source dataset.

We can also reconstruct the data back to the original space by performing:

$$\hat{\boldsymbol{X}} = \underbrace{\boldsymbol{X}\boldsymbol{V}}_{\text{PCA projs.}} \boldsymbol{V}^{\top}. \tag{2.12}$$

We call *embeddings* the representation of the original data points in the lower-dimensional space, but in this particular method, they are also called the PCA projections.

2.2.1.B Multidimensional Scaling

Like PCA, MDS [2, 9] is a linear method that finds the best orientations for new variables from an EVD, but this time the matrix to operate on can consider any form of distance between the original data points.

It can be represented as an optimization problem, where the objective is to minimize the difference of distances between the two spaces, i.e., for a given pair of points on the original space, we want their distance to be equal to the *euclidean* distance between their respective projections in the embedded space.

The closed form methodology starts by building the distance matrix $\boldsymbol{D} \in \mathbb{R}^{n\times n}$ representing the distance between every pair of points. $\boldsymbol{B} \in \mathbb{R}^{n\times n}$ is then calculated as in Equation (2.9):

$$\boldsymbol{B} = -\frac{1}{2}\boldsymbol{H}\boldsymbol{D}^2\boldsymbol{H}, \tag{2.13}$$

note that $\boldsymbol{D}^2$ represent the element-wise squared distance matrix.

After that, an EVD can be performed over the matrix $\boldsymbol{B}$ and we can select the eigenvectors corresponding to the d biggest eigenvalues, and continue as in Equation (2.10).

MDS can be written as an optimization problem:

$$\min_{\boldsymbol{Y}} \quad \sum_{ij} \left(d_{ij} - \|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2\right)^2, \tag{2.14}$$

where $\boldsymbol{Y} = \left[\boldsymbol{y}_1, \cdots, \boldsymbol{y}_n\right] \in \mathbb{R}^{n\times d}$, and d_{ij} represents a distance function between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ (i.e., points in the original space).

Classical MDS, returns the same solution as PCA when the distance function d_{ij} is the Euclidean

distance, that is, $d_{ij} = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2$. In this case, the optimization problem can be rewritten as:

$$\min_{\boldsymbol{Y}} \quad \sum_{ij} \left(\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2 - \|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2\right)^2 \tag{2.15}$$

In the same circumstances, since we know that $\boldsymbol{B}$ will be symmetric and positive semidefinite, we can directly calculate it from the original dataset:

$$\boldsymbol{B} = (\boldsymbol{H}\boldsymbol{X})\,(\boldsymbol{H}\boldsymbol{X})^\top, \tag{2.16}$$

further proceeding with the closed-form approach.

A – Observations Although linear methods can be very useful and accurate on many linear cases, because they assume linearity between variables, when put to use on data sampled from non-linear manifolds they can't capture this complex relation between variables, so their tendency to underperform in function of the curvature of the nonlinear manifold.

2.2.2 Non-Linear Methods

Non-linear methods are designed to capture the complex relations between variables that linear methods fail to represent. They are usually based on the assumption of local linearity, where the manifold can be approximated as linear in small regions around each point. This means that the Euclidean distance between two very close points can be used as an approximation of the true distance along the manifold, ignoring its curvature in that small region.

2.2.2.A Isometric Mapping (Isomap)

The Isomap [3] reduction method can be seen as an extension of MDS, where the distance matrix to be used is not calculated by Euclidean distances but rather by geodesic distances.

Local linearity is an important and common assumption made by many non-linear methods, where the curvature of the manifold in the small region around a point can be considered linear. This means that the true distance between two very close points can be approximated by the Euclidean distance, ignoring the curvature of the manifold in that small region. This is the basis for many non-linear methods, including Isomap.

The geodesic distance is the theoretical distance between two points along a manifold. In practice, by assuming local linearity, the geodesic distance is approximated by the cumulative distance between points along the manifold. It is then possible to, following shortest paths finding algorithms like Dijkstra or Floyd-Warshall, build a geodesic distance matrix.

Having calculated the Distance matrix, this approach follows closely the MDS's methodology, of minimizing the difference between distances in the original and embedded spaces, as in Equation (2.14).

On contrary to the distance in the original space, the distance in the embedded space can always be measured by the Euclidean distance, as the embedded space is always Euclidean.

Like MDS, we can solve the problem in closed form by squaring the distance matrices and centering them.

2.2.2.B Locally Linear Embedding (LLE)

To mitigate MDS and Isomap's main flaw of giving too much focus to relations between points that are far apart, the LLE [4] bases itself only on the relation between neighbor points.

Because this dimensionality reduction method is based on translation, rotation, and rescaling, by creating a matrix of influences $\boldsymbol{W}$ between neighboring points, it is then able to use this relation matrix to embed the data. The optimization problem is formulated as:

$$\min_{\boldsymbol{Y}} \quad \sum_i \|\boldsymbol{y}_i - \sum_j w_{ij}\boldsymbol{y}_{ij}\|^2, \tag{2.17}$$

where $w_{ij} \in [0, 1]$ represents how much a point j influences on i's position. Note that we only calculate influences between points that are k-nearest neighbors. This is how the weight matrix is calculated:

$$\min_{\boldsymbol{W}} \quad \sum_i \|\boldsymbol{x}_i - \sum_j w_{ij}\boldsymbol{x}_{ij}\|^2 \tag{2.18}$$

$$\text{s.t.} \quad \sum_{j \in \mathcal{N}_i} w_{ij} = 1, \quad \forall i = 1, \ldots, n, \tag{2.19}$$

where $\mathcal{N}_i$ is the set of neighbors of i. Consequently, the values of $\boldsymbol{W}$ for points that are not in the neighborhood of i are left as 0, making it a sparse matrix.

LLE's optimization problem minimizes the difference between the current position of a given point and the position that its neighbors would pull it to, originally. This creates a trivial solution where all points coincide at the origin. To fix this, the constraint $\|y_i^{(k)}\|^2 = 1$ for all k dimensions is added, preventing points from ending up in the origin.

From the computed weights matrix $\boldsymbol{W}$, the optimization problem in Equation (2.17) can be solved as:

$$\sum_i \|\boldsymbol{y}_i - \sum_j \boldsymbol{w}_{ij}\boldsymbol{y}_{ij}\|^2 = (\boldsymbol{Y} - \boldsymbol{W}\boldsymbol{Y})^\top(\boldsymbol{Y} - \boldsymbol{W}\boldsymbol{Y}) \tag{2.20}$$

$$= \boldsymbol{Y}^\top(\mathbb{I} - \boldsymbol{W})^\top(\mathbb{I} - \boldsymbol{W})\boldsymbol{Y} \tag{2.21}$$

$$= (\mathbb{I} - \boldsymbol{W})^\top(\mathbb{I} - \boldsymbol{W}), \tag{2.22}$$

where $\mathbb{I} \in \mathbb{R}^{n \times n}$ is the identity matrix.

This optimization problem results in an embedding matrix $\boldsymbol{M} \in \mathbb{R}^{n \times n}$ to which can be applied an EVD as in Equation (2.10), where the selection of the d smallest eigenvectors ends up with the usual eigenbasis for the embedded space.

2.2.2.C Laplacian Eigenmaps (LE)

Like LLE, LE [5] creates a weight matrix $\boldsymbol{W}$ which quantifies the influence of each neighbor of a given point in its coordinates, assuming local linearity.

However, LE now minimizes the weighted distance to each neighbor point directly:

$$\min_{\boldsymbol{Y}} \quad \sum_{ij} \|\boldsymbol{y}_i - \boldsymbol{y}_j\|^2 w_{ij}, \tag{2.23}$$

where $\boldsymbol{W}$ is a sparse matrix computed using the Gaussian kernel function:

$$w_{ij} = e^{-\frac{\|\boldsymbol{y}_i - \boldsymbol{y}_j\|^2}{2\sigma^2}}. \tag{2.24}$$

Meaning that neighbor points in the high-dimensional space are put as close together as possible in the embedding space.

To further solve the optimization problem, the following diagonal degree matrix $\boldsymbol{M} \in \mathbb{R}^{n \times n}$ is built based on $m_{ii} = \sum_j \boldsymbol{w}_{ij}$, consisting of the sum of the weights at each point, note that they do not sum up to 1 as in Equation (2.19).

The problem can then be decomposed into:

$$\sum_{ij} \|\boldsymbol{y}_i - \boldsymbol{y}_j\|^2 \boldsymbol{w}_{ij} = \underbrace{\sum_i \|\boldsymbol{y}_i\|^2 m_{ii} + \sum_j \|\boldsymbol{y}_j\|^2 m_{jj}}_{2\boldsymbol{Y}\boldsymbol{M}\boldsymbol{Y}^\top} - \underbrace{2\sum_{ij} (\boldsymbol{y}_i \boldsymbol{y}_j^\top) \boldsymbol{w}_{ij}}_{2\boldsymbol{Y}\boldsymbol{W}\boldsymbol{Y}^\top}, \tag{2.25}$$

optimizing this problem finds the same solution as the following:

$$\min_{\boldsymbol{Y}} \quad \boldsymbol{Y}^\top \boldsymbol{L} \boldsymbol{Y} \tag{2.26}$$

$$\text{s.t.} \quad \boldsymbol{Y}^\top \boldsymbol{M} \boldsymbol{Y} = \boldsymbol{I}_{\boldsymbol{n}}, \tag{2.27}$$

where $\boldsymbol{L} = \boldsymbol{M} - \boldsymbol{W}$, and the constraint is needed to prevent the same trivial solution as in LLE.

