SCALABLE NONLINEAR SPECTRAL DIMENSIONALITY REDUCTION METHODS FOR STREAMING DATA

by

Suchismit Mahapatra
May 2018

A dissertation submitted to the
Faculty of the Graduate School of
the University at Buffalo, State University of New York
in partial fulfilment of the requirements for the
degree of

Doctor of Philosophy

Department of Computer Science and Engineering

Committee members:

Varun Chandola, Committee Chair/Advisor

Nils Napp, Committee Member

Jaroslaw Zola, Committee Member

Dedication

To Maa, who inspires me every moment of my existence to forge ahead, even in the presence of obstacles.

To Bholu, my soulmate and the anchor to my madness.

To Jeje, whose bedtime stories gave flight to my imagination.

Acknowledgments

First and foremost, I would like to thank my advisor Dr Varun Chandola who has been more of a mentor to me at UB. He gave me the utmost freedom to choose my own battles i.e. decide upon my own research directions which allowed me to work in diverse fields and played a huge role in my well-rounded development as a researcher, for which I will forever be indebted. I would also like to thank him for his patience to allow me to make my own mistakes and learn from them and always supporting my incessant passion to learn. He has had the biggest influence on my research career. I would also like to thank my committee members Dr Jaroslaw Zola and Dr Nils Napp for their valuable input to improve this work. I would also like to thank the multiple professors at the Department of Computer Science and Engineering of UB whose courses I took. Each of you have had an impact on me being the person I am today and for which I will forever be grateful. In particular, I would like to thank Dr Bina Ramamurthy and Dr Xin He who have mentored me at various stages of my stint here at UB. I would like to acknowledge the National Science Foundation for funding this research under award number IIS - 1651475. Additionally I would like to acknowledge the Center for Computational Research for providing computing support.

Finally, I would like to thank my parents and my grandparents and in particular, my wife for their love and support, without whom, this work would not be possible.

Table of Contents

List of Tables

List of Figures

Abstract

High-dimensional data is inherently difficult to explore and analyze owing to the "curse of dimensionality" that render many statistical and Machine Learning (ML) techniques (e.g. clustering, classification, model fitting, etc.) inadequate. In this context, nonlinear spectral dimensionality reduction (NLSDR) methods have proved to be an indispensable tool. However, standard NLSDR methods, e.g. Isomap [74] or Locally Linear Embedding (LLE) [60], have been designed for off-line or batch processing. Consequently, they are computationally too expensive or impractical in cases where dimensionality reduction must be applied on a data stream. Processing data streams efficiently using standard approaches is also challenging in general, given streams require real-time processing and cannot be stored permanently. Any form of analysis, including NLSDR and/or detecting *concept-drift* requires adequate summarization which can deal with the inherent constraints and that can approximate the characteristics of the stream well. In spite of advances in hardware and development of novel processing frameworks, the issue of scalability of ML algorithms still remains. The scalability of an algorithm is measured via how its performance gets affected as the problem size increases. Scalable algorithms should be able to work with any amount of data without consuming ever growing amounts of storage memory and computations. The challenge is often to find a trade-off between quality and processing time i.e. getting "good enough" solutions as "fast" or "efficiently" as possible.

In this thesis, I propose a generalized framework for streaming NLSDR which can work with different manifold learning approaches e.g. Isomap and LLE to be able to deal effectively with data streams, having underlying distributions which can be multi-modal in nature and be non-uniformally sampled as well. In particular, I developed streaming Isomap or S-Isomap [65], an algorithm which via a clever approximation is able to scalably reduce the computation cost of discovering the low-dimensional embedding at a fraction of the cost without affecting the quality significantly. However, S-Isomap [65] was limited in this

scope i.e. it could only deal with unimodal, uniformly sampled distributions. Hence arose the need for S-Isomap++ [50], which ameliorated the flaws of its predecessor in being able to deal with multimodal and/or unevenly sampled distributions.

However, S-Isomap++ [50] can only detect manifolds which it encounters in its batch learning phase and not those which it might encounter in the streaming phase. Thus, S-Isomap++ [50] ceases to "learn" and evolve to be able to limit the embedding error for points in the data stream, which motivated the need for GP-Isomap [51], which via a novel positive-definite geodesic-distance based kernel, and using Gaussian Processes to measure variance, is able to detect *concept-drift* i.e. distinguish among different manifolds and embed streaming samples effectively. Subsequently, I developed the streaming LLE algorithm, for processing streams using LLE as well as discuss a generalized Out-of-Sample Extension methodology for streaming NLSDR, applicable for different manifold learning algorithms. Lastly, I provide theoretical bounds for S-Isomap and GP-Isomap as part of this work.

Chapter 1

Introduction

Ability to analyze massive streams of data is a valuable aspect of any modern data science pipeline. This is important in many contexts, such as high-performance high-fidelity numerical simulations [18], high-resolution scientific instrumentation (microscopes, DNA sequencers, etc.) [61] and even *Internet of Things* [10] where a huge number of devices are currently connected to the Internet and feeding a variety of data streams. Such data sources typically monitor or measure complex system behaviors, using a large number of parameters. Dimensionality reduction methods [77] are typically used to map the resulting high-dimensional data into a smaller, manageable space, while losing as little information as possible. The underlying assumption is that the dataset can be well-described by a set of features, whose number is significantly smaller than the dimension of the original. Thus aim is to find this set of features, thereby recovering the true structure of the data.

Historically, the preferred approach for dimensionality reduction was the linear one. Perhaps the simplest being feature elimination [38]. In feature elimination, attempts are made to isolate the features relevant to the problem, subsequent to which the "non-relevant" features are removed. However, the assumption that only a few of the features are relevant is a strong one and is usually incorrect. Yet another common approach performs linear projection of the data onto a subspace of lower dimensionality. In this approach, the assumption is

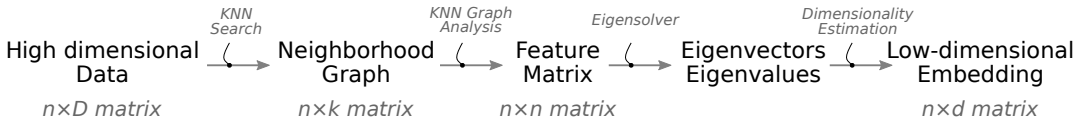


Figure 1.1: General nonlinear spectral dimensionality reduction workflow.

that data is usually controlled by a few underlying latent variables which are related via a linear mapping. Common techniques include Principal Component Analysis (PCA) [31], Multidimensional Scaling (MDS) [41], Factor Analysis [52] and Independent Component Analysis (ICA) [33] among others. While these techniques are largely successful, the assumption that a linear projection describes the data well is incorrect for many scenarios. In many settings, especially when dealing with complex scientific and natural phenomenon, when the data usually lies on a nonlinear manifold, Nonlinear Spectral Dimensionality Reduction (NLSDR) methods are more appropriate. In practical terms, NLSDR effectively brings the original data into a more human-intuitive low dimensional space that enables visualization and makes quantitative and qualitative analysis of nonlinear processes possible.

1.1 Nonlinear Spectral Dimensionality Reduction

In the most general terms, the NLSDR problem can be posed as follows. Given a data matrix $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n]^\top$, such that each $\mathbf{x}_i \in \mathbb{R}^{\mathbf{D}}$, the task is to find a corresponding low-dimensional representation, $\mathbf{y}_i \in \mathbb{R}^{\mathbf{d}}$, for each $\mathbf{x}_i$, where $\mathbf{d} \ll \mathbf{D}$. We assume that there exists a function $\phi : \mathbb{R}^{\mathbf{d}} \to \mathbb{R}^{\mathbf{D}}$ that maps each data sample $\mathbf{y}_i \in \mathbb{R}^{\mathbf{d}}$ to $\mathbf{x}_i \in \mathbb{R}^{\mathbf{D}}$. The goal of NLSDR is to learn the inverse mapping, ϕ^{-1}, that can be used to map high-dimensional $\mathbf{x}_i$ to low-dimensional $\mathbf{y}_i$, i.e. $\mathbf{y}_i = \phi^{-1}(\mathbf{x}_i)$.

However NLSDR techniques come at a cost; most existing NLSDR methods have a computational complexity of $\mathcal{O}(\mathbf{n^3})$, $\mathbf{n}$ being the size of the data. The issue is further exacerbated when the data is streaming, where obtaining exact solution at every step of the stream is computationally infeasible. While adapatations of existing NLSDR methods, such as Isomap [74] and LLE [60], have been proposed for handling data streams [40, 43], such methods, which typi-

cally rely on incremental updates of the underlying solution, do not scale well to massive streams.

Typically, NLSDR techniques are used as learning methods for discovering the underlying low-dimensional structure from samples from high-dimensional data. Existing techniques typically exploit either the global (Isomap, Minimum Volume Embedding [83]) or local (LLE, Laplacian Eigenmaps [4]) properties of the manifold to map each high-dimensional point $\mathbf{x}_i \in \mathbb{R}^{\mathbf{D}}$ to its corresponding low-dimensional embedding, $\mathbf{y}_i \in \mathbb{R}^{\mathbf{d}}$. They are used as a generic nonlinear, non-parametric technique to approximate probability distributions in high-dimensional spaces.

Most existing NLSDR techniques, perform a similar series of data transformations as shown in Figure 1.1. First, a neighborhood graph is constructed, where each node of the graph is connected to its $\mathbf{k}$ nearest neighbors. This involves computing $\mathcal{O}(\mathbf{n^2})$ pairwise distance values. Next, a feature matrix is computed from this neighborhood graph, which encodes properties of the data that should be preserved during dimensionality reduction. For example, in the Isomap formulation, the feature matrix stores shortest paths between each pair of points in the neighborhood graph, which is an approximation of the actual geodesic distance between the points. The cost to compute the feature matrix generally varies in the range $\mathcal{O}(\mathbf{n})$ and $\mathcal{O}(\mathbf{n^3})$. To obtain the low representation of the input data, the feature matrix is factorized and the first $\mathbf{d}$ eigen vectors/values form the output $\mathbf{Y}$. This step has a $\mathcal{O}(\mathbf{n^3})$ cost.

When used on data streams, NLSDR methods typically have to recompute the entire manifold for every new streaming data point, which is computationally expensive. In such scenarios, there is the need for incremental techniques (Out-of-Sample Extension technique [43]), which can process the new streaming points "cheaply", compared to the traditional batch techniques without affecting the quality of the embedding significantly.

Various NLSDR techniques have been successfully applied in many critical applications ranging from advanced multimedia and scientific imagery analysis [32, 47, 55, 73, 75, 58], through molecular modeling [78] to computational biology [46, 61], and they are among the most fundamental methods for both data preprocessing and analytics [45, 72]. However, many nonlinear processes

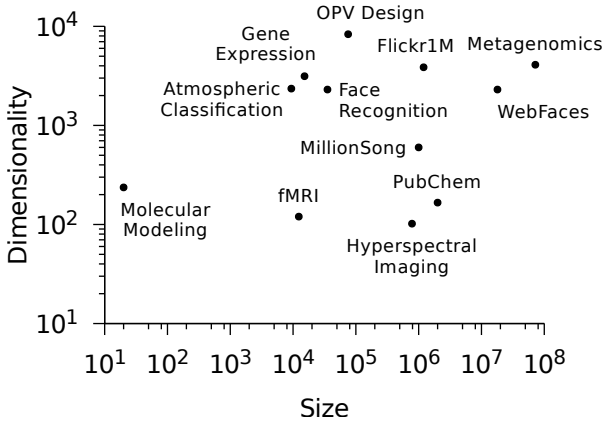


Figure 1.2: Topology of high-dimensional, massive datasets.

of interest that could be studied via NLSDR produce high volume and high velocity data that exceeds capability of the existing NLSDR methods. This is because these techniques scale poorly with the number of available data points and dimensions, and are primarily tailored for batch processing [77]. In Fig. 1.2, we summarized properties of several representative data collections to which NLSDR is directly applicable. These data sets typically consist of millions of points in hundreds and thousands of dimensions. By contrast, the largest data sets analyzed by classic NLSDR approaches usually do not exceed a few thousands points [32, 39, 46, 53]. The scalability challenges behind NLSDR have been recognized by the research community [72]. Consequently, several approximation techniques [22, 42, 43, 73] and parallel computing approaches [2, 8, 13] have been proposed. These, however, remain computationally challenging, address only some NLSDR techniques, and lack a systematic and generalizable approach that would be applicable to a broader spectrum of NLSDR methods.

1.2 Thesis Outline

The remaining of the proposal is structured as follows. In each of these chapters, the content is arranged in sections with introduction, description of the methodology, experiments and results, related work, discussion and conclusion as applicable. In chapter 3, I present S-Isomap [65], an Out-of-Sample Extension algorithm which can adequately map streaming data after NLSDR is performed

on an initial batch dataset. However, S-Isomap could only handle unimodal, uniformly sampled distributions. Hence arose the need for S-Isomap++ [50], which is presented in chapter 4. S-Isomap++ [50] ameliorated the flaws of its predecessor in being to deal with multimodal and/or unevenly sampled distributions. However, S-Isomap++ could only detect manifolds encountered in its batch learning phase and not those which it might encounter subsequently i.e. S-Isomap++ ceases to *learn* and *evolve* to be able to adequately map the data stream, which motivated the need for GP-Isomap [51], which is presented in chapter 5. GP-Isomap [51] via a novel geodesic-distance based kernel, and using Gaussian Processes, is able to detect *concept-drift* i.e. distinguish between different manifolds in the streaming data. Subsequently in chapter 6, I discuss a generalized Out-of-Sample Extension framework for streaming NLSDR, applicable for different manifold learning algorithms as well as present a specific instantiation of the framework, the Streaming-LLE algorithm, for stream processing using LLE. In chapter 7, I conclude this thesis with a brief discussion and talk about possible future research directions for the current line of work.

Chapter 2

Background

2.1 Definition of a Manifold

Mathematically, a manifold $\mathcal{M}$ is defined as a metric space with the following property: if $\mathbf{x} \in \mathcal{M}$, then there exists some neighborhood $\mathcal{U}$ of $\mathbf{x}$ and $\exists \mathbf{n}$ such that $\mathcal{U}$ is homeomorphic to $\mathbb{R}^{\mathbf{n}}$ [70].

The global structure of the high-dimensional ambient space can be more complicated. Usually manifolds are embedded in high-dimensional spaces, but the intrinsic dimensionality is typically low due to fewer degrees of freedom in the underlying data generating process.

2.2 Why Manifold Learning for Nonlinear Dimensionality Reduction ?

Nonlinear Dimensionality Reduction (NLDR) techniques can be broadly divided into two main categories :-

- Approaches motivated by the geometry/topology of data i.e. Isomap [74], LLE [60], LTSA [90]
- Approaches based on Deep learning i.e. Autoencoders and its variants [62, 29, 28, 5, 37, 59]

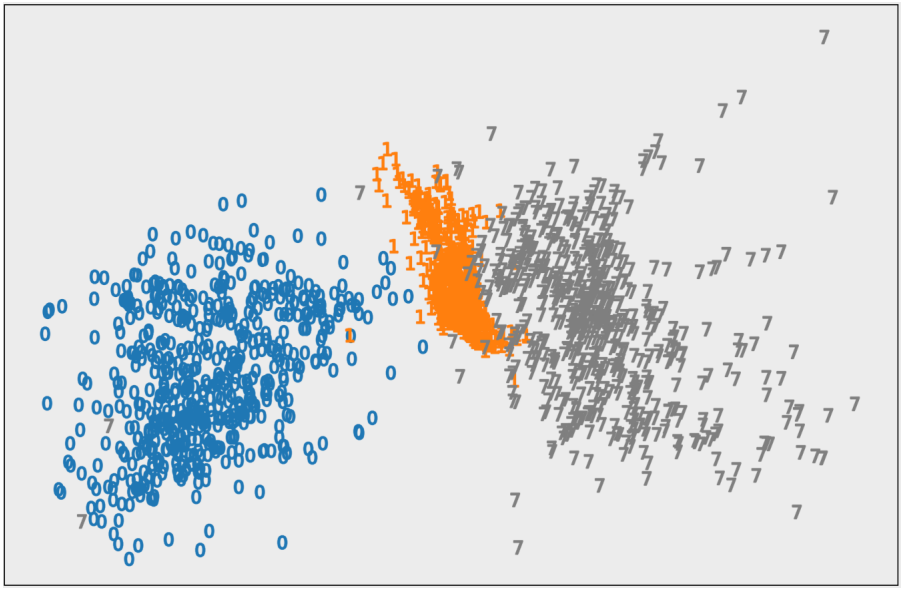


Figure 2.1: Real-world data generally lies near multiple manifolds and is usually separated by regions of low density.

The former category of methods exploit geometric/topological properties of the underlying manifold due to which their outputs are easier to understand. On the other hand, autoencoder-based approaches are not readily interpretable given that the relationship between the inputs and outputs is not transparent. Once the topological property has been defined, the former set of methods try to preserve it and due of this they have fewer free parameters. For auto-encoder based methods, hyper-parameter tuning and network design are challenging since they typically have a large number of parameters.

At a primordial level, autoencoder-based approaches do not make any assumptions regarding the mapping function[1] ϕ^{-1} they are trying to learn. From a theoretical perspective, the complexity of functions that are learnable by these networks increases exponentially with the depth of the network architecture. While this lack of bias might seem appealing however there are drawbacks to this as well. Local minima can severely affect these approaches since they depend a lot on the architecture as well as choice of hyper-parameters used.

[1] Assuming an underlying low-dimensional ground truth which gets mapped to higher dimensional real-world datasets via nonlinear ϕ. ϕ^{-1} refers to the "dimension-reducing" function which does the inverse mapping. Refer to Sections 3.3.1 and 6.1 for additional discussions.

The expressiveness prevalent in neural networks can also lead them to overfit if used without adequate regularization. These methods generally require a large amount of data to train as well as have long training periods. On the other hand, approaches in the former category make "reasonable" assumptions about local/global neighborhood relationships prevalent in the data which makes them more "biased" and hence less expressive. However it also makes it more easier for them to learn the mapping function as well as the improves the interpretability of the resultant embeddings uncovered. The issue of lack of scalability of certain algorithms in this family is adequately handled by the Out-of-Sample Extension based approaches specific to them. Consequently the time it takes for model learning as well as for stream processing is short.

We note here that it is possible to process data streams using autoencoder-based approaches. However since data streams can be potentially infinite, we cannot possibly hope to use entire data streams as training data for these neural networks and consequently, issues like *concept-drift* can severely dent the quality of embeddings uncovered for the test data. However, autoencoder-based methods would be useful in situations where the geometric/topological assumptions about the manifold do not hold. In this work, we assume that these assumptions in general hold and the results of our proposed approaches on several synthetic and real benchmark data sets demonstrate that this assumption is not unrealistic. Apart from the approaches mentioned, there are other techniques, for example, t-SNE [49] which falls somewhere in between, however such methods are an exception. Our discussion here was primarily motivated to weigh the pros and cons of the two main NLDR approaches. We note here that Kernel PCA belongs to the former category, given MDS which Isomap uses internally was shown to be a form of Kernel PCA [84].

We demonstrate results for the state-of-the-art t-SNE algorithm on the *Euler Isometric Swiss Roll* data set in Chapter 4 (see Figure 4.2), given its popularity for the visualization of high-dimensional datasets particularly in scenarios when the underlying manifold dimensionality ($\mathbf{d}$) is $\leq \mathbf{3}$. Even though t-SNE is not a "streaming" algorithm as such, our motivation was to demonstrate its capability in handling multiple manifolds. We show that t-SNE does not quite capture the low-dimensional structure of the underlying manifold in our experiments.

2.3 Quality of Embedding

Given the low-dimensional embedding by a NLSDR algorithm, we need to be able to evaluate how good is the uncovered representation so as to be able to utilize the embedding. If the quality of the embedding is poor, one cannot infer that the embedding is indeed a parametrization of the high-dimensional input. Moreover, if there is no way to estimate the quality of the uncovered embedding, we would not be able to compare the representations found by different algorithms. Finally, knowing the quality of embedding can be useful while parameter selection/optimization involved by the NLSDR algorithm.