It can then be solved as the eigenvalue problem:

$$\boldsymbol{L}\boldsymbol{V} = \boldsymbol{\lambda}\boldsymbol{M}\boldsymbol{V}, \tag{2.28}$$

where selecting the d smallest eigenvectors corresponds to the desired low-dimensional representation.

2.2.2.D Hessian Locally Linear Embedding (HLLE)

While LLE computes the weighted embedded positions based on the k-nearest neighbor's coordinates, the Hessian-based Locally Linear Embedding (HLLE) [10] approximates the curviness of the manifold from the local Hessian and minimizes it in order to flatten the dataset.

To find the Hessian matrix, HLLE assumes local linearity across the neighborhood of each point and computes the tangent space around the k-neighborhood of each point. This is done by computing the principal components of each local patch of the manifold, as in PCA.

Afterwards, the matrix $\boldsymbol{Z}$ is built from the cross-product between the principal components at each point, and a constant column of 1's. This matrix is then orthogonalized through the Gram-Schmidt process.

The local tangent Hessian estimation $\boldsymbol{H}_i$ is now built by transposing the selection of the last $\frac{d(d+1)}{2}$ columns of $\boldsymbol{Z}$, representing the curviness of each local patch. And the global Hessian estimator $\boldsymbol{\mathcal{H}}_l m$ consists of:

$$\boldsymbol{\mathcal{H}}_{lm} = \sum_i \sum_j \left((\boldsymbol{H}_i)_{jl} \times (\boldsymbol{H}_i)_{jm} \right). \tag{2.29}$$

HLLE proceeds by minimizing the curvature of the dataset, described by $\boldsymbol{\mathcal{H}} \in \mathbb{R}^{n \times n}$, solving the eigenvalue problem as in Equation (2.10) and selecting the eigenvectors correspondent to the d smallest non-zero eigenvalues results in the $\boldsymbol{Y} \in \mathbb{R}^{n \times d}$ embedded data.

Importantly, the number k of nearest neighbors around each point to approximate the Hessian must be such that $k > d(1 + \frac{1+d}{2})$, where d is the number of components found by PCA. This means that for regions of the manifold that are nonlinear enough, the number of components necessary to represent the data in a linear subspace may result in a very large k, which may fail in capturing local geometry. This seems to happen in practice, as demonstrated by experiments on natural datasets, which tend to lie on nonlinear manifolds [10].

2.2.2.E Local Tangent Space Analysis (LTSA)

Compared to HLLE, Local Tangent Space Alignment (LTSA) [11] also describes its input dataset as the tangent space of the manifold at each data point.

From this local tangent space $\boldsymbol{\Theta}_i$, the LTSA looks for a mapping into the embedded position of the point in analysis. To put this in practice, the LTSA minimizes the difference between the embedded positions, and the result after performing the mapping on the lower-dimensional point. This is formulated

as the optimization problem:

$$\min_{\boldsymbol{Y},\boldsymbol{L}} \quad \sum_i \|\boldsymbol{y}_i\boldsymbol{J}_i - \boldsymbol{L}_i\boldsymbol{\Theta}_i\|^2, \tag{2.30}$$

where $\boldsymbol{L}$ is the map from each tangent to the objective point $\boldsymbol{y}_i$. $\boldsymbol{J}_i$ is a double-centering matrix transformation equivalent to performing:

$$d_{ij} = -\frac{1}{2}\left(d_{ij} - \frac{1}{k}\sum_{l\in\mathcal{N}_i} d_{il} - \frac{1}{k}\sum_{l\in\mathcal{N}_i} d_{jl} + \frac{1}{k^2}\sum_{l,m\in\mathcal{N}_i} d_{lm}\right), \tag{2.31}$$

where k is the size of the neighborhood of i and $\mathcal{N}_i$ represents the set of its points. This operation ensures that each row and column of the resulting matrix has a mean of 0, and also, the whole matrix has a mean 0.

This optimization method can also be solved as an eigenvalue problem over the alignment matrix $\boldsymbol{B} \in \mathbb{R}^{n\times n}$, which is iteratively computed from a zero matrix, where for the set $\mathcal{N}_i$, the matrix is updated following:

$$\boldsymbol{B}_{\mathcal{N}_i\mathcal{N}_i} = \boldsymbol{B}_{\mathcal{N}_{i-1}\mathcal{N}_{i-1}} + \boldsymbol{J}_k(\boldsymbol{I} - \boldsymbol{V}_i\boldsymbol{V}_i^\top)\boldsymbol{J}_k, \tag{2.32}$$

where $\boldsymbol{V_i}\boldsymbol{V_i}^\top$ represents the PCA projection calculated from the local neighborhood of i, and subsequently, the local alignment of the manifold at the data point i is represented by $\boldsymbol{I} - \boldsymbol{V_i}\boldsymbol{V_i}^\top$. The global alignment matrix $\boldsymbol{B}$ then represents the sum of all the local alignment matrices.

Finishing the process, $\boldsymbol{B}$ is used to find the eigenbasis and subsequently the embedded data, by performing an EVD over $0.5(\boldsymbol{B} + \boldsymbol{B}^\top)$ and selecting the eigenvectors corresponding to the d smallest non-zero eigenvalues.

2.2.2.F t-Distributed Stochastic Neighbor Embedding (t-SNE)

Originating SNE [12], t-distributed Stochastic Neighbor Embedding (t-SNE) [13] is focused on data visualization. That is, it is best fitted for dimensionality reduction into 2 or 3 dimensions. Both require a well-defined final number of dimensions, and capture the differences between data points by computing the conditional probability of a neighborhood $p_{i|j}$, and a similar $q_{i|j}$ for the reduced space. This can be compared with the weights from LE and other weight-based methods, but with its specific calculation:

$$p_{i|j} = \frac{\exp(-\|\boldsymbol{x}_i - \boldsymbol{x}_i\|_D^2/2\sigma^2)}{\sum_{k\neq l}\exp\left(-\|\boldsymbol{x}_k - \boldsymbol{x}_l\|_D^2/2\sigma^2\right)}, \qquad q_{i|j} = \frac{\exp(-\|\boldsymbol{y}_i - \boldsymbol{y}_i\|_d^2)}{\sum_{k\neq l}\exp\left(-\|\boldsymbol{y}_k - \boldsymbol{y}_l\|_d^2\right)} \tag{2.33}$$

Due to being asymmetric, these cause a problem with outlier points (where the distance to most points is large), which causes the point to be further away in the reduced space. t-SNE reduced its impact by using symmetric distances between each point i and j applying the mapping $p_{i|j} = p_{j|i} = \frac{p_{i|j}+p_{j|i}}{2n}$.

Also, to better organize and visualize the data, instead of following a Gaussian distribution of the neighbor points over a given point $\boldsymbol{x}_i$, the t-SNE changes the formulation of $q_{i|j}$ to follow a student t-distribution with one degree of freedom to spread the clusters further while not making the already separated ones too far:

$$q_{i|j} = \frac{(1 + \|\boldsymbol{y}_i - \boldsymbol{y}_j\|^2)^{-1}}{\sum_{k \neq l}(1 + \|\boldsymbol{y}_k - \boldsymbol{y}_l\|^2)^{-1}}. \tag{2.34}$$

This creates a much more balanced gradient function, avoiding situations where SNE would be biased to attract points that were already in seemingly good positions.

The following action is to minimize the sum of the Kullback-Leibler Divergence over all the points:

$$C = KL(P\|Q) \sum_i \sum_j p_{i|j} \log \frac{p_{i|j}}{q_{i|j}}, \tag{2.35}$$

which, due to the symmetry of t-SNE, can be solved with a simple gradient function:

$$\frac{\partial C}{\partial \boldsymbol{y}_i} = 4 \sum_j (p_{i|j} - q_{i|j})(\boldsymbol{y}_i - \boldsymbol{y}_j). \tag{2.36}$$

More importantly, this new probability distribution on the lower-dimensional space also approaches the inverse square law, meaning that, at far distances, the scale of the map becomes virtually irrelevant, making far-apart clusters of map points behave as a single point.

This leads us to the most impactful change in t-SNE: considering clusters of points as single points, the original $O(n^2)$ formulation can be made into $O(n \log(n))$.

Besides, from the fact that the whole structure of this method is focused on 2D and 3D representations of the data, this latest optimization also decays with the increase of dimensions, combined with the fact that the knowledge of final dimensionality a priori, in cases of higher and harder to calculate intrinsic dimensions, t-SNE does not get so competitive comparatively with other dimensionality reduction methods.

2.2.2.G Kernel PCA (K-PCA)

The Kernel Principal Component Analysis (KPCA) [14] is a method generalization from the PCA.

Instead of searching for the best eigenbasis directly from a feature's similarity matrix, the KPCA follows a kernel function to nonlinearly reduce the dimensionality of the data. This operation can be related to MDS where it looks for a new euclidean basis from a given similarity matrix.

Because KPCA no longer computes covariance or correlation between variables, which naturally center the data, a double-centering transformation is now required.

After applying the kernel function and double-centering the resulting kernel $\boldsymbol{K}$, the process continues

with the PCA's solution as in Equation (2.7): solving an EVD on $\boldsymbol{K}$ and selecting the eigenvectors corresponding to the top d eigenvalues.

2.2.2.H Maximum Variance Unfolding (MVU)

MVU [6] has a consequence that ended up fixing KPCA's main flaw, of not having a well-defined kernel function. Its optimization problem can be observed as the search for a good linearizer kernel process. In basic terms, the reasoning behind MVU is to stretch any existing manifold that might exist in the dataset. This is done by maximizing the distance between points of the dataset, while maintaining local isometry between neighboring points. All this, while keeping the data centered, to remove degenerate solutions.