A variety of embedding quality metrics [79, 11, 23] have been suggested, however for this work we use the *Procrustes analysis*, which is widely used for shape analysis [17] given its applicability to our problem. The idea behind Procrustes analysis is to align two matrices, $\mathbf{A}$ and $\mathbf{B}$, by finding the optimal translation $\mathbf{t}$, rotation $\mathbf{R}$, and scaling $\mathbf{s}$ that minimizes the Frobenius norm between the two aligned matrices. In Section 3.3, we discuss more on this as well as suggest a new measure of embedding quality, the Reference Sample method which is applicable in scenarios wherein we do not have the latent ground truth to compare against.

2.4 Isomap

In this section, we define the Isomap algorithm given its centrality to this thesis. In our approach, we focus on Isomap because of its broad adoption in scientific computing and bio-medical research, including fMRI analysis [75], clustering of oncology data [54], genes and proteins expression profiling [44], modeling of spatio-temporal relationships in data [36], and many others.

Isomap provides a simple and elegant way to estimate the intrinsic geometry of the manifold based on the local neighborhood of different points on the manifold. Unfortunately, its high computational complexity means it is too expensive to use on any but relatively small data sets, and it is not suited for stream processing. Consequently, there is a significant demand for an Isomap variant that would be capable to learn a robust manifold from high-throughput

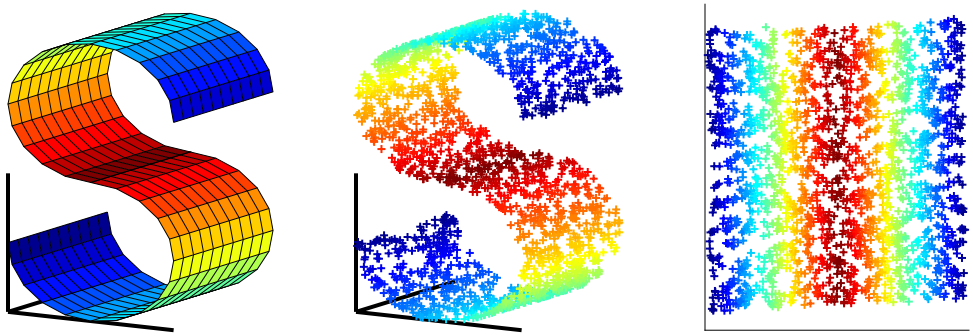


Figure 2.2: Illustration of Isomap on the *S-Curve* dataset. The goal of Isomap is to preserve the intrinsic geometry of the data.

data streams.

The goal of the Isomap algorithm is to preserve the intrinsic geometry of the data, as captured by the geodesic distances (i.e. the length of the shortest path on the manifold) between all pairs of data points. The geodesic distance between close points is approximated by the Euclidian distance, while the geodesic distance between faraway points is approximated by adding up a sequence of short hops between neighboring points. Note that this method attempts to preserve distances between both faraway and neighboring points, i.e. it is global.

The algorithm consists of three steps:

1. Construct the neighborhood graph by assigning to each data instance its nearest neighbors.

2. Compute shortest paths on the neighborhood graph to create the Geodesic Distance matrix.

3. Construct an embedding of the data in $\mathbb{R}^d$ by applying classical MDS to the Geodesic Distance matrix.

The Isomap algorithm, being a global NLSDR technique should ideally provide a more faithful representation and preserve geometry irrespective of scale i.e. map data samples which are close in the manifold to points which are close in the low-dimensional embedding and similarly for distant samples. However, it struggles when dealing with multi-modal and non-uniform distributions.

NLSDR techniques, irrespective of whether they are global or local, require an appropriate choice of the neighborhood size (**k**) to define their local neighborhood to perform appropriately. More specifically, a small neighborhood size might not to capture sufficient information about the manifold geometry, more so when it is smaller than the intrinsic dimensionality of the manifold. A large neighborhood size could lead to issues i.e. short-circuiting when distant, far-off get included in the local neighborhood and corrupt the same. Idiosyncrasies i.e. curvature and uneven sampling in different regions of the manifold make choosing the problem of choosing the local neighborhood size particularly difficult. The Isomap algorithm is also not immune to the above. When the sampling or distribution of data is multi-modal and non-uniform, we are forced to choose a large **k** value so that we can formulate the Geodesic distance matrix for all the data points, which would enable the Isomap algorithm to perform dimensionality reduction. In such scenarios, short-circuiting results in improper embeddings uncovered by Isomap, which typically needs smaller values of **k** to appropriately represent the local neighborhood.

This particular issue can be seen in Figure 4.1 where both the M-Isomap [20] and S-Isomap++ algorithms can deal with individual manifolds well compared to Isomap which severely deforms the individual clusters, which are $\mathbb{R}^{\mathbf{2}}$ Gaussian distributions mapped to $\mathbb{R}^{\mathbf{3}}$ via the Isometric technique suggested by Schoeneman *et al.* [65]. It should also be noted that whereas both the M-Isomap and S-Isomap++ algorithms required small values of **k** i.e. $\mathbf{k} = \mathbf{8}$ to operate, Isomap needed values of $\mathbf{k} \geq \mathbf{500}$ to even work.

The convergence properties of Isomap were studied by Bernstein *et al.* [7] who proved that under some assumptions based on the manifold and the sampling density, the estimated distances converge to the real geodesic distances. This result, together with the robustness of MDS to perturbations [67] assure convergence of Isomap when the assumptions are fulfilled.

2.5 LLE

The LLE algorithm, being a local NLSDR technique, is primarily concerned with preserving the local neighborhood relationships between points. It does this

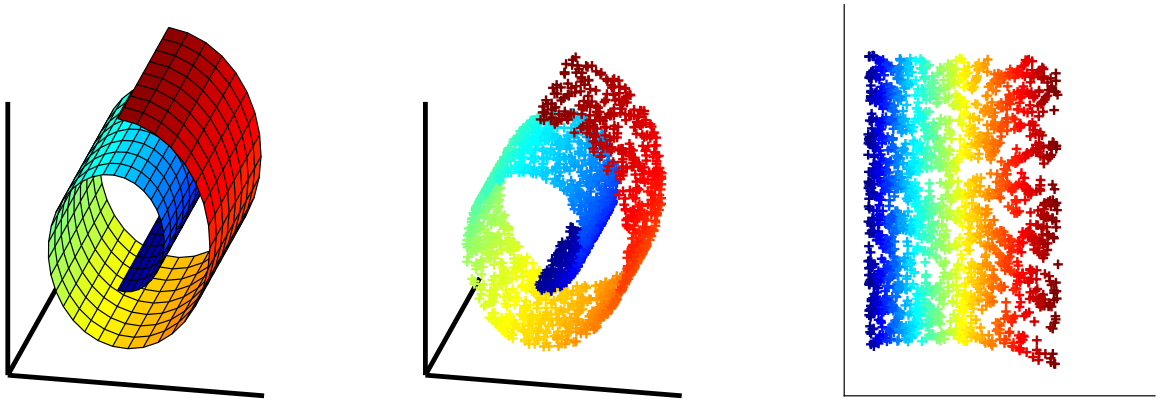


Figure 2.3: Illustration of Isomap on the *Swiss Roll* dataset.

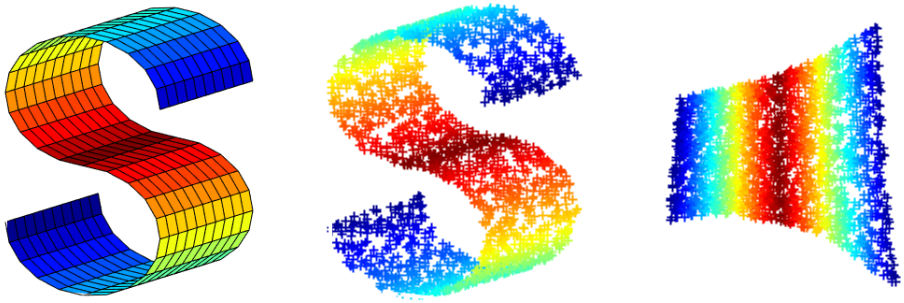


Figure 2.4: Illustration of LLE on the *S-Curve* dataset.

by defining the neighborhood for all points by determining the nearest neighbors for points and subsequently choosing an low-dimensional representation which best preserves the locally linear relationships. Let $\mathbf{X} = \{\mathbf{x}_i \in \mathbb{R}^{\mathbf{D}}\}_{i=1,2,\ldots n}$ be samples belonging to a compact manifold $\mathcal{M}$ of intrinsic dimensionality $\mathbf{d}$, where $\mathbf{d} \ll \mathbf{D}$. LLE initially tries to represent each $\mathbf{x}_i$ as a linear weighted combination of the set of points defined using in its local neighborhood $\mathcal{N}(\mathbf{x}_i)$. Subsequently, it tries to determine a set of low-dimensional representations $\mathbf{Y} = \{\mathbf{y}_i \in \mathbb{R}^{\mathbf{d}}\}_{i=1,2,\ldots n}$ which preserves the different local relationships using the same combinations of weights.

The LLE algorithm has three main steps :-

1. Define a local neighborhood for each $\mathbf{x}_i \in \mathbf{X}$ either using the ϵ-rule or the $\mathbf{K}$-rule.

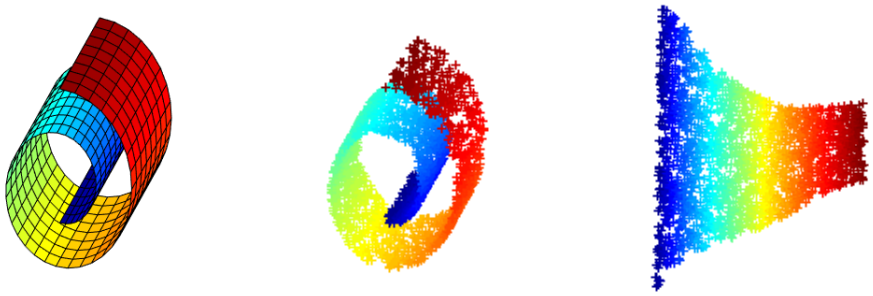


Figure 2.5: Illustration of LLE on the *Swiss Roll* dataset.

2. Determine $\mathbf{w}$ which minimizes $\mathbf{\Phi}(\mathbf{w}) = \left\| \left(\mathbf{x}_i - \sum_{\mathbf{x}_j \in \boldsymbol{\mathcal{N}}(\mathbf{x}_i)} \mathbf{w}_{i,j} \mathbf{x}_j \right) \right\|^2$ i.e.
   $\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmin}}\, \mathbf{\Phi}(\mathbf{w})$
   such that $\forall \mathbf{i} \in \{1, 2 \ldots \mathbf{n}\}, \sum_j \mathbf{w}_{i,j} = 1$ and $\mathbf{w}_{i,j} = 0$ if $\mathbf{x}_j \notin \boldsymbol{\mathcal{N}}(\mathbf{x}_i)$.

3. Compute $\mathbf{Y}$ which can be best reconstructed by $\mathbf{w}^*$ by minimizing $\mathbf{\Lambda}(\mathbf{y}_1, \mathbf{y}_2 \ldots \mathbf{y}_n)$
   where $\mathbf{\Lambda}(\mathbf{y}_1, \mathbf{y}_2 \ldots \mathbf{y}_n) = \sum_{i=1}^{n} \left\| \left(\mathbf{y}_i - \sum_{\mathbf{y}_j \in \boldsymbol{\mathcal{N}}(\mathbf{y}_i)} \mathbf{w}^*_{i,j} \mathbf{y}_j \right) \right\|^2$.

The most computationally expensive part of the the LLE algorithm is the last step i.e. minimization of $\mathbf{\Lambda}(\mathbf{y}_1, \mathbf{y}_2 \ldots \mathbf{y}_n)$ which requires $\mathcal{O}(\mathbf{n^2})$ operations.

Chapter 3

Streaming Isomap or S-Isomap

3.1 Introduction

In this chapter, we try to address the issue of computational costs of classic NLSDR methods and propose an alternative solution that builds on the systematic study of errors in manifold learning. We first describe a protocol to capture a "collective error" of the Isomap method. Then, we show how this error can be used to detect a transition point at which sufficient data has been accumulated to describe a high quality manifold, and computationally lightweight techniques can be used to efficiently map the remaining data in a stream. Our specific contributions are as follows:

1. We formalize a notion of *collective error* in Isomap and describe different strategies to quantify it using *Procrustes analysis*. We perform careful experimental study of the error behavior with respect to the available data using synthetic as well as real-life benchmark data. We identify properties of the error that can be used to detect when manifold becomes stable and robust to incorporating new points.

2. We propose a new efficient algorithm to incorporate streamed data into a stable Isomap manifold. The complexity of the algorithm depends only on the size of the initial stable manifold and is independent of the stream size. Thus, the algorithm is suitable for high-volume and high-throughput stream processing.

The remainder of this chapter is organized as follows. In Section 3.2, we provide preliminary information and high level overview of NLSDR methods. In Section 3, we introduce our approach to quantify error in Isomap, and in Section 4 we report experimental results showing properties of our error measure. In Section 5, we introduce our new efficient algorithm for Out-of-Sample Isomap Extension. We close the chapter with a brief survey of related work in Section 6, and concluding remarks in Section 7.

3.2 Preliminaries

If directly and naively applied on streaming data, NLSDR methods have to recompute the entire manifold each time a new point is extracted from a stream. This quickly becomes computationally prohibitive but guarantees the best possible quality of the learned manifold given the data. To alleviate the computational problem, landmark or general Out-of-Sample Extension methods have been proposed (see Section 3.6). These techniques however either neglect manifold approximation error or remain computationally too expensive for practical applications (for example, their complexity depends directly on the stream size). Here, we propose a different strategy. Initially, we aggregate incoming data and process it using the standard Isomap. At the same time, we trace the quality of the resulting manifold with some computationally acceptable overhead. When we detect that adding new points does not improve the quality of the discovered manifold, i.e. manifold is stable, we drop the standard Isomap and proceed with a faster approximate method that is sufficient to maintain the quality of the manifold. Our approach requires two components that we describe in the following sections: a method to assess accuracy of the learned manifold, and an efficient algorithm to assimilate the remaining points from a stream.

3.3 Proposed Approach to Isomap Error

To quantitatively assess the performance of Isomap in large-scale streaming applications we first need an error metric to determine the accuracy of the learned manifold.

Error in Isomap may arise due to several reasons, including incorrect parameter selection and noisy input data. Parameter selection error may be introduced by a poor choice of neighborhood size, $\mathbf{k}$, or selecting a sub-optimal dimensionality $\mathbf{d}$. Error due to input data can be attributed to noise, missing or limited data entries, or a skewed, rather than uniform, sampling of the underlying manifold. Finally, there could be the intrinsic error of Isomap itself, for instance due to simplifying assumptions about manifold, limited numerical precision, etc.

While error in Isomap has been discussed in prior work (see Section 3.6), most of these studies have focused on one particular aspect of the error. We are interested in understanding the collective error associated with Isomap as a function of the size of the input data. Here we present metrics to measure this error.

3.3.1 Error Definition

We assume that there exists a function $\phi : \mathbb{R}^{\mathbf{d}} \rightarrow \mathbb{R}^{\mathbf{D}}$ that maps each data sample $\mathbf{y}_i \in \mathbb{R}^{\mathbf{d}}$ to $\mathbf{x}_i \in \mathbb{R}^{\mathbf{D}}$. The goal of NLSDR is to learn the inverse mapping, ϕ^{-1}, that can be used to map high-dimensional $\mathbf{x}_i$ to low-dimensional $\mathbf{y}_i$, i.e. $\mathbf{y}_i = \phi^{-1}(\mathbf{x}_i)$.

Let the approximate mapping learned by Isomap be denoted by the inverse function $\hat{\phi}^{-1}$. Let $\mathbf{Y}$ denote the data matrix containing the true low-dimensional mapping for the samples in $\mathbf{X}$, i.e., $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n]^\top$ and $\widehat{\mathbf{Y}}$ denote the matrix with the approximate mapping, i.e. $\widehat{\mathbf{Y}} = [\hat{\phi}^{-1}(\mathbf{x}_1), \hat{\phi}^{-1}(\mathbf{x}_2), \ldots, \hat{\phi}^{-1}(\mathbf{x}_n)]^\top$.

To measure the error between the true representation and the Isomap induced approximate representation, we leverage Procrustes analysis, widely used for shape analysis [17]. The idea behind Procrustes analysis is to align two matrices, $\mathbf{A}$ and $\mathbf{B}$, by finding the optimal translation $\mathbf{t}$, rotation $\mathbf{R}$, and scaling $\mathbf{s}$ that minimizes the Frobenius norm between the two aligned matrices, that is:

$$\mathbf{d}_{Proc}(\mathbf{A}, \mathbf{B}) = \min_{\mathbf{R},\mathbf{t},\mathbf{s}} \|\mathbf{sRB} + \mathbf{t} - \mathbf{A}\|_F. \tag{3.1}$$

The above optimization problem has a closed form solution obtained by performing Singular Value Decomposition of $\mathbf{AB}^T$ [17].

We use the Procrustes analysis to measure the quality of the manifold approximated by Isomap in two ways: 1) A direct method for comparison when low-dimensional ground truth, $\mathbf{Y}$, is available. 2) A reference-sample method that can be used when ground truth is absent. The asymptotic behavior of this method converges to that of the first approach.

3.3.1.1 Direct Procrustes Error

The first approach requires the ground truth reference $\mathbf{Y}$ for the low-dimensional manifold. When both $\mathbf{X}$ and $\mathbf{Y}$ are known, the error ϵ_{Proc} is measured using (3.1), that is:

$$\epsilon_{Proc} = \mathbf{d}_{Proc}(\mathbf{Y}, \widehat{\mathbf{Y}}). \tag{3.2}$$

3.3.1.2 Reference Sample Error

In the absence of low-dimensional ground truth we use a reference-sample method. The reference-sample method works in the following way. Given $\mathbf{X}$, we select $\mathbf{F} \subset \mathbf{X}$, a *reference set*, and two equal sized *sample sets* $\mathbf{R}_1, \mathbf{R}_2 \subset \mathbf{X}$. Next, we create two data sets, $\mathbf{D}_1$ and $\mathbf{D}_2$, such that $\mathbf{D}_i = \mathbf{F} \cup \mathbf{R}_i$ for $i = 1, 2$.

An approximation $\hat{\phi}_i^{-1}$ is learned for each $\mathbf{D}_i$ and error is computed as the Procrustes error for the two learned approximations of the reference set $\mathbf{F}$:

$$\epsilon = \epsilon_{Proc}(\hat{\phi}_1^{-1}(\mathbf{F}), \hat{\phi}_2^{-1}(\mathbf{F})). \tag{3.3}$$

As we increase the size of the sample sets $\mathbf{R}_1$ and $\mathbf{R}_2$ we expect the reference-sample error in (3.3) to behave similar to the Procrustes error computed directly as in (3.2). In fact, we empirically show that both errors have similar asymptotic behavior on several data sets.

3.4 Experimental Results

We present several experiments on a variety of data sets to illustrate the behavior of error using the metrics proposed in Section 3.3. This allows for better understanding of the asymptotic behavior of error in Isomap, required by our

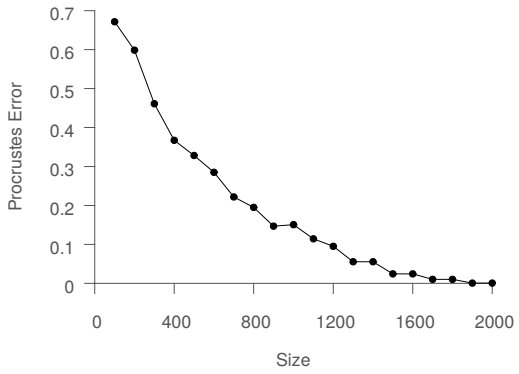


Figure 3.1: Procrustes error decreases asymptotically as **n** increases.

strategy for data stream processing. In particular, our objective is to show that the error converges after a certain number of samples are observed.