It formulates:

$$\max_{\boldsymbol{Y}} \quad \sum_i \|\boldsymbol{y}_i\|^2 \tag{2.37}$$

$$\text{s.t.} \quad \|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^2 = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2, i \sim j \tag{2.38}$$

$$\sum_i \boldsymbol{y}_i = 0, \tag{2.39}$$

the first constraint maintains local isometry between neighbor points i and j (denoted by $i \sim j$), while the following ensures that the data is centered.

Since this operation virtually flattens the manifold, this means that the Euclidean distance between two points in the linearized manifold can approximate the geodesic distance. Since this takes no approximation to find the real distance along a manifold, Isomap can be seen as an approximation to MVU.

The difficulty of this problem comes from its non-convexity and the inexistence of an equivalent closed-form solution. With that, this optimization problem can be converted[1] into an SDP, and thus a convex problem [6]:

$$\max_{\boldsymbol{K}} \quad \text{trace}(\boldsymbol{K}) \tag{2.40}$$

$$\text{s.t.} \quad \boldsymbol{K}_{ii} + \boldsymbol{K}_{jj} - 2\boldsymbol{K}_{ij} = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2 \quad , i \sim j \tag{2.41}$$

$$\boldsymbol{K} \succeq 0 \tag{2.42}$$

$$\sum_{ij} \boldsymbol{K}_{ij} = 0, \tag{2.43}$$

where $\boldsymbol{K} \in \mathbb{R}^{n \times n}$ represent the matrix of inner products between each pair of data points.

After a solution is reached, the value of $\boldsymbol{K}$ represents a possible input kernel for KPCA.

[1] The conversion calculations and constraint relaxation from the non-convex to the SDP problem can be seen in the Appendix at Section A.1.

To finalize, a regular EVD can be performed as in Equation (2.10) to have a representation in the embedded space with minimal loss of information.

As analyzed in [7], MVU presents a computational complexity of $O((nk)^3)$, i.e., the computational expenses grow cubically with the n number of points and the k number of neighbors to consider as the neighborhood of each point, presenting a much faster growth compared to other methods. This is most influenced by the size of the Gram objective matrix, in $\mathbb{R}^{n \times n}$, of inner products between each pair of data points, and on the size of the neighborhood to consider, creating a constraint for each pair of data points. Along with that, and common to all numerical approaches, solving the final eigenvalue problem has the computational complexity of $O(n^3)$.

2.3 Related Work

2.3.1 Out-of-sample Extension

The out-of-sample extension is a technique used to extend the learned low-dimensional representation of a dataset to new, unseen data points. This is particularly important in manifold learning and dimensionality reduction methods, where the goal is to find a lower-dimensional embedding of high-dimensional data while preserving its intrinsic structure.

This application is most frequently applied due to the incremental additions of new data points to the original dataset. However, it is also crucial for datasets that are too large to be processed in a single batch, as it allows for the efficient embedding of new points without having to recompute the entire embedding from scratch.

Several methods have been proposed for out-of-sample extension, including:

2.3.1.A Nystrom Method

The Nystrom method [15] is a technique used to approximate the eigenfunctions and eigenvalues of large kernel matrices, which are commonly encountered in kernel-based machine learning algorithms. The method is particularly useful for scaling up algorithms that involve large datasets, as it allows for efficient computation of the kernel matrix by using a subset of the data.

The Nystrom method works by selecting a small subset of m representative points, also referenced as landmark points, from the original dataset of size n. The kernel matrix is then computed only for these points, resulting in a smaller $m \times m$ matrix. The remaining entries of the full $n \times n$ kernel matrix are approximated using a Nystrom approximation.

Having computed the kernel matrix $\boldsymbol{K}_m$ for the m representative points, the extension then compute the kernel values between the remaining $n - m$ points and the m representative points, resulting in a

matrix $\boldsymbol{K}_{nm}$ of size $(n-m) \times m$.

The Nystrom approximation to compute the full kernel matrix $\boldsymbol{K}$ can be expressed as:

$$\hat{\boldsymbol{K}} \approx \boldsymbol{K}_{nm} \boldsymbol{K}_{m}^{-1} \boldsymbol{K}_{nm}^{\top} \tag{2.44}$$

where $\boldsymbol{K}_{nm}$ is a simpler, usually low-rank, kernel matrix between all data points and the landmark points.

The original method can then continue, using this approximated kernel matrix $\hat{\boldsymbol{K}}$ in place of the full kernel matrix $\boldsymbol{K}$. By using this approximation, the Nystrom method significantly reduces the computational complexity associated with kernel methods, making them more feasible for large-scale problems. However, the quality of the approximation depends on the choice of landmark points, some examples include: random selection, k-means clustering centroids, or selecting points that are representative of different regions of the data space. There will be more emphasis on this topic in the following sections.

2.3.1.B Landmark Extension

The landmark extension is a general approach to scale up manifold learning algorithms to handle larger datasets. This general approach [16] has been successfully applied on MVU [17], isomap [18] and LLE [19].

The main idea is to select a small subset of landmark points from the dataset and perform the dimensionality reduction only on these points. The remaining points are then embedded based on their relationships with the landmark points. Performance-wise, this approach significantly reduces the computational complexity of the algorithms, making them more scalable to larger datasets.

Compatible to any dimensionality reduction method, the landmark extension can be generalized as follows: (a) select a subset of m landmark points from the original dataset of size n; (b) apply the dimensionality reduction algorithm to the landmark points to obtain their low-dimensional representations; (c) in the original space, for each non-landmark point, compute its relationships (e.g., distances or similarities) with the landmark points; (d) use these relationships to infer the low-dimensional representation of the non-landmark points.

Firstly, the landmarks can be chosen randomly from the dataset, which is simple but may not always capture the data's structure effectively. Alternatively, clustering methods can be used to select representative points that better reflect the data distribution. Another approach is to use a MinMax strategy, which iteratively selects points that are maximally distant from the already chosen landmarks, ensuring a more uniform coverage of the data space. The effectiveness of the landmark extension largely depends on how well these landmark points capture the underlying manifold structure.

After selecting the set of landmarks $l_{i=1}^{m}$, the $\boldsymbol{W} \in \mathbb{R}^{n \times n}$ matrix of reconstruction weights is computed. This matrix can be constructed using various methods, such as k-nearest neighbors or radial

basis functions, depending on the specific characteristics of the data and the dimensionality reduction technique used. For the Landmark Maximum Variance Unfolding (L-MVU) [17], the relationships are computed by minimizing the reconstruction error in the original space, ensuring that the local geometry is preserved, solving the following optimization problem:

$$\min_{\boldsymbol{W}} \quad \sum_{i} \|\boldsymbol{x}_i - \sum_{j=1}^{m} W_{ij}\hat{\boldsymbol{x}}_j\|^2 \tag{2.45}$$

$$\text{s.t.} \quad \sum_{j} W_{ij} = 1, \forall i, \tag{2.46}$$

additionally, $W_{ij} = 0$ if $\hat{\boldsymbol{x}}_j$ is not among the k-nearest neighbors of $\boldsymbol{x}_i$.

A linear equation system is now solved, to compute the matrix $\boldsymbol{\Phi} \in \mathbb{R}^{n\times n}$, expressing each point as a linear combination of the landmark points:

$$\boldsymbol{\Phi} = (\mathbb{I}_n - \boldsymbol{W})^T(\mathbb{I}_n - \boldsymbol{W}). \tag{2.47}$$

This matrix $\boldsymbol{\Phi}$ can then be decomposed into quadrants:

$$\boldsymbol{\Phi} = \begin{bmatrix} \boldsymbol{\Phi}_{ll} & \boldsymbol{\Phi}_{lu} \\ \boldsymbol{\Phi}_{ul} & \boldsymbol{\Phi}_{uu} \end{bmatrix}, \tag{2.48}$$

where $\boldsymbol{\Phi}_{ll}$ corresponds of the influence between landmarks. Since each landmark will stay fixed $\boldsymbol{\Phi}_{ll} = \mathbb{I}_m$.

Subsequently, the rightmost quadrants represent how each point's position is influenced by non-landmark points, thus $\boldsymbol{\Phi}_{lu} = \boldsymbol{0}$. Finally, $\boldsymbol{\Phi}_{ul}$ represent the position of non-landmark points in function of landmark points.

The actual matrix of relationships $\boldsymbol{Q}$ is then computed as:

$$\boldsymbol{Q} = \begin{bmatrix} \mathbb{I}_m \\ \boldsymbol{\Phi}_{uu}^{-1}\boldsymbol{\Phi}_{ul} \end{bmatrix}, \tag{2.49}$$

where $\boldsymbol{Q} \in \mathbb{R}^{n\times m}$.

Finally, considering the subsection of the landmark's kernel matrix $\boldsymbol{L}_{\alpha\beta} = l_\alpha \cdot l_\beta$, the full kernel matrix $\boldsymbol{K}$ is found solving the adapted MVU's optimization problem:

$$\max_{\boldsymbol{L}} \quad \text{trace}(\boldsymbol{Q}\boldsymbol{L}\boldsymbol{Q}^\top) \tag{2.50}$$

$$\text{s.t.} \quad \boldsymbol{L} \succeq 0 \tag{2.51}$$

$$\sum_{ij}(QLQ^\top)_{ij} = 0 \tag{2.52}$$

$$(QLQ^\top)_{ii} + (QLQ^\top)_{jj} - 2(QLQ^\top)_{ij} = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2 \quad , \forall i, j \in \boldsymbol{G}, \tag{2.53}$$

2.3.2 Enhanced Neighbourhood Graph

The ENG [8] extension focuses on handling the case of non-uniform sampling densities in the input data. In many real-world scenarios, data points may be unevenly distributed across the manifold, leading to challenges in accurately capturing the underlying structure using traditional neighborhood graph construction methods. This can result in poor embeddings, as areas with high sampling density may dominate the neighborhood relationships, while sparsely sampled regions may be inadequately represented.