3.4.1 Test Data

The *Swiss Roll* data set is typically used for evaluating manifold learning algorithms. *Swiss Roll* data is generated from **2**-dimensional data, embedded into **3** dimensions using a nonlinear function. Specifically, we have $\mathbf{Y} \subset \mathbb{R}^2$ where:

$$\mathbf{Y} = \{(\mathbf{t}, \mathbf{r}) | 1 \leq \mathbf{t} \leq 3, 0 \leq \mathbf{r} \leq 1\}. \tag{3.4}$$

The common approach is to generate the **3-D** *Swiss Roll* from $\mathbf{Y}$ through the use of nonlinear functions of the form $\hat{\mathbf{x}}(\mathbf{t}) = \boldsymbol{\alpha}\mathbf{t} \cdot \cos(\boldsymbol{\beta}\mathbf{t})$, $\hat{\mathbf{y}}(\mathbf{t}) = \boldsymbol{\alpha}\mathbf{t} \cdot \sin(\boldsymbol{\beta}\mathbf{t})$, and $\hat{\mathbf{z}} = \mathbf{r}$. This is problematic due to the fact that $\mathbf{x}(\mathbf{t})$, $\mathbf{y}(\mathbf{t})$ are not isometric, i.e. distances between points in $\mathbf{Y}$ are not preserved along the surface generated in $\mathbb{R}^{\mathbf{3}}$. We propose a new data set to rectify this issue, the *Euler Isometric Swiss Roll*. The idea is to substitute the commonly used $\hat{\mathbf{x}}(\mathbf{t}), \hat{\mathbf{y}}(\mathbf{t})$ with the equations for the *Euler Spiral*:

$$\mathbf{x}(\mathbf{t}) = \int_0^{\mathbf{t}} \sin(\mathbf{s}^2) ds.$$

$$\mathbf{y}(\mathbf{t}) = \int_0^{\mathbf{t}} \cos(\mathbf{s}^2) ds.$$

The Euler spiral has the property that the curvature at any point is propor-

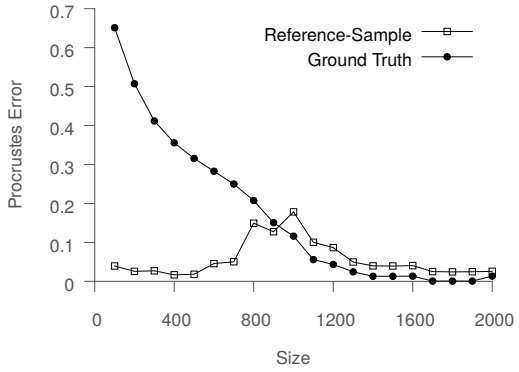


Figure 3.2: The asymptotic behavior error for the Reference Sample method and Procrustes Analysis is similar.

tional to the distance from the origin. This gives constant angular acceleration along the curve thus ensuring that isometry is preserved.

In order to investigate our approach to error analysis in the absence of ground truth, we consider two benchmark data sets: the *MNIST* handwritten digit database and the *Corel Image* data set. The *MNIST* database is composed of **70000** normalized, **28**$\times$**28** grayscale images of handwritten digits **'0'** to **'9'**. Each image is represented by a **784**-dimensional vector resulting from the normalized, grayscale image. Each of the ten digits has roughly **7000** samples. The *Corel Image* data set consists of **68040** photo images from various categories. Each image is represented using **57** features.

First we apply Isomap to samples of data from the *Swiss Roll* dataset. This is to investigate the ability of Isomap to reconstruct the true low-dimensional structure under the conditions of limited data and increasing data. The obtained results are presented in Figure 3.1. Each experiment is performed **10** times and the mean error over all trials is reported.

Figure 3.1 shows that initially for smaller amounts of data the error as formulated in (3.2) is relatively high. As more data samples are used the error first decreases rapidly and then begins to converge at approximately **1500** samples. This suggests that the approximation may be learned within some error tolerance using **1500** points, and that additional data points contribute no significant information. It is at this point that we wish to use more computationally lightweight, approximate methods to process the remaining samples.

Next, we consider the reference-sample method described in Section 3.3. In applying our sampling method to the *Swiss Roll* data we accomplish two goals. Like in the first experiment, we are able to investigate the quality of the Isomap approximation as data availability increases. In addition, because we have ground truth available, we are able to validate our observation that both errors asymptotically converge, in the limit of increasing data.

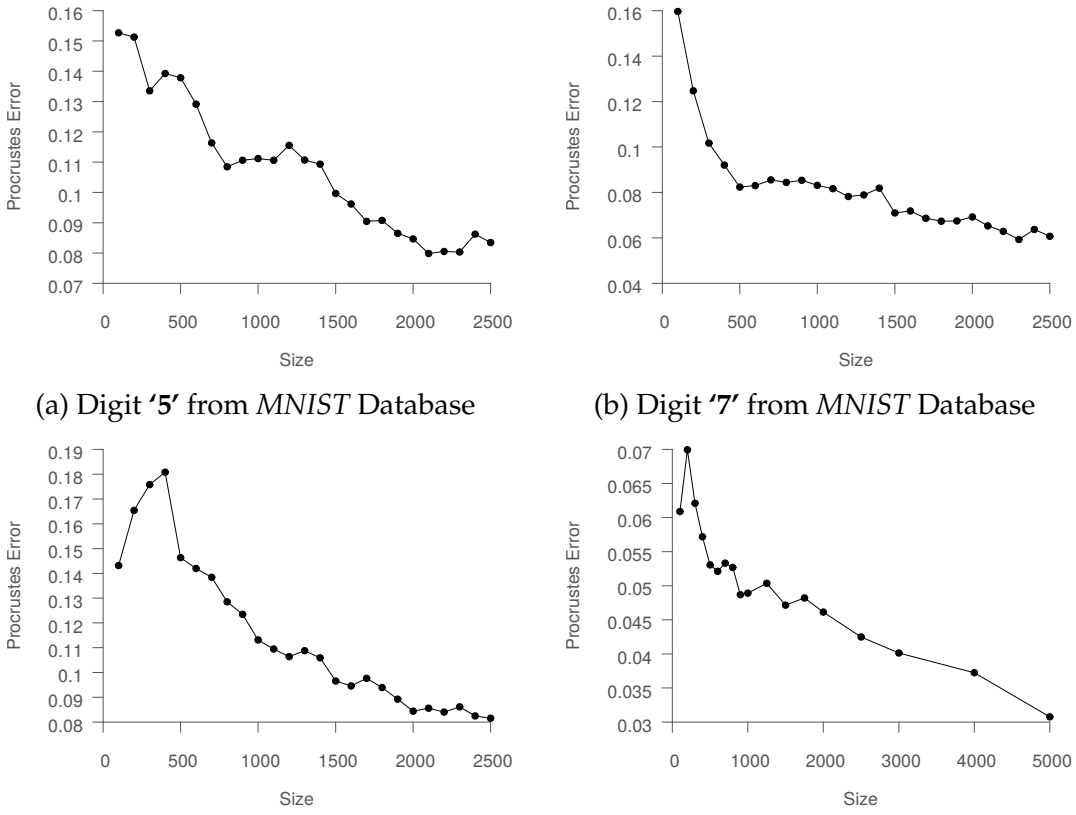
In Figure 3.2, we present results for both approaches to error analysis. In the reference-sample method we take reference set $\mathbf{F}$ with $|\mathbf{F}| = \mathbf{100}$. We start with sample sets $\mathbf{R}_1, \mathbf{R}_2$ with **100** samples and increase their size by **100** samples at each step. The error between $\mathbf{D}_1$ and $\mathbf{D}_2$ is computed as prescribed by (3.3). At the same time, the error for $\mathbf{D}_1$ compared with ground truth coordinates (3.4) is computed and plotted alongside the reference-sample error.

Having gained an understanding of the behavior of errors on synthetic data with availability of ground truth we turn our attention to real-world data. As an example we consider samples for single digits from the *MNIST* database and the *Corel Image* data set. For each data set, we determine its intrinsic dimensionality based on residual variance and then run the reference-sample method to obtain the error plots for increasing sample size, as shown in Figure 3.3.

For each data set, we observe a similar behavior to that seen for both approaches when applied to the *Swiss Roll*. The error decreases rapidly with additional samples. As the size of the sample sets $\mathbf{R}_i$ approaches around **2000-2500** samples, the error begins to stabilize as both approximations become more accurate. The conclusion we draw from this is that for sample of more than **2000** points, a reliable manifold can be learned using both $\mathbf{R}_1$ and $\mathbf{R}_2$, which results in convergence of the error.

3.4.2 Theoretical bounds on the size of $\mathcal{B}$

The threshold for the size for the batch dataset $\boldsymbol{\mathcal{B}}$ i.e. $|\boldsymbol{\mathcal{B}}| = \mathbf{n} > \mathbf{n}_0$ beyond which the Procrustes Error converges (see Figure 3.1) for the synthetically generated *Euler Isometric Swiss Roll* for a single manifold setting is given in the Sec-



(a) Digit **'5'** from *MNIST* Database

(b) Digit **'7'** from *MNIST* Database

(c) Digit **'9'** from *MNIST* Database

(d) *Corel Image* Database

Figure 3.3: Illustrating the stabilization of the learned manifold with increasing data.

tion 3.8. Using the result, we have :

$$\mathbf{n} > \mathbf{n}_0 = (\frac{1}{\widetilde{\boldsymbol{\alpha}}}) \log((\frac{1}{\boldsymbol{\mu}})(\frac{\mathbf{V}}{\widetilde{\mathbf{V}}(\boldsymbol{\delta}/4)}))(\frac{1}{\widetilde{\mathbf{V}}(\boldsymbol{\delta}/2)})$$

To determine the theoretical threshold $\mathbf{n}_0$, we substitute[1] the values of parameters $\widetilde{\boldsymbol{\alpha}}$, the probability of selecting a sample from the fixed distribution $p(\mathbf{y}; \boldsymbol{\theta})$ as $\approx$ **1.0** and $\boldsymbol{\mu}$, the probability associated with the distances ratio bound as $\approx$ **1.0** and substitute parameter $\boldsymbol{\delta}$ associated with the $\boldsymbol{\delta}$-sampling condition as **0.0903**, which is estimated empirically. The value for $\boldsymbol{\eta}_d$, the volume associated

[1] $\widetilde{\boldsymbol{\alpha}}$ was chosen as **1.0** since all points from $\boldsymbol{\mathcal{B}}$ are chosen in the experiment. $\boldsymbol{\mu}$ was chosen as **1.0**, given **0.0** $\leq \boldsymbol{\mu} \leq$ **1.0** and thus any value chosen between **0.0** and **1.0** is reasonable. However we note here that $\boldsymbol{\mu}$ should be ideally chosen closer to **0.0**. Setting $\boldsymbol{\mu} \approx$ **0.0** gives an even higher theoretical threshold on $\mathbf{n}_0$, compared to the result shown in Table 3.1.

with a unit ball in $\mathbb{R}^3$ is given by $\approx \boldsymbol{\eta_d} = \frac{4\pi}{3} = \mathbf{4.1888}$. The value for $(\frac{1}{\widetilde{\mathbf{V}}(\boldsymbol{\delta}/2)})$ is given by $\frac{1}{\eta_d * (\boldsymbol{\delta}/2) * (\boldsymbol{\delta}/2) * (\boldsymbol{\delta}/2)} = \mathbf{2593.8}$. The ratio $(\frac{\mathbf{V}}{\widetilde{\mathbf{V}}(\boldsymbol{\delta}/4)})$ which is the number of balls of radius $(\boldsymbol{\delta}/4)$ needed to cover the volume of manifold $\mathbf{V}$ is estimated empirically as $\approx$ **520**. Thus the value of the theoretically estimated threshold $\mathbf{n}_0$ comes to $\approx (\log(520) * 2593.8) \approx \mathbf{16221}$. The empirical value of threshold $\mathbf{n}_0$ for a single Gaussian patch (see Figure 3.1) is $\approx \frac{2100}{4} = \mathbf{550}$. The theoretically estimated threshold on $\mathbf{n}_0$ is significantly larger than the empirically observed threshold on $\mathbf{n}_0$ in a single manifold setting for the *Euler Isometric Swiss Roll* dataset. The theoretical prediction on $\mathbf{n}_0$ overestimates the empirically observed $\mathbf{n}_0$ for this dataset i.e. we do not require a large $\boldsymbol{\mathcal{B}}$ before the associated Procrustes Error starts to converge.

	Theoretical $\mathbf{n}_0$	Empirical $\mathbf{n}_0$
Swiss Roll	**16221**	**550**

Table 3.1: The theoretically estimated threshold $\mathbf{n}_0$ overestimates the empirically observed threshold $\mathbf{n}_0$ in a single manifold setting for the *Euler Isometric Swiss Roll* dataset.

3.5 Proposed Streaming Isomap Algorithm

To form a stable, well represented manifold, Isomap requires an adequate number of samples to learn the topology of the manifold. More samples can capture the idiosyncrasies of the manifold surface better. However, our experimental results in Section 3.5.3 suggest that there exists a transition point beyond which the quality of the manifold embedding is good. This means that once the point of transition is reached, instead of building the entire manifold in a batch fashion, we can potentially employ a computationally less expensive procedure to map the new samples arriving in the data stream. We demonstrate this result theoretically in the Section 3.8. Based on this result, we propose an Out-of-Sample Extension method called S-Isomap that can adequately predict the embedding of incoming samples in an efficient manner. The algorithm avoids recomputation of the entire geodesic distance matrix or its eigen decomposition, both of which are $\mathcal{O}(\mathbf{n}^3)$. While other Out-of-Sample Extension techniques ex-

ist [6], we demonstrate in Section 3.5.2 that the proposed algorithm is significantly more efficient.

Algorithm 1 STREAMING ISOMAP

Require: $\mathbf{G_b}$, $\mathbf{X}_b$, Y_b, $\mathbf{x}_s$, $\mathbf{k}$
Ensure: $\mathbf{y}_s$
1: $\mathbf{kNN}$, $\mathbf{kDist} \leftarrow \mathrm{KNN}(\mathbf{x}_s, \mathbf{X}_b, \mathbf{k})$
2:
3: **for** $1 \leq \mathbf{i} \leq \mathbf{n}$ **do**
4: $\quad \mathbf{g}_i \leftarrow \min_{1 \leq \mathbf{j} \leq \mathbf{k}} \{\mathbf{kDist}_j + \mathbf{G}_{b_{\mathbf{kNN}_j, i}}\}$
5: **end for**
6:
7: $\mathbf{c} \leftarrow \frac{1}{2}(\bar{\mathbf{g}} \cdot \mathbf{1_n} - \mathbf{g} - \bar{\bar{\mathbf{G}}}_\mathbf{b} \cdot \mathbf{1}_n + \bar{\mathbf{G}}_b)$
8: $\mathbf{p} \leftarrow (\mathbf{Y}_b^\top \mathbf{Y}_b)^{-1} \mathbf{Y}_b^\top \mathbf{c}$
9: $\hat{\mathbf{Y}} \leftarrow [\mathbf{Y}_b; \mathbf{p}]$
10: $\mathbf{y}_s \leftarrow \mathbf{p} - \hat{\mathbf{Y}}$
11: **return** $\mathbf{y}_s$

The algorithm assumes that a transition point has already been identified by monitoring the Isomap error using methods discussed in Section 3.3. Let matrix $\mathbf{X}_b \in \mathbb{R}^{\mathbf{n} \times \mathbf{D}}$ denote the batch of samples encountered in the stream before the point of transition. From our previous experimental results we can assume that $\mathbf{n}$ is relatively small compared to the remaining size of the stream. Let $\mathbf{X}_s \in \mathbb{R}^{\mathbf{m} \times \mathbf{D}}$ represent the remaining part of the input data stream. Furthermore, let $\mathbf{G}_b \in \mathbb{R}^{\mathbf{n} \times \mathbf{n}}$ represent the geodesic distance matrix between the samples in $\mathbf{X}_b$, and let $\mathbf{Y}_b \in \mathbb{R}^{\mathbf{n} \times \mathbf{d}}$ be the matrix containing the corresponding low-dimensional representations of the samples in $\mathbf{X}_b$.

3.5.1 Algorithm

Our proposed method is shown in Algorithm 1. The key assumption is that because the manifold learned from $\mathbf{X}_b$ is stable, we do not have to recompute the entire geodesic distance matrix each time a new point $\mathbf{x}_s$ is added. Instead, it is sufficient that we find the nearest neighbors of $\mathbf{x}_s$ in $\mathbf{X}_b$ and use those to approximate geodesic distance between $\mathbf{x}_s$ and the remaining points in $\mathbf{X}_b$. This step is realized in lines 1-4. Here, $\mathbf{kNN}$ stores indexes of the nearest neigh-

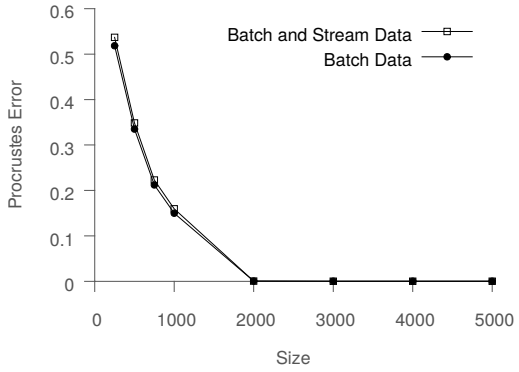


Figure 3.4: The results illustrate that the error due to streaming points is low as well as similar asymptotic behavior.

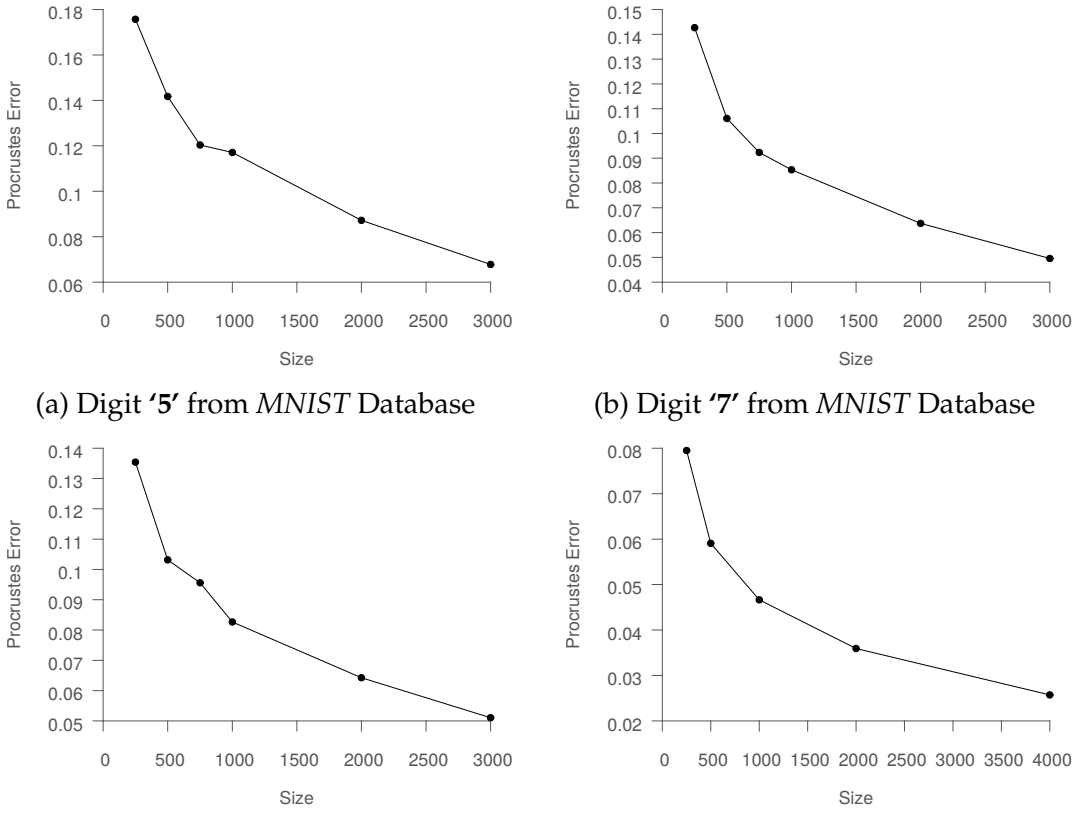
bors of $\mathbf{x}_s$, **kDist** represents the corresponding distances, and $\mathbf{g}_i$ is the approximate geodesic distance between $\mathbf{x}_s$ and the $\mathbf{i}$-th point in $\mathbf{X}_b$. Given the updated geodesic distances we can obtain the low-dimensional coordinates of $\mathbf{x}_s$ using transformations similar to [43]. Specifically, we match the inner product between $\mathbf{y}_s$ and points in $\mathbf{Y}_b$ to the computed geodesics. If we denote the mean of entries of $\mathbf{g}$ by $\bar{\mathbf{g}}$ and the mean of all entries of $\mathbf{G}_b$ by $\bar{\bar{\mathbf{G}}}_\mathbf{b}$, then our desired inner product can be expressed by $\mathbf{c}$ as given in line 6. Here, we use $\mathbf{1}_n$ to represent a vector of $\mathbf{n}$ ones, and $\bar{\mathbf{G}}_b$ to represent the vector of row means of $\mathbf{G}_b$. Then, by solving for $\mathbf{y}_s$, whose inner product with $\mathbf{Y}_b$ is equal $\mathbf{c}$, we obtain the low-dimensional representation of $\mathbf{x}_s$ (lines 7-9).

3.5.2 Analysis

For a single point $\mathbf{x}_s \in \mathbf{X}_s$, the proposed S-Isomap algorithm takes $\mathcal{O}(\mathbf{nD})$ time to compute $\mathbf{k}$NN, $\mathcal{O}(\mathbf{nk})$ to compute $\mathbf{c}$, and $\mathcal{O}(\mathbf{nd}^2)$ for the least-squares estimation in solving for $\mathbf{y}_s$, where $\mathbf{n}$ is the size of the batch $\mathbf{X}_b$. Thus, its runtime complexity is $\mathcal{O}(\mathbf{n}(\mathbf{D} + \mathbf{d}^2 + \mathbf{k}))$ per streaming point. If we consider the cost of learning the initial manifold, the proposed S-Isomap algorithm requires $\mathcal{O}(\mathbf{n}^3 + \mathbf{mn}(\mathbf{D} + \mathbf{d}^2 + \mathbf{k}))$ time to process a stream of size $\mathbf{n} + \mathbf{m}$. This is significant because the cost of mapping a new sample depends only on the batch size $\mathbf{n}$ and parameters $\mathbf{D}, \mathbf{d}$ and $\mathbf{k}$. Since these are very small compared to the entire stream size, the resulting computational savings are significant. Conse-

quently, the algorithm is well suited for high-throughput streams. In contrast, the complexity of the incremental Isomap [43], which is the other method designed for streams, scales quadratically with the size of the stream, and it can be $\mathcal{O}(\mathbf{iD} + \mathbf{i}^2 \log(\mathbf{i}) + \mathbf{i}^2\mathbf{k})$ when inserting $\mathbf{i}$-th sample from the stream if no initial batch is available, or $\mathcal{O}(\mathbf{n}^2 \log(\mathbf{n}) + \mathbf{n}^2\mathbf{k})$ when $\mathbf{X}_b$ is given.

The space complexity of our algorithm is dominated by the terms $\mathcal{O}(\mathbf{n}^2)$ and $\mathcal{O}(\mathbf{nd})$ due to the geodesic distance matrix, $\mathbf{G}_b$, and the low-dimensional representation of the batch samples, $\mathbf{Y}_b$. Thus, the space requirement does not grow with the size of the stream, which makes the algorithm appealing for handling high-volume streams.



(a) Digit **'5'** from *MNIST* Database

(b) Digit **'7'** from *MNIST* Database

(c) Digit **'9'** from *MNIST* Database

(d) *Corel Image* Database

Figure 3.5: The results illustrate that the error due to streaming points is very low, as well as the asymptotic behavior is almost the same.

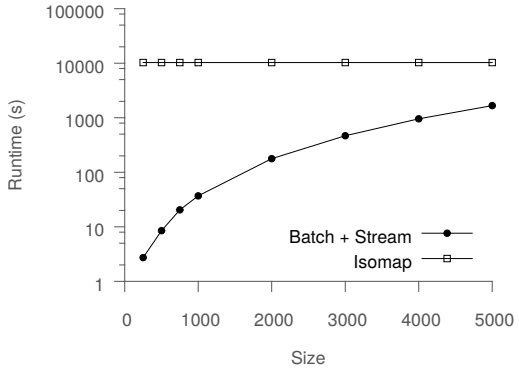


Figure 3.6: Timing results for our method compared to Isomap (horizontal reference line on top) which demonstrate the performance gain achieved.

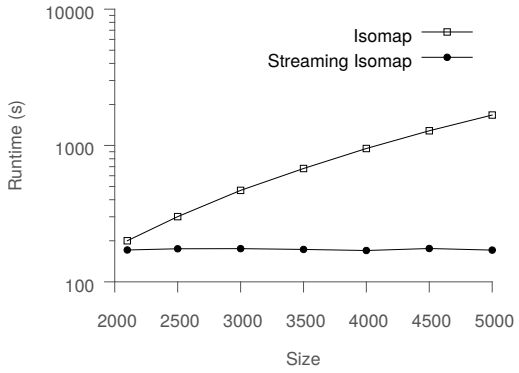


Figure 3.7: Timing results for S-Isomap compared to Isomap for fixed batch size which demonstrate the performance gain achieved.

3.5.3 Experimental Results

To validate the proposed S-Isomap algorithm we perform a set of experiments to asses how accurately and efficiently it maps samples in a stream to a manifold learned from a fixed sized batch.

In the first set of experiments, we measure the error in the mapping obtained using Algorithm 1 and compare it with the error in the mapping from using standard Isomap algorithm. Using a data set $\mathbf{X}$ of size $\mathbf{n} + \mathbf{m}$, we learn a manifold using a batch of size $\mathbf{n}$ and then map the remaining $\mathbf{m}$ samples using S-Isomap. The entire mapping is compared to the ground truth using the Procrustes error. Figure 3.4 shows the results for the *Isometric Swiss Roll* data with $\mathbf{n} + \mathbf{m} = \mathbf{10000}$. Similar results are shown for the other data sets in Figure 3.5.

For the other data sets, we use the mapping obtained by applying Isomap on full data as the ground truth. The results show that while for smaller samples the error is high, the error stabilizes to a low value after the point of transition. The error closely tracks the error for only the batch portion of the data set. This indicates that beyond the point of transition, the mapping obtained using S-Isomap is as accurate as that obtained from the standard Isomap.

To assess the computational efficiency of the S-Isomap algorithm, we show the time taken by S-Isomap (including the time for Isomap on the batch and S-Isomap on remaining stream) in Figure 3.6. The standard Isomap timing result on all **10000** samples is shown as the horizontal baseline for reference. We note that S-Isomap is able to map the samples in the stream much more efficiently than Isomap on the entire data. Overall, Figures 3.4 and 3.6 show that the proposed method is able to map samples in a stream in a highly efficient manner without sacrificing the quality of the manifold.

To understand how S-Isomap would behave in a truly streaming mode we conduct a second timing experiment. Here, we fix the size of the batch to the estimated switching point (**2000** for the *Isometric Swiss Roll* and then progressively add samples to the stream and process it with S-Isomap. We measure the cumulative time taken by S-Isomap to process the remainder of the stream, for different sizes. Figure 3.7 shows the times compared with the runtime for running Isomap on the aggregated batch and stream data. As previously, the cumulative time of running S-Isomap scales linearly with the size of the stream. This further confirms efficiency of the method.

3.6 Related Work

Given the high computational complexity of Isomap, variants of Isomap, such as Landmark Isomap [68] and Out-of-Sample Extension techniques [6], have been proposed as a computationally viable alternative. Both of these methods either use a smaller set of landmark points or approximations to avoid performing the costly eigen decomposition on the $\mathbf{n} \times \mathbf{n}$ geodesic distance matrix, where n is the number of points in the entire data set. However, they still require computing the full geodesic distance matrix which is $\mathcal{O}(\mathbf{rn}^3)$, where $\mathbf{r}$ is the di-

ameter of the embedded **k**NN graph. The Incremental Isomap algorithm [43] avoids both eigen decomposition and a recreation of the geodesic distance matrix. However, it requires updates to the geodesic distance matrix that incurs a significant cost, as discussed in Section 3.5.2. Consequently, the method is unsuitable for the streaming setting.

Errors in Isomap have been discussed in prior work [45], but those studies have been typically in regards to selection of parameters. For example, Samko et al. [63] proposed measuring a simple manifold embedding error for a range of **k** to find the best choice of **k**. Similarly, an approach based on the **k**-edge disjoint minimal spanning tree algorithm has been proposed to construct a neighborhood graph with connectivity guarantees [88]. In the same spirit, several strategies to assess intrinsic manifold dimension **d** are available [45]. However, to the best of our knowledge there is no prior work in defining and understanding the behavior of error that persists even with the selection of optimal parameters. We address this error in taking an abstract view of Isomap, providing a protocol for measuring collective error, and understanding its behavior. In doing so we identify the optimal point where we may switch from exact to lightweight methods.

3.7 Conclusions

The error in Isomap approaches **0** as the density of sampled data tends to infinity. However, in practical settings, we can expect that after a certain level of sampling the error does not change significantly. In other words, the learned manifold becomes stable if the sample size reaches a certain threshold, under the assumption of uniform sampling. In this chapter, we have presented the error metrics that can be used to empirically observe when the manifold becomes stable. In particular, the reference-sample metric is appealing because it can assess the manifold quality even when ground truth data is unavailable.

Equipped with the knowledge of the point of transition, we have presented a streaming algorithm, S-Isomap, that can be used to efficiently map new data samples to the stable manifold, instead of performing costly updates. The fact that the cost of mapping new samples in S-Isomap depends only on the size

of the initial batch used to generate the stable Isomap, and is independent of the size of the stream itself, makes it a very powerful tool to process massive streams of data.

3.8 Theoretical results related to S-Isomap

Proposition 1. *Given an uniformly sampled, unimodal distribution from which the random batch dataset* $\boldsymbol{\mathcal{B}} = \{\mathbf{y}_i \in \mathbb{R}^{\mathbf{D}}\}_{i=1\ldots n}$ *of the S-Isomap algorithm is derived from, there exists a threshold* $\exists \mathbf{n}_0$ *such that when* $\mathbf{n} \geq \mathbf{n}_0$*, the Procrustes Error* $\boldsymbol{\epsilon}_{Proc}\big(\boldsymbol{\tau}_{\boldsymbol{\mathcal{B}}}, \boldsymbol{\tau}_{ISO}\big)$ *between* $\boldsymbol{\tau}_{\mathcal{B}} = \boldsymbol{\phi}^{-1}\big(\boldsymbol{\mathcal{B}}\big)$*, the true underlying representation and* $\boldsymbol{\tau}_{ISO} = \hat{\boldsymbol{\phi}}^{-1}\big(\boldsymbol{\mathcal{B}}\big)$*, the embedding uncovered by Isomap is small (*$\boldsymbol{\epsilon}_{Proc} \approx 0$*) i.e. the batch phase of the S-Isomap algorithm converges.*

Proof. Let us consider the following setting. Low-dimensional ground truth $\mathbf{U}$ originally resides in a convex $\mathbb{R}^{\mathbf{d}}$ Euclidean space. A random subset of samples $\mathbf{X} = \{\mathbf{x}_i\}_{i=1\ldots n}$ where $\mathbf{X} \subseteq \mathbf{U}$ was picked and subsequently mapped via a nonlinear function $\boldsymbol{\phi}$ to $\boldsymbol{\mathcal{B}} \in \mathbb{R}^{\mathbf{D}}$. In this *generative model* perspective, the manifold learning algorithm i.e. Isomap attempts to learn the inverse mapping $\boldsymbol{\phi}^{-1}$, where the associated embedding error is the Procrustes Error $\boldsymbol{\epsilon}_{\text{Proc}}\big(\boldsymbol{\tau}_{\boldsymbol{\mathcal{B}}}, \boldsymbol{\tau}_{\text{ISO}}\big)$.

The proof essentially follows from [7] who showed that in a setting, where given $\boldsymbol{\lambda}_1$, $\boldsymbol{\lambda}_2$, $\boldsymbol{\mu} > 0$ and for appropriately chosen $\boldsymbol{\epsilon} > 0$, as well as a dataset $\mathbf{Y} = \{\mathbf{y}_i\}_{i=1\ldots n}$ sampled from a Poisson distribution with density function $\boldsymbol{\alpha}$ which satisfies the $\boldsymbol{\delta}$-sampling condition i.e.

$$\boldsymbol{\alpha} > \log(\mathbf{V}/(\boldsymbol{\mu}\widetilde{\mathbf{V}}(\boldsymbol{\delta}/4)))/\widetilde{\mathbf{V}}(\boldsymbol{\delta}/2) \tag{3.5}$$

wherein the $\boldsymbol{\epsilon}$-rule is used to construct a graph $\mathbf{G}$ on $\mathbf{Y}$, the following holds with probability at least $(1 - \boldsymbol{\mu})$ for $\forall \mathbf{x}, \mathbf{y} \in \mathbf{Y}$:

$$1 - \boldsymbol{\lambda}_1 \leq \frac{\mathbf{d}_G(\mathbf{x}, \mathbf{y})}{\mathbf{d}_M(\mathbf{x}, \mathbf{y})} \leq 1 + \boldsymbol{\lambda}_2 \tag{3.6}$$

where $\mathbf{V}$ is the volume of the manifold $\boldsymbol{\mathcal{M}}$ and

$$\widetilde{\mathbf{V}}(\mathbf{r}) = \min_{\mathbf{x} \in \boldsymbol{\mathcal{M}}} \text{Vol}(\boldsymbol{\mathcal{B}}_{\mathbf{x}}(\mathbf{r})) = \boldsymbol{\eta}_{\mathbf{d}}\mathbf{r}^{\mathbf{d}} \tag{3.7}$$

is the volume of the smallest metric ball in $\mathcal{M}$ of radius $\mathbf{r}$ and $\delta > 0$ is such that

$$\delta = \lambda_2 \epsilon / 4 \tag{3.8}$$

A similar result can be derived in the scenario where $\mathbf{n}$ points are sampled independently from the fixed probability distribution $p(\mathbf{y}; \theta)$, in which case we have :

$$\mathbf{n}\widetilde{\alpha} = \alpha \tag{3.9}$$

where $\widetilde{\alpha}$ is the probability of selecting a sample from $p(\mathbf{y}; \theta)$.

Using (3.7), (3.8) and (3.9) in (3.5), we have :

$$\begin{aligned} \mathbf{n}\widetilde{\alpha} &> \log(\mathbf{V}/(\mu\widetilde{\mathbf{V}}(\delta/4)))/\widetilde{\mathbf{V}}(\delta/2) \\ &= \left[\log(\mathbf{V}/\mu\eta_d(\lambda_2\epsilon/16)^{\mathbf{d}})\right]/\eta_d(\lambda_2\epsilon/8)^{\mathbf{d}} \end{aligned} \tag{3.10}$$

$$\begin{aligned} \mathbf{n} &> (1/\widetilde{\alpha})\left[\log(\mathbf{V}/\mu\eta_d(\lambda_2\epsilon/16)^{\mathbf{d}})\right]/\eta_d(\lambda_2\epsilon/8)^{\mathbf{d}} \\ &= \mathbf{n}_0 \end{aligned} \tag{3.11}$$

where $\mathbf{n}_0 = (1/\widetilde{\alpha})\left[\log(\mathbf{V}/\mu\eta_d(\lambda_2\epsilon/16)^{\mathbf{d}})\right]/\eta_d(\lambda_2\epsilon/8)^{\mathbf{d}}$, is the condition which ensures that (3.6) is satisfied.

Thus we have derived an adequate threshold for the size of the batch dataset $\mathcal{B}$ which ensures (3.11) is satisfied for the ϵ-rule. We can derive a similar threshold for the $\mathbf{K}$-rule, observing that there is a direct one-to-one mapping between $\mathbf{K}$ and ϵ.

To complete the proof, we observe that (3.6) implies that $\mathbf{d}_{\mathrm{G}}(\mathbf{x}, \mathbf{y})$, the graph based distance between points $\mathbf{x}, \mathbf{y} \in \mathbf{G}$ is a perturbed version of $\mathbf{d}_{\mathrm{M}}(\mathbf{x}, \mathbf{y})$, the true Euclidean distance between points $\mathbf{x}$ and $\mathbf{y}$ in the low-dimensional $\mathbb{R}^{\mathbf{d}}$ space. Let $\widetilde{\mathbf{D}}_{\mathrm{M}}$ and $\widetilde{\mathbf{D}}_{\mathrm{G}}$ represent the squared distance matrix corresponding to $\mathbf{d}_{\mathrm{M}}(\mathbf{x}, \mathbf{y})$ and $\mathbf{d}_{\mathrm{G}}(\mathbf{x}, \mathbf{y})$ respectively. Thus we have $\widetilde{\mathbf{D}}_{\mathrm{G}} = \widetilde{\mathbf{D}}_{\mathrm{M}} + \Delta\widetilde{\mathbf{D}}_{\mathrm{M}}$ where $\Delta\widetilde{\mathbf{D}}_{\mathrm{M}} = \{\widetilde{\mathbf{d}}_{\mathrm{M}}(\mathbf{i}, \mathbf{j})\}_{1 \le i,j \le n}$ and $\widetilde{\mathbf{d}}_{\mathrm{M}}(\mathbf{i}, \mathbf{j})$ are bounded due to (3.6).

[67] demonstrated the robustness of MDS to small perturbations as follows. Let $\mathbf{F}$ represent the zero-diagonal symmetric matrix which perturbs the true squared distance matrix $\mathbf{B}$ to $\mathbf{B} + \Delta\mathbf{B} = \mathbf{B} + \epsilon\mathbf{F}$. Then the Procrustes Error between the embeddings uncovered by MDS for $\mathbf{B}$ and for $\mathbf{B} + \Delta\mathbf{B}$ is given by

$\frac{\epsilon^2}{4}\sum_{j,k}\frac{\mathbf{e}_j^T\mathbf{F}\mathbf{e}_k{}^2}{\lambda_j+\lambda_k}$, which is very small for small entries $\{\mathbf{f}_{i,j}\}_{1\leq i,j\leq n}\in\mathbf{F}$, $\{\mathbf{e}_k(\lambda_k)\}_{k=1\ldots n}$ represent the eigenvectors (eigenvalues) of $\mathbf{B}$ and the double summation is over pairs of $(\mathbf{j},\mathbf{k})=1,2,\ldots(\mathbf{n}-1)$ but excluding those pairs $(\mathbf{j},\mathbf{k})$ wherein both entries of which lie in the range $(\mathbf{K}+1),(\mathbf{K}+2),\ldots(\mathbf{n}-1)$, $\mathbf{K}=\sum_{k=1}^{n}\mathcal{I}(\lambda_k>0)$ and $\mathcal{I}$ is the indicator function. Substituting $\epsilon=1$ and replacing $\mathbf{B}$ with $\widetilde{\mathbf{D}}_{\mathrm{M}}$ and $\Delta\mathbf{B}$ with $\Delta\widetilde{\mathbf{D}}_{\mathrm{M}}$ above, we get our result, since the entries of $\Delta\widetilde{\mathbf{D}}_{\mathrm{M}}$ are very small i.e. $\{0\leq\Delta\widetilde{D}_{\mathrm{M}}(i,j)\leq\lambda^2\}_{1\leq i,j\leq n}$ where $\lambda=\max(\lambda_1,\lambda_2)$ for small λ_1, λ_2, given the condition $\mathbf{n}>\mathbf{n}_0$ is satisfied for (3.6). Thus we have that the embedding uncovered by Isomap for a batch dataset $\mathcal{B}$ where $|\mathcal{B}|=\mathbf{n}>\mathbf{n}_0$ converges asymptotically to their true embedding upto translation, rotation and scaling factors. ■

We additionally note that the asymptotic behavior of error for the Reference-Sample method described in Section 3.3, matching that for the Procrustes Error, for increasing size of the batch dataset is a corollary of Proposition 1 (refer Figure 3.2).