The ENG approach addresses this issue by introducing an addition to the usual k-nearest neighbors (k-NN) graph construction. Instead of solely relying on the k-NN method, which can be sensitive to local density variations, ENG analyses the possible relationship between disconnected groups of points. This is achieved by adaptively and iteratively connecting these groups based on their proximity and the overall structure of the data.

Assuming the dataset given to this method forms a disconnected neighbourhood graph, the algorithm starts by identifying the connected components of the initial k-NN graph. During each iteration, the algorithm considers each disconnected component and evaluates potential connections to its closest neighboring component. The computation to find the best connections between two components is as follows:

- Compute the intrinsic dimensionality d of the data using a Maximum Likelihood Estimator [20];
- Compute the average contribution ratio $\eta^{(d)}$ of the top d principal directions along the component;
- Find the l shortest connections between the two components (l starts at $d+1$);
- Compute the connections contribution ratio $\eta^{(d,l)}$ of the top d singular values;
- While $\eta^{(d,l)} \geq \eta^{(d)}$ connect the two components with the l shortest connections, increment l and repeat from step (c), l is capped to the size of the smaller component.

The ENG method starts by calculating an $\eta_i^{(d)}$ parameter. This variable, given the neighborhood of a point i, represents how the d most representative directions explain the variance of the data around that point. Computing:

$$\eta_i^{(d)} = \sum_{j=1}^{d} \sigma_{ij} / \sum_{j=1}^{min(D,k)} \sigma_{ij}, \tag{2.54}$$

where d is the intrinsic dimensionality found by the Maximum Likelihood Estimator, and σ_{ij} is the j-th largest singular value of the neighborhood around i.

The average contribution ratio $\eta^{(d)}$ of the top d principal directions along the entire dataset is then computed as:

$$\eta^{(d)} = \frac{1}{n} \sum_{i=1}^{n} \eta_i^{(d)}. \tag{2.55}$$

Given a pair of components p and q, the algorithm then computes the ranking of l shortest connections between the pair:

$$\boldsymbol{\Delta}^{\{p,q\}(l)} = \boldsymbol{Y}^{\{p\}(l)} - \boldsymbol{Y}^{\{q\}(l)}, \tag{2.56}$$

where $\boldsymbol{Y}^{\{p\}(l)} \in \mathbb{R}^{l \times d}$ is the list of the l closest points from component p to component q.

The sorted singular values $\sigma_1^{\{p,q\}(l)} > \sigma_2^{\{p,q\}(l)} > ... > \sigma_{min(D,l)}^{\{p,q\}(l)}$ of the matrix $\boldsymbol{\Delta}^{\{p,q\}(l)}$ are then computed. The contribution ratio of the top d singular values is then calculated as:

$$\eta^{(d,l)} = \sum_{j=1}^{d} \sigma_j^{\{p,q\}(l)} / \sum_{j=1}^{min(D,l)} \sigma_j^{\{p,q\}(l)}. \tag{2.57}$$

Having both $\eta^{(d)}$ and $\eta^{(d,l)}$, the algorithm adds the shortest available connections until the condition $\eta^{(d,l)} \geq \eta^{(d)}$ is no longer met, or all connections have been considered.

Finally, for each iteration, if the graph is not fully connected, the extension applies the process above between each component and its closest component. This process is repeated until the graph is fully connected.

2.3.3 Colored Maximum Variance Unfolding (C-MVU)

The Colored Maximum Variance Unfolding (C-MVU) [21] is an extension of MVU that incorporates class label information into the dimensionality reduction process. This method is particularly useful in supervised learning scenarios where the data points are associated with specific classes or categories.

In C-MVU, the optimization problem is modified to not only maximize the variance of the data in the reduced space but also to ensure that points belonging to the same class are mapped closer together, while points from different classes are pushed further apart. This is achieved by introducing additional constraints based on the class labels.

The modified optimization problem can be formulated as:

$$\max_{\boldsymbol{Y}} \quad \sum_{ij} \|\boldsymbol{y}_i - \boldsymbol{y}_j\|^2 - \lambda \sum_{i,j} c_{ij} \|\boldsymbol{y}_i - \boldsymbol{y}_j\|^2 \tag{2.58}$$

$$\text{s.t.} \quad \sum_{i} \boldsymbol{y}_i = 0 \tag{2.59}$$

$$\|\boldsymbol{y}_i - \boldsymbol{y}_j\|^2 = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2, \forall i, j \in \boldsymbol{G}, \tag{2.60}$$

where c_{ij} is a binary indicator that equals 1 if points i and j belong to different classes and 0 otherwise, and λ is a regularization parameter that controls the trade-off between maximizing variance and enforcing class separation.

By incorporating class label information, C-MVU aims to produce embeddings that are more discrim-

inative for classification tasks. The resulting low-dimensional representations are expected to enhance the performance of classifiers by preserving both the intrinsic structure of the data and the relationships between different classes.

Similar to MVU, C-MVU can be solved using SDP techniques, although the inclusion of class-based constraints may increase the complexity of the optimization problem. Nonetheless, C-MVU provides a valuable approach for supervised dimensionality reduction, particularly in scenarios where class separability is a key concern.

3

Maximum Variance Unfolding on Disjoint Manifolds

Contents

Both out-of-sample extensions and disconnection handling present improvements to the original MVU algorithm. However, they can be adjusted to overcome some of MVU's unchanged limitations.

By significantly reducing the overall density of the data, the out-of-sample extensions might emphasize the presence of noise and disconnections. This is particularly true when the data is already sparse or contains outliers. In such cases, the reduced density can lead to a loss of important structural information, making it more challenging to capture the underlying manifold accurately. Additionally, out-of-sample extensions typically rely on the assumption that new data points are similar to those in the training set. If the new points are significantly different or if they lie in regions of the space that were not well-represented in the original dataset, the extension may fail to provide meaningful embeddings. This latter issue is particularly relevant when using the Nyström method to treat disjoint datasets, where the kernel is computed for one component and then extended to the rest of the components, which may be

very different from the first one.

The ENG method assumes a constant intrinsic dimensionality along different regions of the manifold. This assumption is not always valid, as some manifolds may exhibit varying intrinsic dimensionalities in different areas. Additionally, this intrinsic dimensionality is required to be computed as a prerequisite, before any type of linearization is done.

Furthermore, and general to the connection inserting extensions, attempting to make the k-NN graph fully connected, meaning the dimensionality reduction process is guaranteed to be computationally as expensive or more than the regular MVU. Particularly, the ENG creates multiple connections to represent the relationship between two disconnected components, which may be unnecessary or further complicate the dimensionality reduction task. The amount of connections created is also dependent on the intrinsic dimensionality of the data, which is either a user-defined parameter or estimated using a Maximum Likelihood Estimator method. This estimation may not be accurate, leading to suboptimal connections between components.

To address these limitations, a new extension is proposed, combining the advantages of both out-of-sample extensions and disconnection handling methods. This new extension aims to efficiently manage disconnected data while also being capable of handling large datasets through parallelization.

The proposed extension, named MVU-DM, will then handle the infeasibility problem by parallelizing the computation of each disjoint component and addressing these disconnections by linearizing the global structure of the disjoint components.

3.1 Algorithm Description

The MVU-DM is an extension of the MVU algorithm designed to handle datasets that are composed of multiple disjoint manifolds, as is ENG. This situation often arises in real-world applications where data points may belong to different categories, classes, or just high variability among the data, leading to natural separations in the data distribution. In other cases, simply sampling irregularities in the data, or the presence of noise, may lead to disconnections in the k-NN graph. In these cases, MVU-DM can also be applied to effectively manage these disconnections.

The core idea behind MVU-DM is to first identify and separate the disjoint manifolds within the dataset, and independently apply MVU to each manifold to learn their respective low-dimensional embeddings. Finally, linearize the global structure of the disconnected components, finding the best relations between disjoint manifolds, and inserting the locally linearized components into the global structure, in the best way possible.

The MVU-DM approach can be described as follows:

- Find the disjoint components of the dataset (e.g., building the k-NN graph);

- Separate the dataset $\boldsymbol{X}$ into disjoint components: $\boldsymbol{\mathcal{X}}_p$ representing the subset of point indices in component p, $\sum_{p=1}^{C} |\boldsymbol{\mathcal{X}}_p| = |\boldsymbol{X}|$;
- Run MVU on each component $\boldsymbol{\mathcal{X}}_p$ independently, obtaining $\boldsymbol{\mathcal{Y}}_p$, the local embeddings;
- Compute the set of representative point indices $\boldsymbol{\mathcal{Z}}_p$ for each component $\boldsymbol{\mathcal{Y}}_p$;
- Compute the inter-component connections $\boldsymbol{\mathcal{L}}$, calculated in the original space, adding the used points to $\boldsymbol{\mathcal{Z}}_p$;
- Run the global MVU on $\bigcup_{p=1}^{C} \boldsymbol{\mathcal{Z}}_p$, using the connections $\boldsymbol{\mathcal{L}}$ and retaining intra-component distances, obtaining $\boldsymbol{Z}_p$, the global embeddings;
- Find the transformation that maps $\boldsymbol{\mathcal{Z}}_p$ to $\boldsymbol{Z}_p$, apply it on each component $\boldsymbol{\mathcal{Y}}_p$, obtaining the final embedding $\boldsymbol{Y}_p$;
- Combine the embeddings $\boldsymbol{Y}_p$ into the final embedding $\boldsymbol{Y}$.