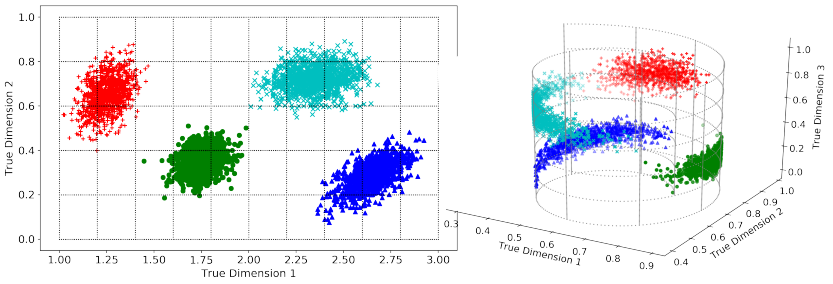
Chapter 4

S-Isomap++

4.1 Introduction

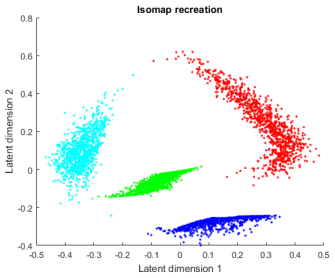
Most existing NLSDR methods have a computational complexity of $\mathcal{O}(\mathbf{n^3})$, $\mathbf{n}$ being the size of the data. The issue is further exacerbated when the data is streaming, where obtaining exact solution at every step of the stream is computationally infeasible. While adapatations of existing NLSDR methods, such as Isomap [74] and LLE [60], have been proposed for handling data streams [40, 43], such methods, which typically rely on incremental updates of the underlying solution, do not scale well to massive streams. In a recent work [65], a two phase strategy has been proposed to adapt Isomap to streaming data. The algorithm, called *S-Isomap*, operates on the core principle that a small batch of data is necessary to *learn* the underlying small-dimensional manifold using an exact and computationally expensive, but data-bounded, learning method. The remainder of the stream may be *mapped* onto the learnt manifold using a relatively inexpensive mapping procedure.

However, the above solution, and other related efforts to adapt NLSDR methods to streaming data [43], rely on the assumption that the data samples lie on a *single* low-dimensional manifold. There have been limited attempts that allow for multiple manifolds [20, 21], however, they assume that the manifolds do not intersect in any ambient space. This is illustrated in Figures 4.1 and 4.3. In Figure 4.1, the synthetic data set in the top panel consists of four "patches" in **2-D**

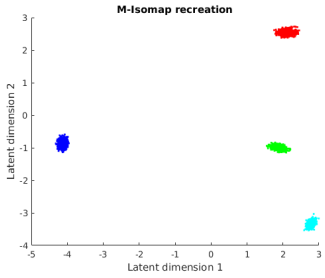


(a) Original Data in **2-D**

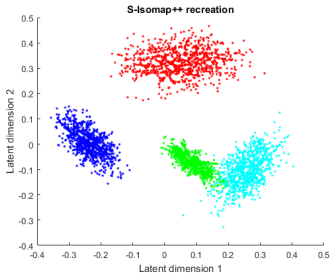
(b) Embedded Data (input) in **3-D**



(c) Isomap Output



(d) M-Isomap Output



(e) Proposed S-Isomap++ Output

Figure 4.1: Multi-manifold *Swiss Roll* data set. The **2-D** samples in (a) are embedded into **3-D** in (b) via the Euler Isometric mapping technique [65]. The reduction to **2-D** is obtained using: (c). Isomap, (d). M-Isomap [20], and (e). the proposed S-Isomap++ algorithm.

space which are embedded onto different regions of a **3-D** *Swiss Roll*. Thus the **3-D** patches data set maybe considered as the high-dimensional data set consisting of samples from multiple manifolds. Direct application of Isomap, which assumes that data comes from a single manifold, results in poor recreation of the ground truth (Figure 4.1c). An existing method, *M-Isomap* [20], that explic-

itly handles multiple manifolds, gives somewhat better results (Figure 4.1d). In Figure 4.2 we demonstrate results of the state-of-the-art t-SNE algorithm on the *Swiss Roll* dataset, given its popularity for the visualization of high-dimensional datasets particularly in scenarios when the underlying manifold dimensionality (**d**) is $\leq$ **3**. The quality of the low-dimensional embedding results uncovered is not good and in fact, we can observe a general *lack of stability* in the results as well as "broken" manifolds. We note here even though t-SNE is not a "streaming" algorithm as such, however our motivation here is to demonstrate its capability in handling multiple manifolds. We show here that t-SNE does not quite capture the low-dimensional structure of the underlying manifold for this dataset. In Figure 4.3, the synthetic data set consists of data from two **2-D** manifolds embedded in a **3-D** space, as an *Isometric Swiss Roll* and a plane, intersecting with each other. In this case, both Isomap and M-Isomap fail (Figure 4.3c), primarily because M-Isomap assumes that the multiple manifolds do not intersect.

The core contribution of this chapter is a streaming NLSDR algorithm, called **S-Isomap++**. The algorithm assumes that the high dimensional input data consists of samples that truly lie on one or more, potentially intersecting, low-dimensional manifolds and are embedded into the high dimensional space via nonlinear transformations. The proposed algorithm extends the widely used Isomap algorithm to handle multiple intersecting manifolds in a streaming setting. Thus, the proposed algorithm operates under one of the least restrictive set of assumptions, explored so far in the context of NLSDR methods (See Figures 4.1e and 4.3d). Moreover, the ability to handle large streams of data makes it highly applicable in a broad variety of domains.

Another contribution of the chapter is a novel *tangent based clustering* strategy to separate samples from the input batch, in the original high-dimensional space, into different clusters. Each cluster is processed independently to obtain the manifold and the corresponding low-dimensional reduction of the corresponding data samples, using Isomap. The reduced data samples are then mapped into a common *ambient* space by exploiting the relationship between the samples across the clusters in the original space. The streaming samples are then mapped, in parallel, on each manifold. An evaluation strategy is employed

to choose the best manifold for each streaming sample.

The rest of the chapter is organized as follows: we provide necessary background about manifold learning in Section 4.2. Related works are discussed in Section 4.3. The proposed algorithm, S-Isomap++, is presented in Section 4.4. Experimental results on synthetic and benchmark datasets, are summarized in Section 4.5.

4.2 Background and Motivation

Our motivation for this work stems from one of the foundation principles of *Manifold Learning*, which assumes that the distribution of the data in the high-dimensional observed space is not uniform and in reality, the data lies near a nonlinear low-dimensional manifold embedded in the high-dimensional space. In many real-world problems such as those resulting from multi-modal or unevenly sampled distributions, the data lies on multiple manifolds of possibly different "dimensionalities" and is typically separated by regions of low density as depicted in Figure 4.4. Thus, to find a representative low-dimensional embedding of the data, one needs to first cluster the data appropriately and subsequently find a low-dimensional representation for the data in each cluster. Even then, manifolds can be very close to each other and can have arbitrary intrinsic dimensions, curvature and sampling which makes it a hard problem to solve.

Typically, nonlinear spectral dimensionality reduction (NLSDR) techniques are used as learning methods for discovering the underlying low-dimensional structure from samples from high-dimensional data. Existing techniques typically exploit either the global (Isomap, Minimum Volume Embedding [83]) or local (LLE, Laplacian Eigenmaps [4]) properties of the manifold to map each high-dimensional point $\mathbf{x}_i \in \mathbb{R}^{\mathbf{D}}$ to its corresponding low-dimensional embedding, $\mathbf{y}_i \in \mathbb{R}^{\mathbf{d}}$. They are used as a generic nonlinear, non-parametric technique to approximate probability distributions in high-dimensional spaces.

The Isomap algorithm, being a global NLSDR technique should ideally provide a more faithful representation and preserve geometry irrespective of scale i.e. map data samples which are close in the manifold to points which are close

in the low-dimensional embedding and similarly for distant samples. However, it struggles when dealing with multi-modal and non-uniform distributions.

4.2.1 Handling Multiple Manifolds

In the ideal scenario, when manifolds are densely sampled and sufficiently separated, existing NLSDR methods can be extended to perform clustering before applying the dimensionality reduction step [56, 20], by choosing an appropriate local neighborhood size so as not to include points from other manifolds and still be able to capture the local geometry of the manifold. However, if the manifolds are close or intersecting (See Figures. 4.3, 4.6), such methods typically fail.

4.3 Related Work

Most existing NLSDR techniques can only deal with a single manifold which leads to them discovering error-prone low dimensional embeddings given inter-manifold distances are usually much larger than the intra-manifold distances.

Wu *et al.* [86] was among the earliest attempts to work with multiple manifolds via NLSDR techniques. Since then, other sophisticated approaches [20, 27, 76, 89, 19] have emerged, apart from techniques in the area of manifold alignment [82, 26] and manifold clustering [69, 20]. Some assumed a supervised setting [27, 76], and learn multiple sub-manifolds corresponding to different given classes in a dataset. The MMDA method [89] is based on Locality Preserving Projections. Similarly, the SMCE algorithm [19] makes assumptions about sparsity and linearity of the embedding.

There have been earlier attempts to cluster sub-manifolds [69, 20], which are primarily based on the idea of forming a graph with edges only between a node and its nearest neighbors. However, these methods cannot deal with intersecting manifolds when it is possible for the local neighborhood of a point to have nearest neighbors from different sub-manifolds. Manifold alignment approaches [82, 26] typically align manifolds using a set of correspondences between data points. Whereas [82] uses Procrustes Analysis, [26] tries to solve a

constrained embedding problem, where the embeddings of the corresponding points from different sets are constrained to be identical.

In a batch setting, the M-Isomap [20] algorithm comes close to our proposed work. The algorithm attempts to work with multiple manifolds embedded in a high-dimensional space. First, it performs clustering to identify the individual sub-manifolds via a nearest neighbor approach and subsequently runs Isomap on each of these sub-manifolds. Finally, it stitches the sub-manifolds together via a set of support points, by finding an optimal transformation between the embeddings uncovered by Multidimensional Scaling(MDS) [41] and Isomap, respectively. However, the nearest neighborhood clustering strategy employed can misrepresent individual sub-manifolds if they are intersecting and/or very close to each other by grouping them together (See Figure 4.3).

4.4 Methodology

There are two key challenges that a streaming manifold learning algorithm has to address: 1) handle streaming data in a scalable manner, and, 2) learn in presence of multiple, possibly intersecting, manifolds.

The proposed S-Isomap++ algorithm follows the two-phase strategy proposed in our earlier work [65], where we first learn exact manifolds from an initial batch, and then employ a computationally inexpensive mapping method to process the remainder of the stream. An error metric is used to decide on when to *switch* from expensive and exact learning to inexpensive and approximate mapping [65]. To address the second challenge, we first cluster the batch data using a *tangent-based manifold clustering* approach and then apply exact Isomap on each cluster. The resulting low-dimensional data for the clusters is then *stitched* together to obtain the data reduced to a low (and closer to true) dimensionality.

The overall S-Isomap++ algorithm is outlined in Algorithm 2. The algorithm takes a batch data set, $\boldsymbol{\mathcal{B}}$ and the streaming data, $\boldsymbol{\mathcal{S}}$ as inputs such that, $\boldsymbol{\mathcal{B}}, \boldsymbol{\mathcal{S}} \in \mathbb{R}^{\mathrm{D}}$. Note that in practical applications, one might not have data split into batch and streaming parts. In that scenario, one may track the quality of the output of the batch phase using suitable error metrics [65], and switch when

a reliable solution for the batch is obtained. For simplicity, we will assume that the optimal batch size has been pre-determined. The processing is split into two phases: a batch learning phase (Lines 1–12) and a streaming phase (Lines 13–20). The batch learning phase consists of three steps:

- Step 1: Cluster samples in $\boldsymbol{\mathcal{B}}$ into $\mathbf{p}$ clusters (Line 1).
- Step 2: Learn $\mathbf{p}$ individual manifolds corresponding to each cluster, and map samples within each cluster to a low-dimensional representation[1] (Lines 6–7).
- Step 3: Map reduced samples from individual manifolds into a global reduced space (Lines 8–12).

In the streaming phase, each sample in the stream set $\boldsymbol{\mathcal{S}}$ is mapped onto each of the $\mathbf{p}$ manifolds by using an inexpensive mapping procedure (Lines 14-17). The nearest manifold is identified by comparing each reduced representation of the sample to the "center" of each manifold (Line 18), and choosing the corresponding reduced representation for the stream sample (Line 19).

The individual components of the proposed S-Isomap++ algorithm are discussed in the subsequent subsections.

4.4.1 Clustering Multiple Intersecting Manifolds

The objective of the first step in Algorithm 2 is to separate the batch samples into clusters, such that each cluster corresponds to one of the multiple manifolds present in the data. Note that, in this chapter, we do not assume that the number of manifolds ($\mathbf{p}$) is specified; it is automatically inferred by the clustering algorithm. In cases of uneven/low density sampling, the clustering strategy discussed might possibly generate many small clusters. In such cases, one can try to merge clusters, based on their affinity/closeness to allow the number of clusters to remain within required limits. Given that the batch samples

[1]The true dimensionality of the manifolds corresponding to the clusters can vary. We assume that the true dimensionality for each cluster has been determined using techniques such as studying the spectral properties of the geodesic distance matrix computed as part of Isomap learning (See Figure 4.5).

Algorithm 2 S-Isomap++

Require: Batch dataset: $\boldsymbol{\mathcal{B}}$, Streaming dataset: $\boldsymbol{\mathcal{S}}$; Parameters: $\boldsymbol{\epsilon}$, $\mathbf{k}$, $\mathbf{l}$, $\boldsymbol{\lambda}$
Ensure: $\boldsymbol{\mathcal{Y}}_{\mathcal{S}}$: low-dimensional representation for $\boldsymbol{\mathcal{S}}$

$\boldsymbol{\mathcal{C}}_{i=1,2\ldots\mathbf{p}} \leftarrow$ FIND_CLUSTERS($\boldsymbol{\mathcal{B}}$, $\boldsymbol{\epsilon}$)
$\boldsymbol{\xi}_s \leftarrow \emptyset$

for $1 \leq \mathbf{i} \leq \mathbf{p}$ **do**
    $\boldsymbol{\mathcal{LDE}}_i \leftarrow$ ISOMAP($\boldsymbol{\mathcal{C}}_i$)
end for

$\boldsymbol{\xi}_s \leftarrow \bigcup_{i=1}^{p} \bigcup_{j=i+1}^{p} \mathrm{NN}(\boldsymbol{\mathcal{C}}_i, \boldsymbol{\mathcal{C}}_j, \mathbf{k}) \cup \mathrm{FN}(\boldsymbol{\mathcal{C}}_i, \boldsymbol{\mathcal{C}}_j, \mathbf{l})$

$\boldsymbol{\mathcal{GE}}_s \leftarrow \mathrm{MDS}(\boldsymbol{\xi}_s)$

for $1 \leq \mathbf{j} \leq \mathbf{p}$ **do**
    $\boldsymbol{\mathcal{I}} \leftarrow \boldsymbol{\xi}_s \cap \boldsymbol{\mathcal{C}}_j$
    $\boldsymbol{\mathcal{A}} \leftarrow \begin{bmatrix} \boldsymbol{\mathcal{LDE}}_j^{\mathcal{I}} \\ \mathbf{e}^T \end{bmatrix}$
    $\boldsymbol{\mathcal{R}}_i, \mathbf{t}_i \leftarrow \boldsymbol{\mathcal{GE}}_{\mathcal{I},s} \times \boldsymbol{\mathcal{A}}^T \left(\boldsymbol{\mathcal{A}}\boldsymbol{\mathcal{A}}^T + \boldsymbol{\lambda}\mathbf{I}\right)^{-1}$
end for

for $\mathbf{s} \in \boldsymbol{\mathcal{S}}$ **do**
    for $1 \leq \mathbf{i} \leq \mathbf{p}$ **do**
        $\mathbf{y}_s^i \leftarrow$ S-ISOMAP($\mathbf{s}, \boldsymbol{\mathcal{C}}_i$)
        $\boldsymbol{\mathcal{GE}}_s^i \leftarrow \boldsymbol{\mathcal{R}}_i \mathbf{y}_s^i + \mathbf{t}_i$
    end for

    $\mathbf{m} \leftarrow \mathrm{argmin}_i \left| \mathbf{y}_s^i - \mu(\boldsymbol{\mathcal{C}}_i, \boldsymbol{\mathcal{R}}_i, \mathbf{t}_i) \right|$
    $\boldsymbol{\mathcal{Y}}_{\mathcal{S}} \leftarrow \boldsymbol{\mathcal{Y}}_{\mathcal{S}} \cup \mathbf{y}_s^m$
end for

return $\boldsymbol{\mathcal{Y}}_{\mathcal{S}}$

lie on low-dimensional and potentially intersecting manifolds, it is evident that the standard clustering methods, such as K-Means [35], that operate on the observed data in $\mathbb{R}^{\mathbf{D}}$, will fail in correctly identifying the clusters.

To handle this challenge, we propose a novel clustering algorithm that is based on the notion of smoothness of manifold surfaces. Consider a single batch data sample, $\mathbf{x}_i \in \mathbb{R}^{\mathbf{D}}$. Let $\boldsymbol{\mathcal{N}}(\mathbf{x}_i)$ be the set of $\mathbf{k}$ nearest neighbor samples of $\mathbf{x}_i$

Algorithm 3 Tangent Manifold Clustering

function FIND_CLUSTERS($\boldsymbol{\mathcal{B}}$, $\boldsymbol{\epsilon}$)
  $\boldsymbol{\mathcal{S}}_{i=1,2...n} \leftarrow$ MSVD($\boldsymbol{\mathcal{B}}$)
  $\boldsymbol{\rho} \leftarrow \mathbf{0}_{n\times 1}$
  $\mathbf{idx} \leftarrow \mathbf{1}$
  while $\boldsymbol{\rho}_{i=1,2...n} \neq \mathbf{0}$ **do**
    $\boldsymbol{\mathcal{C}}_{idx}$, $\boldsymbol{\rho} \leftarrow$ CLUSTER($\boldsymbol{\mathcal{B}}, \boldsymbol{\mathcal{S}}, \boldsymbol{\rho}, \mathbf{idx}, \boldsymbol{\epsilon}$)
    $\mathbf{idx} \leftarrow \mathbf{idx} + \mathbf{1}$
  end while
  return $\boldsymbol{\mathcal{C}}_{i=1,2...p}$
end function

in the batch $\boldsymbol{\mathcal{B}}$. Let $\boldsymbol{\mathcal{T}}_i$ denote a $\mathbf{d}'$ dimensional *tangent plane* represented using $\mathbf{d}'$ basis vectors, $\mathbf{t}_{i1}, \mathbf{t}_{i2}, \ldots, \mathbf{t}_{id'}$, i.e., $\boldsymbol{\mathcal{T}}_i = span(\mathbf{t}_{i1}, \mathbf{t}_{i2}, \ldots, \mathbf{t}_{id'})$. Here, $\mathbf{d}'$ denotes the intrinsic dimensionality of the tangent plane. We assume that each $\mathbf{x}_i$ belongs to a single manifold $\boldsymbol{\mathcal{M}}_j, \exists \mathbf{j} \in \{1, 2 \ldots \mathbf{p}\}$.