This process is also represented in algorithmic form in Algorithm 3.1.

Algorithm 3.1: Maximum variance unfolding on disjoint manifolds

Input: $\boldsymbol{X} \in \mathbb{R}^{(n \times D)}$
Hyperparameter: $k \in \mathbb{R}$
Output: $\boldsymbol{Y} \in \mathbb{R}^{(N \times d)}$
begin

$$\boldsymbol{\mathcal{X}}_1, \ldots, \boldsymbol{\mathcal{X}}_C \longleftarrow build\_neighborhood\_graph(\boldsymbol{X}, k)$$
$$\boldsymbol{\mathcal{Y}}_1, \ldots, \boldsymbol{\mathcal{Y}}_C \longleftarrow MVU(\boldsymbol{X}, \boldsymbol{\mathcal{X}}_c), \quad p = 1, \ldots, C$$
$$\boldsymbol{\mathcal{Z}}_1, \ldots, \boldsymbol{\mathcal{Z}}_C \longleftarrow choose\_representative\_points(\boldsymbol{\mathcal{Y}}_p), \quad p = 1, \ldots, C$$
$$\boldsymbol{\mathcal{L}}, \boldsymbol{\mathcal{Z}}_1, \ldots, \boldsymbol{\mathcal{Z}}_C \longleftarrow choose\_intercomponent\_connections(\boldsymbol{X}, \{\boldsymbol{\mathcal{X}}_p\}_{p=1}^{C}, \{\boldsymbol{\mathcal{Z}}_p\}_{p=1}^{C})$$
$$\boldsymbol{Z}_1 \ldots, \boldsymbol{Z}_C \longleftarrow global\_MVU(\boldsymbol{X}, \{\boldsymbol{\mathcal{X}}_p\}_{p=1}^{C}, \{\boldsymbol{\mathcal{Y}}_p\}_{p=1}^{C}, \{\boldsymbol{\mathcal{Z}}_p\}_{p=1}^{C}, \boldsymbol{\mathcal{L}})$$
$$\boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_C \longleftarrow translate\_components(\boldsymbol{Y}_p, \boldsymbol{\mathcal{Z}}_p), \quad p = 1, \ldots, C$$

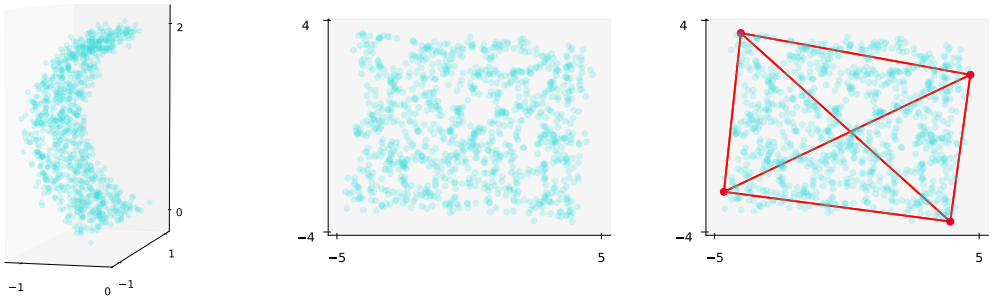
Return $[\boldsymbol{Y}_1 \quad \cdots \quad \boldsymbol{Y}_C]$

3.1.1 Local Stage

The local stage of the MVU-DM algorithm focuses on independently applying the MVU algorithm to each disjoint component of the dataset. This stage is crucial for capturing the intrinsic structure of each manifold while ensuring that the local relationships between data points are preserved. This is the key improvement over the vanilla MVU and the ENG. These, instead of computing disjoint components individually, connect them, ending up with at least $n \times k$ connections, and a kernel matrix of size $n \times n$. MVU-DM instead embeds each component individually, resulting in smaller kernel matrices of size $n_p \times n_p$, where n_p is the number of points in component p. It is mathematically proven that the computational

complexity of a regular MVU grows cubically with the number of points [7], thus, by dividing the dataset into C components, the overall computational complexity is significantly reduced to $\sum_{p=1}^{C} O((n_p k)^3)$.

To distinguish between the different components, the extension builds the k-NN graph, with k as a user-defined parameter. The graph is then analyzed to find its connected components, which represent the disjoint manifolds in the dataset, possibly through a Depth-First Search (DFS) or Breadth-First Search (BFS) algorithm. Each connected component is then treated as an independent dataset, and the MVU algorithm is applied to each component separately.



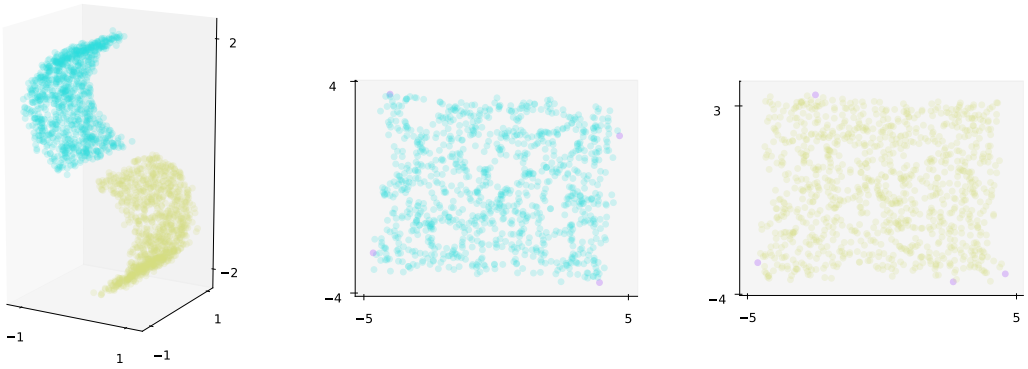
(a) Original component in $\mathbb{R}^3$ **(b)** Linearized component in $\mathbb{R}^2$ **(c)** Representative component

Figure 3.1: Linearization of a single component

After obtaining each component's embeddings, the next step is to find a reasonable way to position these embeddings in a common low-dimensional space. This is done by selecting a set of reference points for each component, which will be used to represent the component in the global structure. These reference points should ideally capture the essential geometry of the component's geometry and distribution. Several strategies can be employed to select these representative points, including: (a) computing the principal d directions, and selecting the outmost points as representative points, selecting $2d$ points; (b) iteratively selecting the points that are maximally distant from the already chosen representative points, selecting 2^d points; (c) compute the convex hull of each component, and use its vertices as representative points. The methods tested were (a) and (b), with (a) being the default method. The number of points mentioned is the minimum to guarantee the representation of all dimensions; those were the values used for the experiments.

3.1.2 Global Stage

The global stage of the MVU-DM algorithm focuses on integrating the embeddings of the disjoint components obtained from the local stage into a unified low-dimensional space. This stage is crucial for preserving the overall structure and relationships between the different manifolds while ensuring that



(a) Original dataset in $\mathbb{R}^3$ **(b)** First component linearization **(c)** Second component linearization

Figure 3.2: Linearization and representative points computation

the local geometries of each component are maintained.

Not being restricted to representative points, the algorithm starts by finding inter-component connections that best connect disjoint components. Analysed in the original space, different strategies can be applied: (a) iteratively connecting the largest component to its closest component, until the graph is fully connected; (b) connecting each component to its l closest components, where l is a user-defined parameter; (c) iteratively connecting each component to its closest component, until the graph is fully connected. The latter approach is the default process that scikit-learn uses to handle any disconnections it may find. It can be seen as a simpler ENG. The first approach described (a) was the one selected for the experiments, as it was found to produce better results.

Once the inter-component connections are established, the subset of representative points and their computed connections are the main focus. This set does not only consist of the locally calculated representative points from each component, but also includes the points selected on the inter-component connections. From now on, the mentioned set of representative points will refer to this reference set, the inter-component connections refer to the connections between different components, and the intra-component connections refer to the connections within each component.

The MVU algorithm is then applied to this new set of representative points, with an important modification: the intra-component distances remain constant. This means that during the optimization process, the relative positions between points within each component are fixed, allowing only inter-component adjustments. This constraint helps to maintain the local structures learned in the previous stage while enabling a coherent global arrangement.

After obtaining the global embedding of the representative points, the final step involves aligning each component's embedding with its corresponding representative points in the global space. This is done by finding an affine transformation that maps the representative points to their new positions in the

global embedding. It is calculated by solving the equations:

$$\boldsymbol{Z}_p = \boldsymbol{\mathcal{Y}'}_p \boldsymbol{A} + \boldsymbol{b}, \tag{3.1}$$

equivalent to:

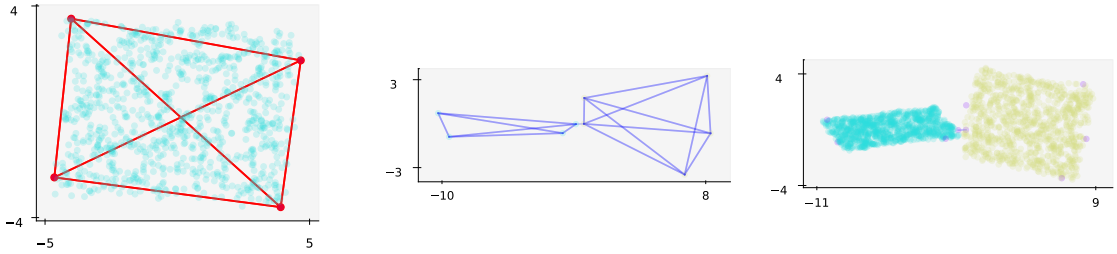
$$\begin{bmatrix} \boldsymbol{A} \\ \boldsymbol{b} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\mathcal{Y}'}_p & \mathbf{1} \end{bmatrix}^{\dagger} \boldsymbol{Z}_p, \tag{3.2}$$

where $\boldsymbol{\mathcal{Y}'}_p$ the local embeddings $\boldsymbol{\mathcal{Y}}_p$ while restricted to the set $\boldsymbol{\mathcal{Z}}_p$, and $\boldsymbol{Z}_p$ is the global embeddings of the same set.

This transformation is then applied to all points within each component's local embedding:

$$\boldsymbol{Y}_p = \begin{bmatrix} \boldsymbol{\mathcal{Y}}_p & \mathbf{1} \end{bmatrix} \begin{bmatrix} \boldsymbol{A} \\ \boldsymbol{b} \end{bmatrix}. \tag{3.3}$$

Finally, the embeddings of all components are combined to form the final low-dimensional representation of the entire dataset: $\boldsymbol{Y} = \bigcup_{p=1}^{C} \boldsymbol{Y}_p$.



(a) Global representation of a component

(b) Global dataset linearized

(c) Final dataset

Figure 3.3: Linearization the global structure and final embedding

4

Results

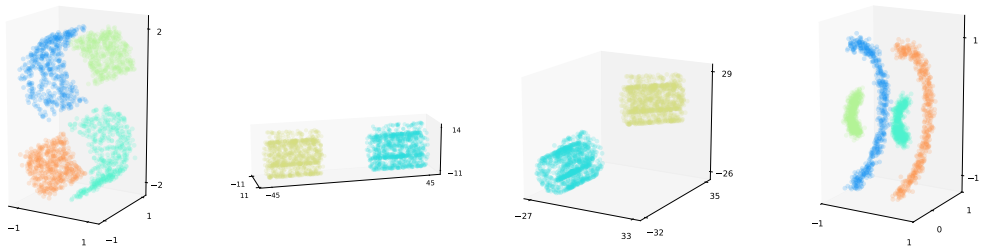
Contents

In this chapter, the results of the experiments conducted to evaluate the performance of the proposed MVU-DM algorithm are presented and analyzed. The evaluation is based on several datasets, both artificial and natural, and includes comparisons with other dimensionality reduction techniques.

The methods compared in the experiments include: 1. **Isomap** [3]; 2. **Isomap+ENG** [3, 8]; 3. **LLE** [4]; 4. **HLLE** [10]; 5. **LE** [5]; 6. **LTSA** [11]; 7. **KPCA** [14]; 8. **MVU** [6].

The results were compared with previous studies: [22] [8], from which were chosen the most relevant methods for the comparison.

All methods are implemented using the Python [23] library `scikit-learn` [24]. Additionally, the MVU also relies on solving an optimization problem, using also the matlab.engine library to interface with MATLAB from Python [23, 25].



(a) Broken S-curve **(b)** Arbitrary Swiss Rolls **(c)** Parallel Swiss Rolls **(d)** Four Moons

Figure 4.1: Artificial datasets used in the experiments: (a) BSC, (b) SR1, (c) SR2, and (d) FM.

4.1 Datasets

The evaluation is performed on a variety of datasets, including both artificial and natural datasets. The artificial datasets include: 1. **Broken S-Curve (BSC)**: A 2D broken, linear manifold, bent into a 3-dimensional S shape, illustrated in Figure 4.1(a); 2. **Arbitrarily positioned Swiss Rolls (SR1)**: Two 2D Swiss Roll manifolds, bent into 3D, and positioned arbitrarily in the same space (see Figure 4.1(b)); 3. **Parallel Swiss Rolls (SR2)**: Two 2D Swiss Roll manifolds, bent into 3D, and positioned parallel to each other (see Figure 4.1(c)); 4. **Four Moons (FM)**: Four 2D half-circle manifolds, positioned in a 3D space (see Figure 4.1(d)).

More precisely, all the artificial datasets consist of 2,000 points, sampled with added Gaussian noise ($\sigma = 0.05$). The BSC dataset is built following the equations:

$$\boldsymbol{X} = \begin{bmatrix} \sin(t) \\ h \\ \text{sign}(t) \cdot (\cos(t) - 1) \end{bmatrix} + \epsilon, \tag{4.1}$$

where $t \in [-1.5\pi, 1.5\pi] \setminus ([-1.1\pi, -0.9\pi] \cup [-0.1\pi, 0.1\pi] \cup [0.9\pi, 1.1\pi])$, $h \in [0, 2]$, and ϵ is Gaussian noise. The SR1 and SR2 datasets are built from the base definition of the Swiss roll:

$$\boldsymbol{X}_{\text{raw}} = \begin{bmatrix} t \cdot \cos(t) \\ h \\ t \cdot \sin(t) \end{bmatrix} + \epsilon, \tag{4.2}$$

where $t \in [1.5\pi, 3\pi]$, $h \in [0, 30]$, and ϵ is Gaussian noise.

For the SR1 dataset, two rolls are created, being subject to the following transformations:

$$\boldsymbol{X}_1 = \begin{bmatrix} \cos(-\pi/4) & -\sin(-\pi/4) & 0 \\ \sin(-\pi/4) & \cos(-\pi/4) & 0 \\ 0 & 0 & 1 \end{bmatrix} \boldsymbol{X}_{\text{raw}} + \begin{bmatrix} 20 \\ 20 \\ 30 \end{bmatrix} \tag{4.3}$$

$$\boldsymbol{X}_2 = \boldsymbol{X}_{\text{raw}} + \begin{bmatrix} 0 \\ -20 \\ 0 \end{bmatrix}. \tag{4.4}$$

For the SR2 dataset, one of the datasets is left unchanged, while the other is translated:

$$\boldsymbol{X}_2 = \boldsymbol{X}_{\text{raw}} + \begin{bmatrix} 0 \\ -20 \\ 0 \end{bmatrix}. \tag{4.5}$$

A single pair of moons is generated using the equations:

$$\boldsymbol{X}_{\text{raw}} = \begin{bmatrix} \sin(t) + \epsilon \\ \cos(t) + \epsilon \end{bmatrix} \cup \begin{bmatrix} \frac{1-\sin(t)}{4} + \epsilon \\ \frac{\cos(t)}{4} + \epsilon \end{bmatrix}, \tag{4.6}$$

where $t \in [0, \pi]$ and ϵ is Gaussian noise. The FM dataset is then created:

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{X}_{\text{raw}} & \mathbf{0} \\ \boldsymbol{X}_{\text{raw}} & \mathbf{1} \end{bmatrix}, \tag{4.7}$$

where $\mathbf{0}$ and $\mathbf{1}$ are vectors of zeros and ones, respectively, with the same number of rows as $\boldsymbol{X}_{\text{raw}}$. The final dataset consists of four half-moons positioned in a 3D space.

The natural datasets include: 1. **Columbia Object Image Library (COIL20)**: A dataset of 1,440 grayscale images (128x128 pixels) of 20 different objects, each captured from various angles; 2. **Olivetti Research Laboratory (ORL)**: A dataset of 400 grayscale images (112x92 pixels) of 40 different individuals' faces; 3. **Massachusetts Institute of Technology Center for Biological and Computational Learning (MIT-CBCL)**: A dataset of 2,059 grayscale images (64x64 pixels) of faces, with variations in lighting and facial expressions; 4. **Olivetti**: directly extracted from the scikit-learn library, this dataset imports the original data from the ORL dataset, but adds a data treatment layer (e.g., each image is resized to 64x64 pixels, removing the background and relying more on each face's characteristics).

4.2 Metrics

To assess the quality of the embeddings produced by MVU-DM and other algorithms, three metrics are employed: 1-NN classification error, Trustworthiness, and Continuity.

4.2.1 1-NN Classification Error

The 1-NN classification error measures the accuracy of the embeddings by evaluating how well they can be used for nearest neighbor classification. A lower error rate indicates better preservation of the original data structure.

Given the embeddings $\boldsymbol{Y} \in \mathbb{R}^{n \times d}$ and the original labels $\boldsymbol{L} \in \mathbb{R}^n$, the 1-NN classification error is computed as follows:

$$\text{1-NN Error} = \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}(L_i \neq L_{\text{NN}(i)}), \tag{4.8}$$

where $\mathbb{I}$ is the indicator function, and $\text{NN}(i)$ is the index of the nearest neighbor of point i in the embedding space.

4.2.2 Trustworthiness

Trustworthiness measures how well the local structure of the original data is preserved in the embedding. This metric penalizes points that are close in the embedding but were far apart in the original space. The trustworthiness $T(k)$ for a given neighborhood size k is defined as:

$$T(k) = 1 - \frac{2}{nk(2n - 3k - 1)} \sum_{i=1}^{n} \sum_{j \in U_k(i)} (r(i,j) - k), \tag{4.9}$$

where $U_k(i)$ is the set of k nearest neighbors of point i in the original space, but no longer in the embedding; $r(i,j)$ is the rank of j in the neighborhood of point i in the embedding space, this rank is the position that j is in the sorted list of closest points to i.

4.2.3 Continuity

Analogously, the continuity penalizes points that were close in the original space but are far apart in the embedding. The continuity $C(k)$ for a given neighborhood size k is defined as:

$$C(k) = 1 - \frac{2}{nk(2n - 3k - 1)} \sum_{i=1}^{n} \sum_{j \in V_k(i)} (\hat{r}(i,j) - k), \tag{4.10}$$

where $V_k(i)$ is the set of k nearest neighbors of point i in the embedding space, but no longer in the original space; $\hat{r}(i,j)$ is the rank of j in the neighborhood of point i in the original space.