The proposed clustering algorithm (Algorithm 3) is based on the following intuition: For a given sample, $\mathbf{x}_i$, and its neighbor $\mathbf{x}_j \in \boldsymbol{\mathcal{N}}(\mathbf{x}_i)$:

$$\text{If } \boldsymbol{\mathcal{M}}_i = \boldsymbol{\mathcal{M}}_j \Rightarrow \phi(\boldsymbol{\mathcal{T}}_i, \boldsymbol{\mathcal{T}}_j) \geq \boldsymbol{\epsilon} \tag{4.1}$$

$\phi(\boldsymbol{\mathcal{T}}_i, \boldsymbol{\mathcal{T}}_j) = \cos(\boldsymbol{\theta})$, where $\boldsymbol{\theta}$ is the angle between the two tangent planes[2], $\boldsymbol{\mathcal{T}}_i$ and $\boldsymbol{\mathcal{T}}_j$. Similarly,

$$\text{If } \boldsymbol{\mathcal{M}}_i \neq \boldsymbol{\mathcal{M}}_j \Rightarrow \phi(\boldsymbol{\mathcal{T}}_i, \boldsymbol{\mathcal{T}}_j) < \boldsymbol{\epsilon} \tag{4.2}$$

In other words, within a tight neighborhood, a given data sample and its neighbors are expected to lie on tangent planes that are approximately similar in orientation, and, thus, the cosine of the angle between the two planes will be closer to $\mathbf{1}$ ($\cos(\boldsymbol{\theta}) \approx 1$). However, if a sample's neighborhood contains samples that lie on other intersecting manifolds, their tangent planes should be significantly different, and $\cos(\boldsymbol{\theta}) \ll \mathbf{1}$.

[2] $\boldsymbol{\mathcal{T}}_i$ and $\boldsymbol{\mathcal{T}}_j$ are the tangent planes for the samples $\mathbf{x}_i$ and $\mathbf{x}_j$.

Algorithm 4 Incremental Partitioning Strategy

function CLUSTER($\boldsymbol{\mathcal{B}}, \boldsymbol{\mathcal{S}}, \boldsymbol{\rho}, \mathbf{m}, \epsilon$)
  $\boldsymbol{\mathcal{C}}_m \leftarrow \emptyset, \boldsymbol{\mathcal{C}}_o \leftarrow \emptyset$
  $\boldsymbol{\mathcal{I}} \leftarrow \{\mathbf{i} | \boldsymbol{\rho}_i = \mathbf{0}\}, \mathbf{k} \sim$ RANDOM($\boldsymbol{\mathcal{I}}$)

  $\boldsymbol{\mathcal{C}}_m \leftarrow \boldsymbol{\mathcal{C}}_m \cup \boldsymbol{\mathcal{B}}_k$
  $\boldsymbol{\mathcal{C}}_o \leftarrow \boldsymbol{\mathcal{C}}_o \cup \boldsymbol{\mathcal{B}}_k, \boldsymbol{\rho}_k \leftarrow \mathbf{m}$
  $\boldsymbol{\sigma}_n \leftarrow \mathbf{1}$, **mode** = 'L1'

  while $\boldsymbol{\sigma}_n > \mathbf{0}$ **do**
    $\boldsymbol{\sigma}_n \leftarrow \mathbf{0}, \boldsymbol{\mathcal{C}}_n \leftarrow \emptyset$

    for $\forall \mathbf{i} \in \boldsymbol{\mathcal{C}}_o$ **do**
      $\boldsymbol{\mathcal{I}}_{knn} \leftarrow$ KNN($\boldsymbol{\mathcal{B}}$, **i**)
      for $\forall \mathbf{j} \in \boldsymbol{\mathcal{I}}_{knn}$ **do**
        if $\mathbf{labels}_j = \mathbf{0}$ **then**
          $\mathbf{sim}_{i,j} \leftarrow$ SIM($\boldsymbol{\mathcal{S}}_i, \boldsymbol{\mathcal{S}}_j$, **mode**)
          if $\mathbf{sim}_{i,j} \geq \epsilon$ **then**
            $\boldsymbol{\mathcal{C}}_n \leftarrow \boldsymbol{\mathcal{C}}_n \cup \boldsymbol{\mathcal{B}}_j$
            $\mathbf{labels}_j \leftarrow \mathbf{m}$
            $\boldsymbol{\sigma}_n \leftarrow \boldsymbol{\sigma}_n + \mathbf{1}$
          end if
        end if

      end for
    end for

    $\boldsymbol{\mathcal{C}}_m \leftarrow \boldsymbol{\mathcal{C}}_m \cup \boldsymbol{\mathcal{C}}_n, \boldsymbol{\mathcal{C}}_o \leftarrow \boldsymbol{\mathcal{C}}_n$
  end while

  return $\boldsymbol{\mathcal{C}}_m, \boldsymbol{\rho}$
end function

4.4.1.1 Learning a Tangent Plane for a Given Sample

We use *Multiscale Singular Value Decomposition* (or MSVD [48]) on the local neighborhood of $\mathbf{x}_i$, to determine basis vectors, $\mathbf{t}_{i1}, \mathbf{t}_{i2}, \ldots, \mathbf{t}_{id'}$, which define the tangent plane, $\mathcal{T}_i$.

Use of SVD allows us to follow the intuitions expressed in (4.1) and (4.2), since it explores directions in which the spread of points is maximal. In the

presence of multiple intersecting manifolds, these directions get mangled up, whereas non-intersecting regions have better agreement with regards to principal directions.

MSVD allows us to deal with the problem of estimating the intrinsic dimension of noisy, high-dimensional point clouds. For the linear case, SVD analysis can estimate the intrinsic dimensionality of $\mathcal{M}$ correctly, with high probability. However, when $\mathcal{M}$ is a nonlinear manifold, curvature forces the dimensionality of the best-approximating hyperplane to be much higher, which hinders attempts to uncover the true intrinsic dimensionality of $\mathcal{M}$. Little *et al.* [48] show that this is due to performing PCA globally rather than locally.

MSVD estimates the intrinsic dimensionality of $\mathcal{M}$ by computing the singular values, $\sigma_i^{z,r}$ for all $\forall \mathbf{z} \in \mathcal{M}$ at different scales $\mathbf{r} > 0$ and $\mathbf{i} \in \{1, 2, \ldots \mathbf{D}\}$. Small values of $\mathbf{r}$ lead to not enough samples in $\mathcal{B}(\mathbf{z}, \mathbf{r})$, while large values of $\mathbf{r}$ lead to curvature making the SVD computation over estimate the intrinsic dimensionality. At the right scale (value of $\mathbf{r}$), the true $\sigma_i^{z,r}$'s separate from the noise $\sigma_i^{z,r}$'s due to their different rates of growth and the true dimensionality of $\mathcal{M}$ is revealed. Figure 4.5 demonstrates how $\sigma_i^{\mathbf{z},\mathbf{r}}$ behave over different scales when MSVD is done a noisy $\mathcal{R}^5$ sphere embedded in $\mathbb{R}^{100}$ ambient space. Notice how the noise dimensions decay out, leaving only the primary components at the appropriate scale.

4.4.1.2 Computing Angle Between Two Tangent Planes

We explore several strategies of computing the similarity between a pair of tangent planes, $\mathcal{T}_i$ and $\mathcal{T}_j$. As mentioned earlier, this is equivalent to computing the cosine of the angle between the two planes. We consider one approach, as proposed by Gunawan *et al.* [24]. Let $\mathcal{T}_i$ and $\mathcal{T}_j$ be orthonormal subspaces[3]. If $\boldsymbol{\theta}$ is the angle between $\mathcal{T}_i$ and $\mathcal{T}_j$, then:

$$\phi(\mathcal{T}_i, \mathcal{T}_j) = \cos(\boldsymbol{\theta}) = \sqrt{det(\mathcal{N}\mathcal{N}^\top)} \tag{4.3}$$

[3]One can use QR factorization to orthonormalize any subspace, which is not already orthonormal.

where $\mathcal{N}$ is a matrix, such that $\mathcal{N}[\mathbf{u}][\mathbf{v}] = \langle \mathbf{t}_{iu}, \mathbf{t}_{jv} \rangle$, where $\mathbf{t}_{iu}$ is the $\mathbf{u}^{th}$ basis vector for $\mathcal{T}_i$ and $\mathbf{t}_{jv}$ is the $\mathbf{v}^{th}$ basis vector for $\mathcal{T}_j$.

Additionally, when the dimensionality of $\mathcal{T}_i$ and $\mathcal{T}_j$ is same, the expression simplifies to:

$$\phi(\mathcal{T}_i, \mathcal{T}_j) = \cos(\theta) = |det(\mathcal{N})| \tag{4.4}$$

Alternately, one can use the following procedure. Without loss of generality, let us assume that $\mathbf{t}_{i1}, \mathbf{t}_{i2}, \ldots, \mathbf{t}_{ik}$ are the singular vectors for the plane $\mathcal{T}_i$ corresponding to the top $\mathbf{k}$ singular values. Similarly, let $\mathbf{t}_{j1}, \mathbf{t}_{j2}, \ldots, \mathbf{t}_{jk}$ be the top-$\mathbf{k}$ singular vectors for the plane $\mathcal{T}_j$. Then we can compute $\phi(\mathcal{T}_i, \mathcal{T}_j)$ as:

$$\phi(\mathcal{T}_i, \mathcal{T}_j) = \frac{1}{\mathbf{k}} \sum_{l=1}^{k} |\mathbf{t}_{il}^{\top} \mathbf{t}_{jl}| \tag{4.5}$$

We refer to the above as the $L1$ metric. In the same way, one can define the $L2$ metric as:

$$\phi(\mathcal{T}_i, \mathcal{T}_j) = \sqrt{\frac{1}{\mathbf{k}} \sum_{l=1}^{k} (\mathbf{t}_{il}^{\top} \mathbf{t}_{jl})^2} \tag{4.6}$$

4.4.1.3 Tangent Manifold Clustering Algorithm

The proposed tangent manifold clustering strategy is outlined in Algorithm 3. Algorithm 4 is the support method to the above. The inputs to the Algorithm 3 are the batch dataset $\mathcal{B}$ and a threshold value ϵ.

Algorithm 3 initially calls MSVD($\cdot$) (See Section 4.4.1.1) on the input batch set, $\mathcal{B}$, to decide on an appropriate scale $\mathbf{r}$ to use and subsequently to extract the top-$\mathbf{k}$ singular vectors $\mathcal{S}_{i=1,2\ldots n}$ for all $\mathbf{x}_i \in \mathcal{B}$, at the scale $\mathbf{r}$. Initially all points are unlabeled i.e. ρ is all zeros initially. Algorithm 3 calls CLUSTER($\cdot$) repeatedly till all $\mathbf{x}_i \in \mathcal{B}$ have labels assigned to them, which represents the different clusters, $\mathcal{C}_i$ for $\mathbf{i} = 1, 2 \ldots \mathbf{q}$ where $\bigcup_{i=1}^{q} \mathcal{C}_i = \mathcal{B}$.

Algorithm 4, which contains the function CLUSTER($\cdot$)[4], works as follows: it picks a currently unlabeled $\mathbf{x}_i$ at random, and assigns it to a new cluster $\mathcal{C}_m$. Subsequently, it looks at the unassigned nearest neighbors of $\mathbf{x}_i$ i.e. $\mathbf{x}_j \in \mathcal{N}(\mathbf{x}_i)$

[4]We use 'L1' as the mode by default (Line 6) since it provides the best accuracy.

and checks to see how close their tangent planes are. If they are similar enough i.e. the similarity score $\phi(\mathcal{T}_i, \mathcal{T}_j) \geq \epsilon$, then the unassigned nearest neighbor is assigned to $\mathcal{C}_m$. The algorithm proceeds similarly in a breadth-first manner till no new points remain to be tested.

It internally calls Algorithm SIM$(\cdot)$ to measure similarity, using one of the three strategies, discussed in Section 4.4.1.2 (See (4.4), (4.5), and (4.6)).

4.4.2 Processing multiple manifolds

The S-Isomap++ algorithm independently learns the manifolds for each cluster (Lines 3–5). However, since these manifolds are not necessarily aligned with respect to each other, an additional step is needed to represent the reduced samples from each cluster into a common space. We refer to this process as *stitching*, and is essential to recreate the final reduced representation. This step, similar to the approach in M-Isomap, maintains the information of the global location of different manifolds using a set of support points which form the skeleton on which it can later places the different manifolds. This support set is formed using the $\mathbf{k}$ nearest neighbor pairs as well as the $\mathbf{l}$ farthest neighbor pairs between every pair of manifolds present i.e. $\forall\{\mathcal{C}_i, \mathcal{C}_j\}_{j \neq i}$, let $\mathcal{X}_{i,j} \in \mathbb{R}^{|\mathcal{C}_i| \times |\mathcal{C}_j|}$ denote the $\mathbb{R}^{\mathbf{D}}$ Euclidean distance matrix between all points in clusters $\mathcal{C}_i$ and $\mathcal{C}_j$, then support set $\boldsymbol{\xi}_s$ contains the co-ordinates (index sets $\mathcal{I}_i$ and $\mathcal{I}_j$ from $\mathcal{C}_i$ and $\mathcal{C}_j$ respectively) of both the smallest $\mathbf{k}$ values as well as the largest $\mathbf{l}$ values in $\mathcal{X}_{i,j}$. The former are calculated by method NN$(\cdot)$ and the latter FN$(\cdot)$ (Line 6). Subsequently, a global reduced space embedding $\mathcal{GE}_s$ for this support set is calculated using MDS (Line 7). After this, for each manifold $\mathcal{M}_j, \exists \mathbf{j} \in \{1, 2 \ldots \mathbf{p}\}$, a least-squares problem is solved to generate the transformation components $\mathcal{R}_i, \mathbf{t}_i$ which can project reduced samples from each cluster into the global space (Lines 8–12).

4.4.3 Mapping Streaming Samples

In the streaming part, each sample in the stream set $\mathcal{S}$ is mapped onto each of the $\mathbf{p}$ manifolds in parallel, using the inexpensive S-ISOMAP$(\cdot)$ algorithm proposed in our earlier work [65] (Line 15) and subsequently mapped to the global

space using $\{\boldsymbol{\mathcal{R}}_i, \mathbf{t}_i\}\ \exists \mathbf{i} \in \{1, 2 \ldots \mathbf{p}\}$ (Line 16). The nearest manifold is identified by comparing each reduced representation of the sample to the mean $\mu(\cdot)$ of each manifold (Line 18), and choosing the corresponding reduced representation for the stream sample (Line 19).

4.5 Results and Analysis

4.5.1 Experimental Setup

We present several experiments here on a variety of data sets to illustrate the behavior of different approaches proposed in the Section 4.4.

We use four different datasets in our experiments. Given *Swiss Roll* datasets are typically used for evaluating manifold learning algorithms, we use the *Euler Isometric Swiss Roll* dataset, proposed by Schoeneman *et al.* [65], wherein a $\mathbb{R}^{\mathbf{2}}$ data set having $\mathbf{n} = \mathbf{3000}$ points, chosen at random, are embedded into $\mathbb{R}^{\mathbf{3}}$ using a nonlinear function $\psi(\cdot)$. We use this in conjunction with a $\mathbb{R}^{\mathbf{3}}$-dimensional hyperplane passing through it as shown in Figure. 4.3b having $\mathbf{n} = \mathbf{1500}$ points, chosen at random. We know the ground truth for both parts (See Figure. 4.3a). We use this to evaluate the S-Isomap++ algorithm as shown in Figure. 4.3. We also use an extension of this, wherein two $\mathbb{R}^3$-dimensional hyperplanes pass through the *Isometric Swiss Roll*, wherein the points are chosen in random and each hyperplane has $\mathbf{n} = \mathbf{3000}$ points, as shown in Figure. 4.6.

Apart from this, we use different artificial datasets consisting of intersecting manifolds i.e. two intersecting $\mathbb{R}^{\mathbf{3}}$-dimensional unit hyperspheres, having $\mathbf{n} = \mathbf{1000}$ points each and a $\mathbb{R}^{\mathbf{3}}$-dimensional plane intersecting a $\mathbb{R}^{\mathbf{3}}$-dimensional hypersphere, again having $\mathbf{n} = \mathbf{1000}$ points each, as shown in Figure. 4.7. We use these datasets to test our tangent manifold approach more rigorously. We also use patches on the *Euler Isometric Swiss Roll* dataset (Figure. 4.1) which are Gaussian in nature, to study the effect of the different parameters, apart from evaluating our algorithm, as well as the *MNIST* digits dataset.

Our evaluation metrics for the experiments primarily focus on 1) ability on our tangent manifold clustering strategy to be able to cluster points from multiple intersecting/non-intersecting manifolds correctly, 2) test the quality of the

embedding uncovered by our algorithm, for the streaming dataset $\boldsymbol{\mathcal{S}}$, with regards to agreeability with ground truth via an appropriate distance metric, as well as, tightness of clustering and last but not the least, 3) scalability of our algorithm over different sizes of both batch and streaming datasets $\boldsymbol{\mathcal{B}}$ and $\boldsymbol{\mathcal{S}}$ respectively.

4.5.2 Results on Artificial Datasets

4.5.2.1 Gaussian patches on *Isometric Swiss Roll*

Figures. 4.1c, 4.1d, 4.1e demonstrate the results with this dataset for Isomap, M-Isomap and our approach respectively. Both the M-Isomap and S-Isomap++ algorithms can deal with individual manifolds better than Isomap, which severely deforms the individual clusters. It should also be noted that whereas both the M-Isomap and S-Isomap++ algorithms required small values of $\mathbf{k}$ i.e. $\mathbf{k} = \mathbf{8}$ to operate, Isomap needed values of $\mathbf{k} \geq \mathbf{500}$ to even work. As a consequence, idiosyncrasies i.e. short-circuiting become a factor to distort the uncovered embedding. M-Isomap has scaling issues and can only seem to attempt to position the individual manifolds in the global ambient space correctly, without being able to recreate the spread, which defined the individual manifolds. We think that M-Isomap internally normalizes individual manifolds which results in this behavior. Our approach, S-Isomap++ is the most robust in its recreation of the ground truth.

4.5.2.2 Intersecting *Swiss Roll* with $\mathbb{R}^3$-dimensional plane

Figure. 4.3 demonstrates our experiments with this dataset. We evaluate different algorithms to see how well they recreate the ground truth (Figure. 4.3a). Both Isomap and M-Isomap produce the same output, given M-Isomap employs a nearest-neighbor based clustering strategy to disambiguate between manifolds, and hence is unable to handle intersecting manifolds, which results in highly distorted recreations of the ground truth. As before, S-Isomap++ produces the most robust recreation of the ground truth. Figure 4.6, demonstrates how well S-Isomap++ recreates the original manifolds, in case the batch $\boldsymbol{\mathcal{B}}$ is

clustered correctly. M-Isomap/Isomap are unable to recreate the ground truth and severely contort the ground truth.

4.5.2.3 Tangent Manifold Clustering

Here we present clustering results for intersecting manifolds. (See Figures. 4.3, 4.7 for the different datasets). Table 4.1 below demonstrates accuracy values[5] with which the **L-1**, **L-2** metric schemes proposed in this work, along with the technique proposed by Gunawan *et al.* [24] clustered the different intersecting manifolds. The **L-2** metric performed much better than Gunawan's approach, however the **L-1** metric performed the best. The accuracy values are also indicative of the level of difficulty associated with clustering the different scenarios correctly.

Method	*L-1*	*L-2*	*Gunawan*
Sphere-Sphere	**0.825**	0.619	0.5
Sphere-Plane	**0.759**	0.602	0.5
Swiss Roll-Plane	**0.838**	0.621	0.5

Table 4.1: Accuracy scores for the different tangent manifold clustering approaches.

4.5.2.4 Effect of different parameters

Here we present results of the effect of changing the different parameters of the S-Isomap++ algorithm, *while keeping all other parameters fixed*. Figures 4.8, 4.9, 4.10, demonstrates the effect of parameter $\boldsymbol{\lambda}$, $\mathbf{k}$ and $\mathbf{l}$ on the embeddings uncovered by the S-Isomap++ algorithm. Larger values of $\mathbf{k}$ seems to make the manifolds more uniform or rounded. Larger values of parameter $\mathbf{l}$ seem to stretch the manifolds. Parameter $\boldsymbol{\lambda}$ seem to separate the manifolds apart when it has larger values. This is really interesting since it means we can use it to visualize manifolds better on account of separability.

[5]Gunawan's approach was unable to distinguish between the intersecting manifolds scenarios and always clustered them as one and hence its accuracy was **0.5** in all cases.

Figure 4.11 demonstrates the scalability of our algorithm with regards to streaming data $\mathcal{S}$. Batch $\mathcal{B}$ having size $\mathbf{n} = \mathbf{2000}$ was used for this experiment. The timing results are in log scale and clearly demonstrate the efficiency gained. M-Isomap has the same result as Isomap since it cannot distinguish between intersecting manifolds and treats them as one. While the run-time for Isomap/M-Isomap increases rapidly with increasing stream size, the run time for S-Isomap++ does not grow much at all, making it highly conducive to large stream processing.

4.5.3 Results on *MNIST* Dataset

Table 4.2 below shows results for different digits of the *MNIST* dataset. Using a batch dataset $\mathcal{B}$ of size $\mathbf{n} = \mathbf{2000}$, a streaming dataset $\mathcal{S}$ of size $\mathbf{m} = \mathbf{4000}$ was recreated in **3-D** by the S-Isomap++ algorithm, for each of the digits. Subsequently the **3-D** recreation was compared to the **3-D** ground truth obtained by running Isomap on all digits, using the Procrustes Error metric to measure the quality of the recreation.

The Procrustes Error metric determines an optimal alignment between two matrices $\mathcal{X}$ and $\mathcal{Y}$ and returns a goodness-of-fit criterion, based on sum of squared errors. As the results below demonstrate, the recreation error is pretty low, even after embedding in the common global space. This shows the efficacy of the S-Isomap++ algorithm.

digit '0'	**0.0296**	*digit '3'*	**0.0364**	*digit '6'*	**0.0476**
digit '1'	**0.0806**	*digit '4'*	**0.0586**	*digit '8'*	**0.0712**
digit '2'	**0.0499**	*digit '5'*	**0.0449**	*digit '9'*	**0.0498**

Table 4.2: Procrustes error values for different digits of the *MNIST* dataset, computed by comparing the original with **3-D** recreation via S-Isomap++.

4.6 Conclusion

The proposed S-Isomap++ algorithm allows for scalable nonlinear dimensionality reduction of streaming high-dimensional data. By allowing for the samples

to belong to multiple manifolds, or sampled non-uniformly from a single manifold, we have developed an algorithm that can be applied to a wide variety of practical settings. Moreover, the two-phase strategy for streaming Isomap, first proposed in [65], and adapted here for multiple manifold learning, allows us to scale a computationally intensive algorithm (Isomap) to arbitrarily large streams.

The ability to cluster data lying on multiple intersecting manifolds is a key innovation, proposed as the Tangent Manifold Clustering algorithm, allows us to automatically identify the number of underlying manifolds. One limitation of the method, however, is that it assumes that all manifolds are represented in the batch data set, which means that a novel manifold behavior that might appear subsequently in the stream, will not be learned. This issue will be studied in future research.

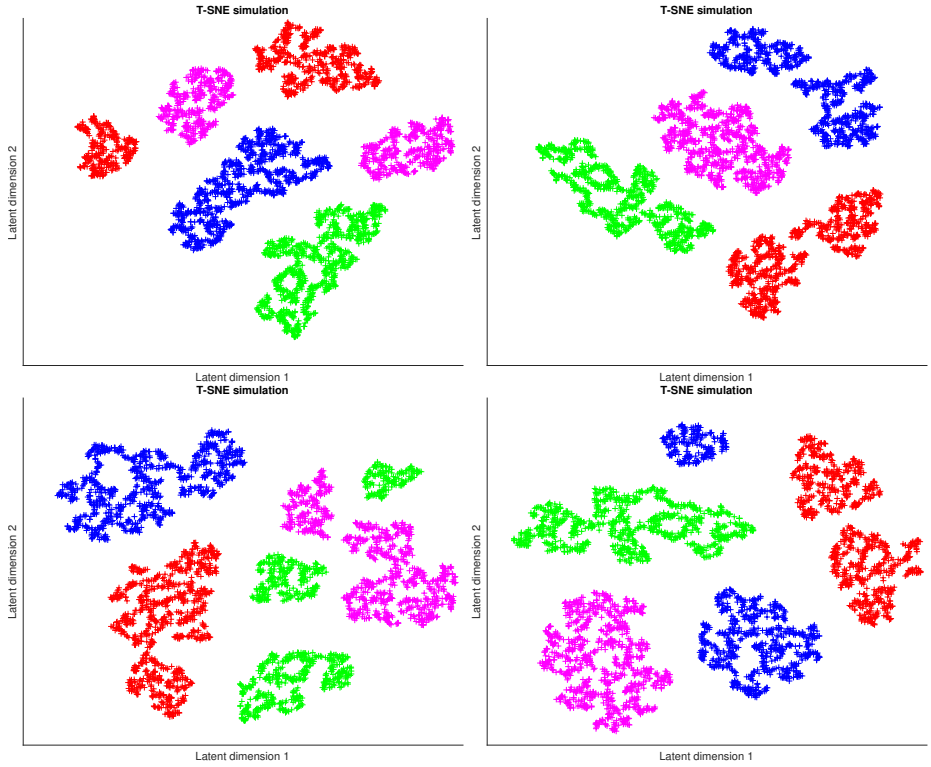
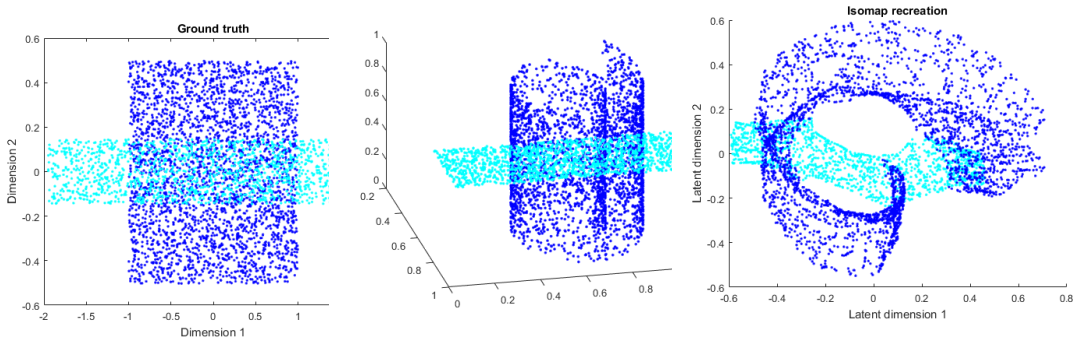


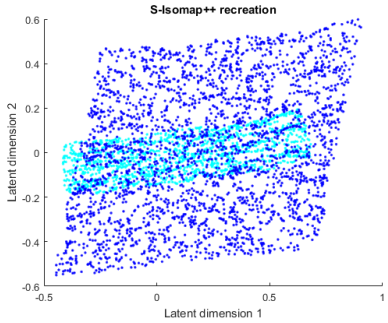
Figure 4.2: Multi-manifold *Swiss Roll* data set. **2-D** samples are embedded into **3-D** via the Euler Isometric mapping technique [65]. Considering the stochastic nature of the state-of-the-art t-SNE algorithm [49] and given its popularity for the visualization of high-dimensional datasets particularly in scenarios when the underlying manifold dimensionality (**d**) is $\leq$ **3**, here we demonstrate the results for the low-dimensional embeddings uncovered by four different simulations of the t-SNE algorithm on the same batch data set. The quality of low-dimensional embeddings uncovered by t-SNE is not good and the results vary quite a bit in every simulation i.e. there is a general *lack of stability*. We can observe that at least one of the manifolds is "broken" in every simulation. We note here even though t-SNE is not a "streaming" algorithm as such, however our motivation here is to demonstrate its capability in handling multiple manifolds. We show here that t-SNE does not quite capture the low-dimensional structure of the underlying manifold for this dataset. Refer to Figure 4.1 for results of S-Isomap++ as well as the low-dimensional ground-truth.



(a) Original Data in **2-D**

(b) Embedded Data (input) in **3-D**

(c) Isomap/M-Isomap Output



(d) Proposed S-Isomap++ Output

Figure 4.3: Multi-manifold *intersecting* data set. One set of **2-D** samples (blue) in (a) are embedded into **3-D** in (b) via the Euler Isometric mapping technique [65]. Second set (cyan) are embedded using a linear mapping. The reduction to **2-D** is obtained using: (c). Isomap/M-Isomap, and (d). the proposed S-Isomap++ algorithm. Both Isomap and M-Isomap give the same output because M-Isomap cannot handle intersecting manifolds and, thus, reverts to a single manifold scenario.

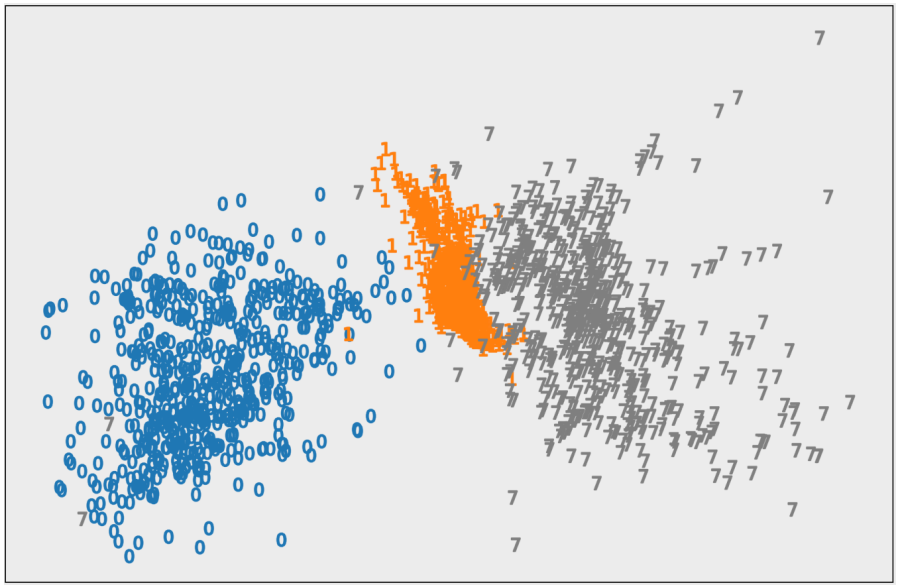


Figure 4.4: **2-D** reduction of a sample of images from the *MNIST* digits dataset. Real-world data generally lies near multiple manifolds and is usually separated by regions of low density.

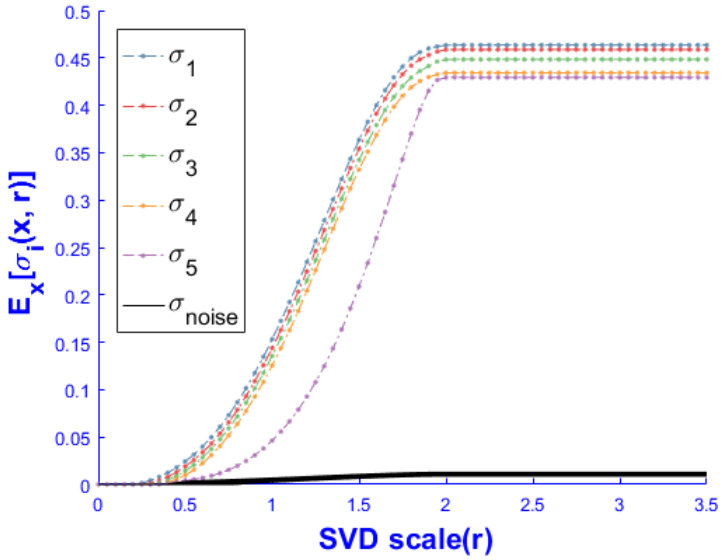


Figure 4.5: Multiscale SVD on a noisy $\mathbb{R}^5$ sphere embedded in $\mathbb{R}^{100}$ ambient space.

Algorithm 5 Similarity between tangent planes between points

function SIM($\mathcal{S}_i$, $\mathcal{S}_j$, **mode**)
    $\boldsymbol{\eta}_{i=1,2...k} \leftarrow$ extract($\mathcal{S}_i$)
    $\boldsymbol{\kappa}_{i=1,2...k} \leftarrow$ extract($\mathcal{S}_j$)

    if mode $=$ '$L1$' **then**
        $\mathbf{s} \leftarrow \frac{1}{\mathbf{k}} \sum_{i=1}^{k} \left| {\boldsymbol{\eta}_i}^T \boldsymbol{\kappa}_i \right|$

    else if mode $=$ '$L2$' **then**

        $\mathbf{s} \leftarrow \sqrt{\sum_{i=1}^{k} \frac{1}{k} ({\boldsymbol{\eta}_i}^T \boldsymbol{\kappa}_i)^2}$

    else if mode $=$ 'HG' **then**
        $\boldsymbol{\mathcal{M}}_\eta \leftarrow matrix(\boldsymbol{\eta}_{i=1,2...k})$
        $\boldsymbol{\mathcal{M}}_\kappa \leftarrow matrix(\boldsymbol{\kappa}_{i=1,2...k})$

        $\boldsymbol{\mathcal{M}} \leftarrow {\boldsymbol{\mathcal{M}}_\eta}^T \boldsymbol{\mathcal{M}}_\kappa$
        $\mathbf{s} \leftarrow |det(\boldsymbol{\mathcal{M}})|$
    end if

    return s
end function

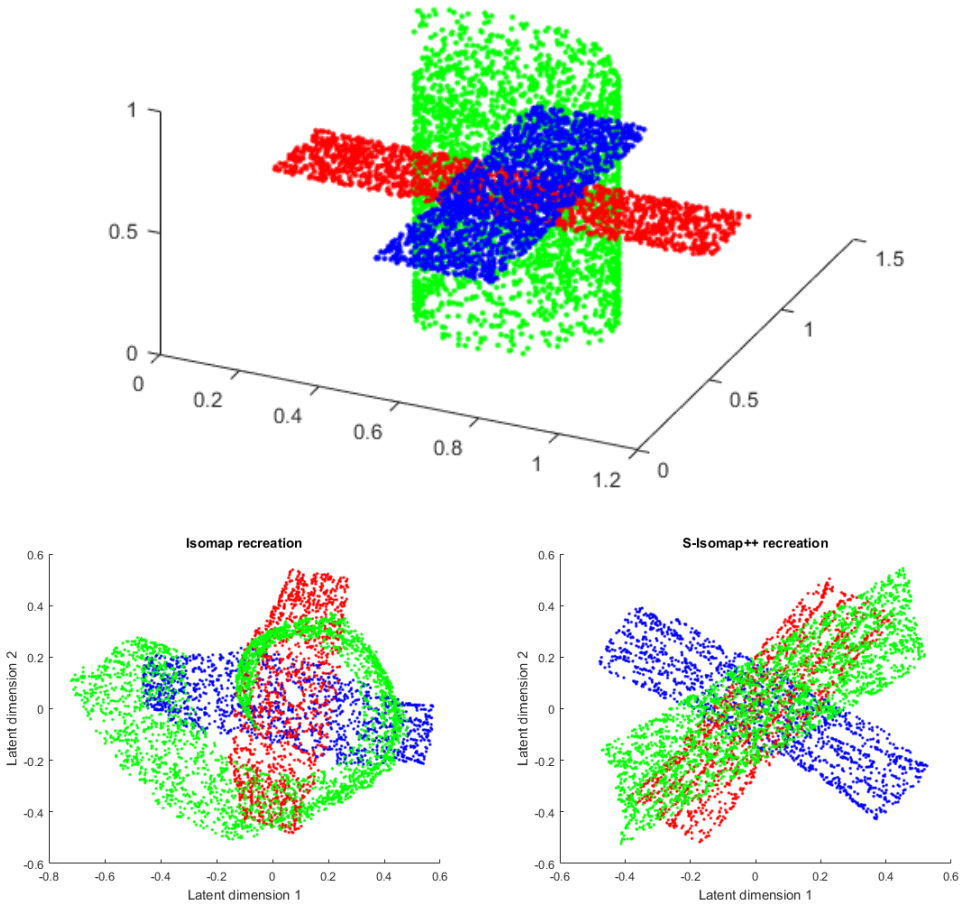


Figure 4.6: Top Left: Actual manifolds in $\mathbb{R}^3$ space, clustered to demonstrate individual manifolds, Top Right: Recreation by Isomap/M-Isomap, Bottom Row: Recreation by our approach, S-Isomap++.

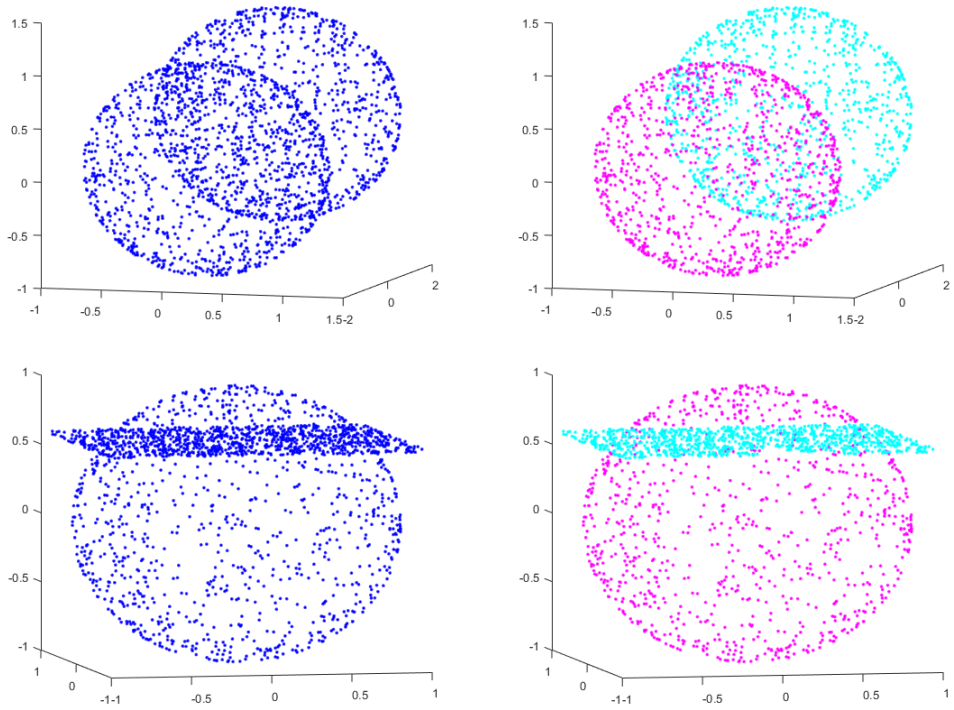


Figure 4.7: Left: Original datasets unclustered, Right: Clustered using the proposed tangent clustering method.