4.3 Results and Discussion

In this section, we present the results of our experiments and discuss the implications of these findings.

Table 4.1: 1-NN results (Smaller values are better)

	Artificial Datasets				Natural Datasets			
	BSC	SR1	SR2	FM	COIL20	ORL	MIT-CBCL	Olivetti
Isomap	6.50%	15.30%	9.15%	**0.00%**	5.83%	11.25%	1.60%	18.25%
Isomap+ENG	13.10%	15.15%	8.70%	4.95%	7.36%	11.75%	1.65%	18.25%
LLE	**4.15%**	26.75%	26.45%	**0.00%**	7.43%	9.00%	1.70%	14.75%
HLLE	5.20%	**7.55%**	8.05%	0.10%	7.29%	25.75%	2.43%	20.50%
LE	5.05%	32.15%	32.25%	1.05%	10.35%	13.25%	1.99%	31.00%
LTSA	9.20%	11.90%	**7.55%**	0.15%	7.01%	25.75%	2.38%	37.25%
T-SNE	48.40%	49.10%	49.10%	51.15%	93.33%	98.00%	91.50%	98.00%
K-PCA	50.45%	26.75%	16.85%	**0.00%**	5.83%	**4.00%**	**1.41%**	**13.25%**
MVU	6.70%	13.35%	13.10%	**0.00%**	5.69%	10.75%	1.89%	14.50%
MVU+ENG	5.70%	10.60%	13.30%	17.55%	5.00%	6.25%	1.99%	14.50%
MVU-DM	4.85%	9.20%	9.75%	**0.00%**	**4.38%**	6.25%	1.94%	14.50%

Table 4.2: Trustworthiness results (Larger values are better)

	Artificial Datasets				Natural Datasets			
	BSC	SR1	SR2	FM	COIL20	ORL	MIT-CBCL	Olivetti
Isomap	99.18%	98.05%	99.76%	99.10%	99.09%	98.69%	99.67%	97.16%
Isomap+ENG	98.01%	98.12%	99.93%	97.27%	98.26%	98.46%	99.42%	97.16%
LLE	**99.53%**	94.81%	94.92%	99.17%	97.99%	95.86%	99.06%	91.16%
HLLE	98.85%	99.81%	99.95%	98.68%	97.91%	90.73%	99.06%	88.92%
LE	98.94%	93.76%	93.69%	99.72%	98.56%	98.20%	99.73%	94.03%
LTSA	97.58%	99.41%	**99.96%**	98.72%	96.98%	90.73%	99.20%	88.52%
T-SNE	50.10%	50.20%	50.10%	50.11%	50.70%	50.80%	50.40%	51.20%
K-PCA	92.90%	92.19%	89.41%	**100.00%**	**99.43%**	**99.37%**	**99.90%**	**98.45%**
MVU	97.99%	98.44%	97.32%	98.28%	97.86%	97.54%	99.33%	97.03%
MVU+ENG	98.30%	98.30%	98.50%	93.30%	97.90%	98.10%	99.32%	97.03%
MVU-DM	99.50%	**99.90%**	99.70%	99.05%	99.10%	98.10%	99.10%	97.03%

The results of the experiments are summarized in Tables 4.1 to 4.3. Each table is independent and present the best result from each method, on each dataset, for the respective metric. The user parameter k can take different values between the tables, as some tasks may consider single measures.

The best result on each dataset is highlighted in bold.

There is a visible and unexpected lead in performance of the KPCA algorithm in the natural datasets. However, PCA, as a linear method, on a fast analysis also performed well on these datasets. It is possible that they don't evaluate the full potential of the non-linear methods. This happening however, does not undermine the performance of the other methods, and the necessity to explore their capabilities further.

Relatively to the performance difference between ENG and regular methods, it seems that there was some improvement, but not overwhelmingly positive.

Table 4.3: Continuity results (Larger values are better)

	Artificial Datasets				Natural Datasets			
	BSC	SR1	SR2	FM	COIL20	ORL	MIT-CBCL	Olivetti
Isomap	**99.85%**	99.55%	99.91%	99.55%	**99.80%**	99.68%	99.87%	99.41%
Isomap+ENG	99.60%	99.55%	**99.95%**	99.60%	99.64%	99.66%	99.77%	99.37%
LLE	98.49%	99.40%	99.40%	98.68%	99.14%	97.30%	99.31%	92.47%
HLLE	95.70%	95.69%	95.80%	93.50%	98.76%	94.86%	98.60%	88.65%
LE	96.86%	99.40%	99.35%	80.30%	99.06%	99.16%	99.63%	96.80%
LTSA	87.55%	93.10%	95.59%	93.04%	99.11%	94.86%	98.52%	88.65%
T-SNE	50.40%	50.20%	50.20%	50.04%	49.50%	51.40%	49.70%	51.10%
K-PCA	99.28%	98.78%	98.03%	**100.00%**	**99.80%**	99.48%	**99.91%**	99.12%
MVU	99.24%	99.58%	99.76%	99.40%	99.73%	**99.71%**	99.82%	**99.59%**
MVU+ENG	99.60%	**99.90%**	99.90%	99.56%	99.70%	99.70%	99.83%	**99.59%**
MVU-DM	99.80%	**99.90%**	99.80%	99.54%	**99.80%**	99.70%	99.80%	99.54%

Table 4.4: Time speedup of MVU-DM compared to MVU for different values of k

	Artificial Datasets				Natural Datasets			
	BSC	SR1	SR2	FM	COIL20	ORL	MIT-CBCL	Olivetti
$k = 5$	6.69	3.24	2.43	6.10	4.02	1.15	7.96	0.55
$k = 10$	12.18	6.40	6.61	12.81	3.03	1.09	4.54	1.23
$k = 15$	11.13	4.14	5.67	15.94	1.59	1.09	2.48	1.39

When comparing the MVU and the proposed MVU-DM, it is evident that the latter consistently outperforms the former across most datasets and metrics. This improvement can be attributed to the ability of MVU-DM to handle the disconnected components individually, leading to better preservation of the local structure.

There was no evidence of subpar performance of the MVU-DM compared to the MVU. However, there is the possibility that the inter-component connections used are not representative enough of the global structure. For example, in the process shown in Figure 3.1, in the case of two components (intrinsically 2-dimensional), by using only a single connection between them, the two components are free to rotate around the vector described by the line crossing the mean points of both components. This rotational freedom can lead to a misrepresentation of the global structure. Note that this limitation is also present on the original MVU.

The additional reasoning for this problem not to surge is that the metrics used do not account for this type of distortion, as they are more focused on the local structure preservation. Future work could explore the use of additional metrics that capture global structure preservation to provide a more comprehensive evaluation of the global structure of the embeddings.

The main improvement from the vanilla MVU is best represented, however, in the time performance. As shown in Table 4.4, the MVU-DM achieves significant speedups compared to the MVU, especially

for smaller values of k, creating more disconnected components. This speedup is more pronounced on Synthetic datasets, where the sizes of disconnected components are more balanced. In natural datasets, there was a clear presence of a dominant large component, making the individual optimization tasks less balanced, and thus the speedup less pronounced.

On the MVU-DM python implementation, there is a section for launching a given number of matlab engines. On natural datasets, it was found that the data frequently forms a principal manifold; it was visible that while one engine was solving the largest component, a single additional engine was able to process all the other tasks. Thus, on natural datasets, the number of engines was limited to two. On synthetic datasets, however, with a more balanced distribution of points per component, the use of more engines is helpful; one engine per component was the ideal case.

Further studies could evaluate the impact of purposely creating more disconnected components, even if the data is not naturally structured that way. This could potentially lead to further speedups, although it may also impact the quality of the embeddings. Studies like [26] have already explored the idea of adaptively selecting the neighborhood size to create more balanced components, which could show promising results on this front.

Overall, the results demonstrate the effectiveness of the proposed MVU-DM algorithm in preserving the local structure of the data while significantly improving computational efficiency. These findings suggest that MVU-DM is a promising approach for dimensionality reduction, particularly in scenarios where the data may lie on multiple disconnected manifolds, and the size of the data is large.

5

Conclusion

This dissertation presented a novel approach to enhance the scalability of the Maximum Variance Unfolding (MVU) algorithm, a well-known technique for nonlinear dimensionality reduction. The proposed method, termed MVU-DM (Maximum Variance Unfolding with Disconnected Manifolds), addresses the computational challenges associated with MVU by decomposing the dataset into smaller, manageable components based on their connectivity in the original space. Each component is then independently embedded using MVU, and a global embedding is constructed by aligning these local embeddings through a set of representative points. This approach not only reduces the computational burden but also preserves the local structure of the data, which is crucial for effective dimensionality reduction.

The experimental results demonstrated that MVU-DM achieves comparable performance to the standard MVU algorithm in terms of classification accuracy, trustworthiness, and continuity, while significantly reducing the computational time required for embedding. The method was tested on both artificial and natural datasets, showcasing its versatility and effectiveness across different types of data. Notably, MVU-DM outperformed other traditional dimensionality reduction techniques such as Isomap, LLE, HLLE, LE, LTSA, T-SNE, and K-PCA in several scenarios, particularly in preserving local structures and achieving lower classification errors.

Bibliography

[1] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics and intelligent laboratory systems*, vol. 2, no. 1-3, pp. 37–52, 1987.

[2] W. S. Torgerson, "Multidimensional scaling: I. theory and method," *Psychometrika*, vol. 17, no. 4, pp. 401–419, 1952.

[3] J. B. Tenenbaum, V. d. Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

[4] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[5] M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural computation*, vol. 15, no. 6, pp. 1373–1396, 2003.

[6] K. Q. Weinberger and L. K. Saul, "Unsupervised learning of image manifolds by semidefinite programming," *International journal of computer vision*, vol. 70, pp. 77–90, 2006.

[7] B. Borchers and J. G. Young, "Implementation of a primal–dual method for sdp on a shared memory parallel architecture," *Computational Optimization and Applications*, vol. 37, pp. 355–369, 2007.

[8] J. Fan, T. W. Chow, M. Zhao, and J. K. Ho, "Nonlinear dimensionality reduction for data with disconnected neighborhood graph," *Neural Processing Letters*, vol. 47, pp. 697–716, 2018.

[9] J. B. Kruskal, "Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis," *Psychometrika*, vol. 29, no. 1, pp. 1–27, 1964.

[10] D. L. Donoho and C. Grimes, "Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data," *Proceedings of the National Academy of Sciences*, vol. 100, no. 10, pp. 5591–5596, 2003.

[11] Z. Zhang and H. Zha, "Principal manifolds and nonlinear dimensionality reduction via tangent space alignment," *SIAM journal on scientific computing*, vol. 26, no. 1, pp. 313–338, 2004.

[12] G. E. Hinton and S. Roweis, “Stochastic neighbor embedding,” *Advances in neural information processing systems*, vol. 15, 2002.

[13] L. Van der Maaten and G. Hinton, “Visualizing data using t-sne.” *Journal of machine learning research*, vol. 9, no. 11, 2008.

[14] B. Schölkopf, A. Smola, and K.-R. Müller, “Nonlinear component analysis as a kernel eigenvalue problem,” *Neural computation*, vol. 10, no. 5, pp. 1299–1319, 1998.

[15] C. Williams and M. Seeger, “Using the nyström method to speed up kernel machines,” *Advances in neural information processing systems*, vol. 13, 2000.

[16] M. Vladymyrov and M. Á. Carreira-Perpinán, “Locally linear landmarks for large-scale manifold learning,” in *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2013, Prague, Czech Republic, September 23-27, 2013, Proceedings, Part III 13*. Springer, 2013, pp. 256–271.

[17] K. Weinberger, B. Packer, and L. Saul, “Nonlinear dimensionality reduction by semidefinite programming and kernel matrix factorization,” in *International Workshop on Artificial Intelligence and Statistics*. PMLR, 2005, pp. 381–388.

[18] V. Silva and J. Tenenbaum, “Global versus local methods in nonlinear dimensionality reduction,” *Advances in neural information processing systems*, vol. 15, 2002.

[19] B. Ghojogh, A. Ghodsi, F. Karray, and M. Crowley, “Locally linear embedding and its variants: Tutorial and survey,” *arXiv preprint arXiv:2011.10925*, 2020.

[20] E. Levina and P. J. Bickel, “Maximum likelihood estimation of intrinsic dimension,” *Advances in neural information processing systems*, vol. 17, 2004.

[21] L. Song, A. Gretton, K. Borgwardt, and A. Smola, “Colored maximum variance unfolding,” *Advances in neural information processing systems*, vol. 20, 2007.

[22] L. Van Der Maaten, E. O. Postma, H. J. Van Den Herik *et al.*, “Dimensionality reduction: A comparative review,” *Journal of machine learning research*, vol. 10, no. 66-71, p. 13, 2009.

[23] G. van Rossum, Python, a general-purpose programming language.

[24] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[25] I. The MathWorks, “Calling matlab from python,” 2024, https://www.mathworks.com/help/matlab/matlab-engine-for-python.html.

[26] P. Chen, L. Jiao, F. Liu, Z. Zhao, and J. Zhao, “Adaptive sparse graph learning based dimensionality reduction for classification,” *Applied Soft Computing*, vol. 82, p. 105459, 2019.

A

Appendix

A.1 Convex MVU

The convex version of MVU is derived from the original formulation:

$$\max_{\boldsymbol{y}_1,\dots,\boldsymbol{y}_N} \quad \sum_{i=1}^{N} \|\boldsymbol{y}_i\|_2^2 \tag{A.1}$$

$$\text{s.t.} \quad \|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^2 = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2, \quad i \sim j \tag{A.2}$$

$$\sum_{i=1}^{N} \boldsymbol{y}_i = \mathbf{0}. \tag{A.3}$$

Expanding the squared terms:

$$\max_{\boldsymbol{y}_1,\dots,\boldsymbol{y}_n} \quad \sum_{i=1}^{n} \boldsymbol{y}_i^\top \boldsymbol{y}_i \tag{A.4}$$

$$\text{s.t.} \quad \boldsymbol{y}_i^\top \boldsymbol{y}_i - 2\boldsymbol{y}_i^\top \boldsymbol{y}_j + \boldsymbol{y}_j^\top \boldsymbol{y}_j = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2, \quad i \sim j \tag{A.5}$$

$$\sum_{i,j=1}^{n} \boldsymbol{y}_i^\top \boldsymbol{y}_j = 0 \tag{A.6}$$

Collecting the embedded points into the matrix $\boldsymbol{Y} \in \mathbb{R}^{n \times d}$, the problem can be rewritten as:

$$\begin{aligned}
\max_{\boldsymbol{Y}} \quad & \text{tr}(\boldsymbol{Y}^\top \boldsymbol{Y}) && \text{(A.7)} \\
\text{s.t.} \quad & \boldsymbol{e}_i^\top \boldsymbol{Y}^\top \boldsymbol{Y} \boldsymbol{e}_i - 2\boldsymbol{e}_i^\top \boldsymbol{Y}^\top \boldsymbol{Y} \boldsymbol{e}_j + \boldsymbol{e}_j^\top \boldsymbol{Y}^\top \boldsymbol{Y} \boldsymbol{e}_j = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2, \quad i \sim j && \text{(A.8)} \\
& \mathbf{1}^\top \boldsymbol{Y}^\top \boldsymbol{Y} \mathbf{1} = 0, && \text{(A.9)}
\end{aligned}$$

where $\boldsymbol{e}_i$ is a "selection" or "one-hot" vector, i.e., its elements are all zero except for the element at the i-th index, which is 1. Note that, now, the whole problem depends on $\boldsymbol{Y}^\top \boldsymbol{Y}$, so we introduce a new variable $\boldsymbol{K} = \boldsymbol{Y}^\top \boldsymbol{Y} \in \mathbb{R}^{N \times N}$ to linearize the terms that depend on $\boldsymbol{Y}^\top \boldsymbol{Y}$:

$$\begin{aligned}
\max_{\boldsymbol{K}, \boldsymbol{Y}} \quad & \text{tr}(\boldsymbol{K}) && \text{(A.10)} \\
\text{s.t.} \quad & \boldsymbol{e}_i^\top \boldsymbol{K} \boldsymbol{e}_i - 2\boldsymbol{e}_i^\top \boldsymbol{K} \boldsymbol{e}_j + \boldsymbol{e}_j^\top \boldsymbol{K} \boldsymbol{e}_j = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2, \quad i \sim j && \text{(A.11)} \\
& \mathbf{1}^\top \boldsymbol{K} \mathbf{1} = 0 && \text{(A.12)} \\
& \boldsymbol{K} = \boldsymbol{Y}^\top \boldsymbol{Y} && \text{(A.13)}
\end{aligned}$$

Because $\boldsymbol{K} = \boldsymbol{Y}^\top \boldsymbol{Y}$ defines an inner product matrix, we can replace that constraint if we make sure that $\boldsymbol{K}$ is both symmetric, positive semidefinite and that the rank of $\boldsymbol{K}$ is not greater than the dimension of the $\boldsymbol{y}_i$s:

$$\begin{aligned}
\max_{\boldsymbol{K}} \quad & \text{tr}(\boldsymbol{K}) && \text{(A.14)} \\
\text{s.t.} \quad & \boldsymbol{e}_i^\top \boldsymbol{K} \boldsymbol{e}_i - 2\boldsymbol{e}_i^\top \boldsymbol{K} \boldsymbol{e}_j + \boldsymbol{e}_j^\top \boldsymbol{K} \boldsymbol{e}_j = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2, \quad i \sim j && \text{(A.15)} \\
& \mathbf{1}^\top \boldsymbol{K} \mathbf{1} = 0 && \text{(A.16)} \\
& \boldsymbol{K} \succeq 0 && \text{(A.17)} \\
& \text{rk}(\boldsymbol{K}) \leq n && \text{(A.18)}
\end{aligned}$$

Now, the only nonconvexity in the problem is given by the rank constraint. If we remove it, we arrive at a convex relaxation of the original problem, which is the final formulation of MVU:

$$\max_{\boldsymbol{K}} \quad \text{tr}(\boldsymbol{K}) \tag{A.19}$$

$$\text{s.t.} \quad \boldsymbol{e}_i^\top \boldsymbol{K} \boldsymbol{e}_i - 2\boldsymbol{e}_i^\top \boldsymbol{K} \boldsymbol{e}_j + \boldsymbol{e}_j^\top \boldsymbol{K} \boldsymbol{e}_j = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2, \quad i \sim j \tag{A.20}$$

$$\mathbf{1}^\top \boldsymbol{G} \mathbf{1} = 0 \tag{A.21}$$

$$\boldsymbol{K} \succeq 0 \tag{A.22}$$

Here, $\boldsymbol{K} \in \mathbb{R}^{N \times N}$ is a Gramian (or inner product) matrix, so that maximizing its trace corresponds to maximizing the variance of the data in the target space.