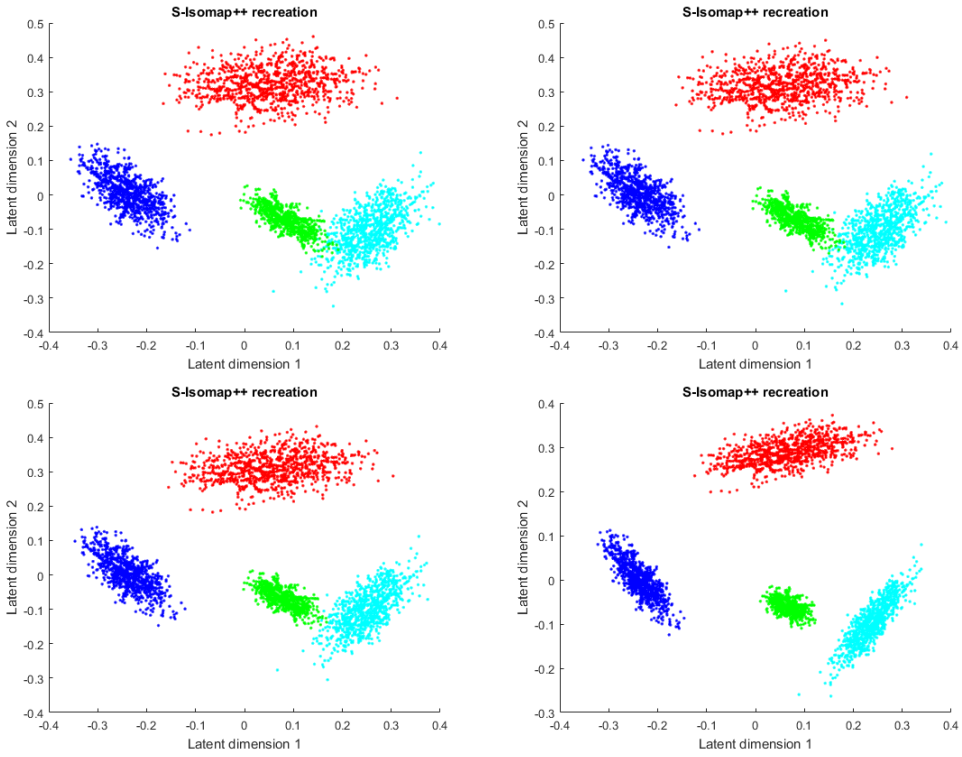


Figure 4.8: Effect of changing $\boldsymbol{\lambda}$. Top Left: $\boldsymbol{\lambda} = \mathbf{0.01}$, Top Right: $\boldsymbol{\lambda} = \mathbf{0.02}$, Bottom Left: $\boldsymbol{\lambda} = \mathbf{0.04}$, Bottom Right: $\boldsymbol{\lambda} = \mathbf{0.16}$

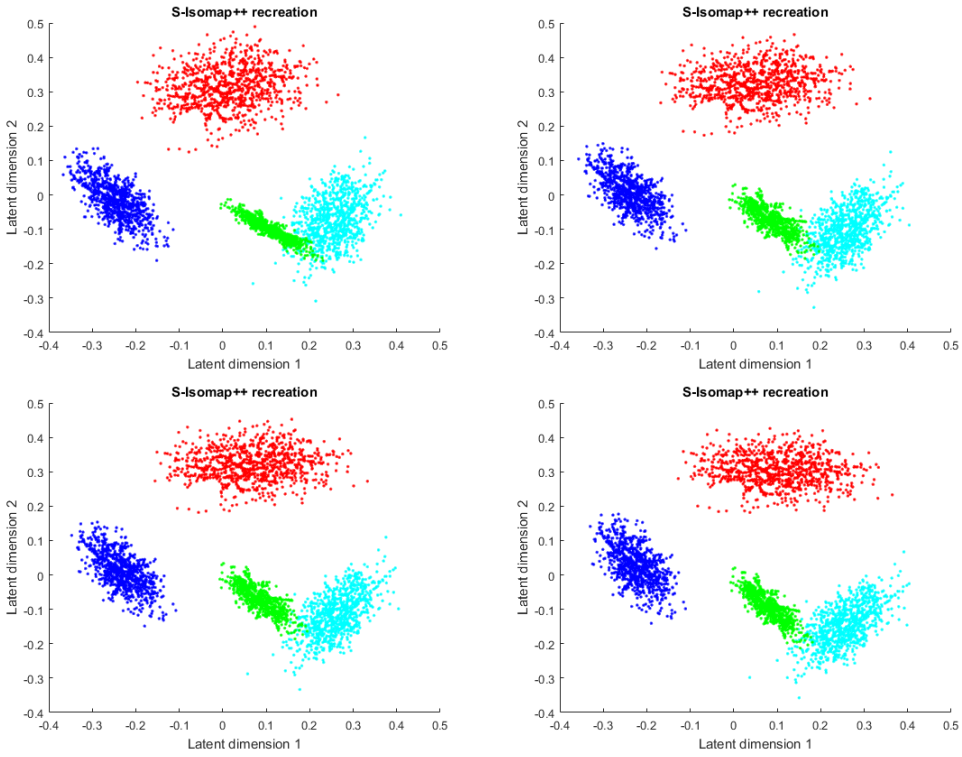


Figure 4.9: Effect of changing **k**. Top Left: $\mathbf{k} = \mathbf{8}$, Top Right: $\mathbf{k} = \mathbf{16}$, Bottom Left: $\mathbf{k} = \mathbf{24}$, Bottom Right: $\mathbf{k} = \mathbf{32}$

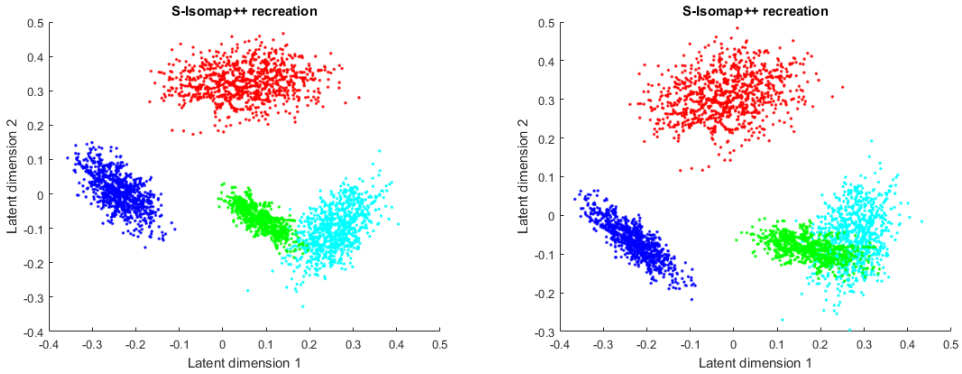


Figure 4.10: Effect of changing **l**. Left: $\mathbf{l} = \mathbf{1}$, Right: $\mathbf{l} = \mathbf{4}$

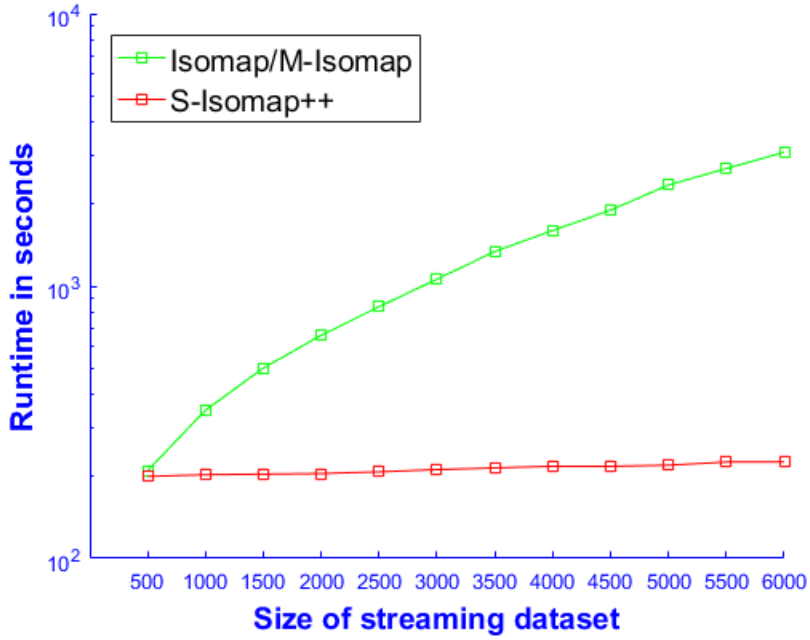


Figure 4.11: The results are in log scale and demonstrate the scalability of our proposed algorithm.

Chapter 5

GP-Isomap

5.1 Introduction

High-dimensional data is inherently difficult to explore and analyze, owing to the "curse of dimensionality" that render many statistical and machine learning techniques inadequate. In this context, *nonlinear spectral dimensionality reduction* (NLSDR) has proved to be an indispensable tool. Manifold learning based NLSDR methods, such as Isomap[74], Local Linear Embedding (LLE)[60], etc., assume that the distribution of the data in the high-dimensional observed space is not uniform and in reality, the data lies near a non-linear low-dimensional manifold embedded in the high-dimensional space.

If directly applied on streaming data, NLSDR methods have to recompute the entire manifold each time a new point is extracted from a stream. This quickly becomes computationally prohibitive but guarantees the best possible quality of the learned manifold given the data. To alleviate the computational problem, landmark [68] or general Out-of-Sample Extension methods [86] have been proposed. These techniques are still computationally expensive for practical applications. Recent streaming adaptations of NLSDR methods have relied on exact learning from a smaller batch of observations followed by approximate mapping of subsequent stream of observations [65]. Extensions to cases when the observations are sampled from multiple and possibly intersecting manifolds have been proposed as well [50].

However, existing streaming manifold learning methods [65, 50] assume that the underlying generative distribution is stationary over the stream, and are unable to detect when the distribution "drifts" or abruptly "shifts" away from the base, resulting in incorrect low-dimensional mappings (See Fig. 5.1). We develop a methodology to identify such changes (drifts and shifts) in the stream properties and inform the streaming algorithm to update the base model.

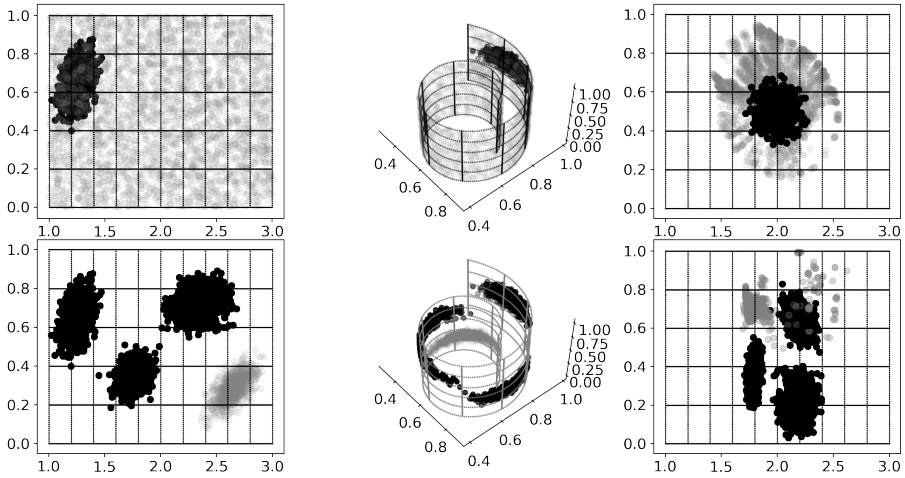


Figure 5.1: Impact of changes in the data distribution on streaming NLSDR. In the *top* panel, the true data lies on a **2-D** manifold (*top-left*) and the observed data is in $\mathbb{R}^3$ obtained by using the *Swiss Roll* transformation of the **2-D** data (*top-middle*). The streaming algorithm [65] uses a batch of samples from a **2-D** Gaussian (black), and maps streaming points sampled from a uniform distribution (gray). The streaming algorithm performs well on mapping the batch points to $\mathbb{R}^2$ but fails on the streaming points that "drift" away from the batch (*top-right*). In the *bottom* panel, the streaming algorithm [50] uses a batch of samples from three **2-D** Gaussians (black). The stream points are sampled from the three Gaussians and a new Gaussian (gray). The streaming algorithm performs well on mapping the batch points to $\mathbb{R}^2$ but fails on the streaming points that are "shifted" from the batch (*bottom-right*).

We employ a Gaussian Process (GP) [85] based adaptation of Isomap [74], a widely used NLSDR method, to process high throughput streams. The use of GP is enabled by a novel kernel that measures the relationship between a pair of observations along the manifold, and not in the original high-dimensional

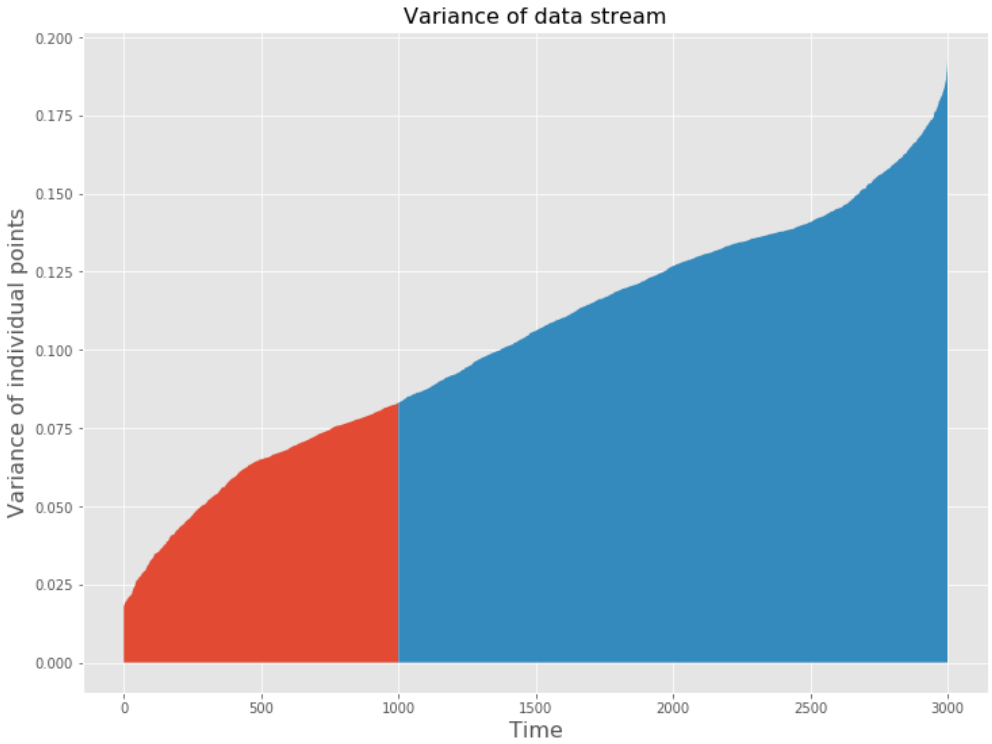


Figure 5.2: A single patch from the *Euler Isometric Swiss Roll* is used as the "batch" and streaming points sampled from a uniform distribution are mapped using GP-Isomap. We can observe the *smooth* and *gradual* increase in variance of predictions as the streaming points move further away from the batch which demonstrates that the usage of geodesic distances via the proposed kernel is apt when dealing with manifolds to mapping samples, rather than using Euclidean distance based kernels. See also Figure 5.4. Variance ($\mathbf{t} \leq \mathbf{1000}$, brown) is pretty small for streaming samples within/close to the Gaussian patch, while for samples ($\mathbf{t} \geq \mathbf{1000}$, blue), the predicted variance is much higher.

space. We prove that the low-dimensional representations inferred using the GP based method, *GP-Isomap*, are equivalent to the representations obtained using the state-of-art streaming Isomap methods [65, 50]. Additionally, we empirically show, on synthetic and real datasets, that the predictive variance associated with the GP predictions is an effective indicator of the changes (either gradual drifts or sudden shifts) in the underlying generative distribution, and can be employed to inform the algorithm to "re-learn" the core manifold.

5.2 Problem Statement and Preliminaries

We first formulate the NLSDR problem and provide background on Isomap and discuss its Out-of-Sample and Streaming Extensions [6, 65, 50, 43]. Additionally, we provide brief introduction to Gaussian Process (GP) analysis.

5.2.1 Nonlinear Spectral Dimensionality Reduction

Given high-dimensional data $\mathbf{Y} = \{\mathbf{y}_i\}_{i=1\ldots n}$, where $\mathbf{y}_i \in \mathbb{R}^{\mathbf{D}}$, the NLSDR problem is concerned with finding its corresponding low-dimensional representation $\mathbf{X} = \{\mathbf{x}_i\}_{i=1\ldots n}$, such that $\mathbf{x}_i \in \mathbb{R}^{\mathbf{d}}$, where $\mathbf{d} \ll \mathbf{D}$.

NLSDR methods assume that the data lies along a low-dimensional manifold embedded in a high-dimensional space, and exploit the global (Isomap [74], Minimum Volume Embedding [83]) or local (LLE [60], Laplacian Eigenmaps [4]) properties of the manifold to map each $\mathbf{y}_i$ to its corresponding $\mathbf{x}_i \in \mathbb{R}^{\mathbf{d}}$.

The Isomap algorithm [74] maps each $\mathbf{y}_i$ to its low-dimensional representation $\mathbf{x}_i$ in such a way that the geodesic distance along the manifold between any two points, $\mathbf{y}_i$ and $\mathbf{y}_j$, is as close to the Euclidean distance between $\mathbf{x}_i$ and $\mathbf{x}_j$ as possible. The geodesic distance is approximated by computing the shortest path between the two points using the k-nearest neighbor graph and is stored in the geodesic distance matrix $\mathbf{G} = \{\mathbf{g}_{i,j}\}_{1\leq i,j\leq n}$, where $\mathbf{g}_{i,j}$ is the geodesic distance between the points $\mathbf{y}_i$ and $\mathbf{y}_j$. $\widetilde{\mathbf{G}} = \{\mathbf{g}_{i,j}^2\}_{1\leq i,j\leq n}$ contains squared geodesic distance values. The Isomap algorithm recovers $\mathbf{x}_i$ by using the classical Multi Dimensional Scaling (MDS) on $\widetilde{\mathbf{G}}$. Let $\mathbf{B}$ be the inner product matrix between different $\mathbf{x}_i$. $\mathbf{B}$ can be retrieved as $\mathbf{B} = -\mathbf{H}\widetilde{\mathbf{G}}\mathbf{H}/2$ by assuming $\sum_{i=1}^{n} \mathbf{x}_i = 0$, where $\mathbf{H} = \{\mathbf{h}_{i,j}\}_{1\leq i,j\leq n}$ and $\mathbf{h}_{i,j} = \boldsymbol{\delta}_{i,j} - 1/\mathbf{n}$, where $\boldsymbol{\delta}_{i,j}$ is the Kronecker delta. Isomap uncovers $\mathbf{X}$ such that $\mathbf{X}^T\mathbf{X}$ is as close to $\mathbf{B}$ as possible. This is achieved by $\mathbf{X} = \{\sqrt{\boldsymbol{\lambda}_1}\mathbf{q}_1\ \sqrt{\boldsymbol{\lambda}_2}\mathbf{q}_2\ \ldots\ \sqrt{\boldsymbol{\lambda}_d}\mathbf{q}_d\}^T$ where $\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2 \ldots \boldsymbol{\lambda}_d$ are the $\mathbf{d}$ largest eigenvalues of $\mathbf{B}$ and $\mathbf{q}_1, \mathbf{q}_2 \ldots \mathbf{q}_d$ are the corresponding eigenvectors.

To measure error between the true, underlying low-dimensional representation to that uncovered by NLSDR methods, Procrustes analysis [16] is typically used. Procrustes analysis involves aligning two matrices, $\mathbf{A}$ and $\mathbf{B}$, by finding the optimal translation $\mathbf{t}$, rotation $\mathbf{R}$, and scaling $\mathbf{s}$ that minimizes the *Frobenius*

norm between the two aligned matrices, i.e.,:

$$\epsilon_{\text{Proc}}(\mathbf{A}, \mathbf{B}) = \min_{\mathbf{R},\mathbf{t},\mathbf{s}} \|\mathbf{sRB} + \mathbf{t} - \mathbf{A}\|_{\text{F}}$$

The above optimization problem has a closed form solution obtained by performing Singular Value Decomposition (SVD) of $\mathbf{AB}^T$ [16]. Consequently, one of the properties of Procrustes analysis is that $\epsilon_{\text{Proc}}(\mathbf{A}, \mathbf{B}) = 0$ when $\mathbf{A} = \mathbf{sRB} + \mathbf{t}$ i.e. when one of the matrices is a scaled, translated and/or rotated version of the other, which we leverage upon in this work.

5.2.2 Streaming Isomap

Given that the Isomap algorithm has a complexity of $\mathcal{O}(\mathbf{n}^3)$ (where $\mathbf{n}$ - size of data), recomputing the manifold is computationally too expensive or impractical to use in a streaming setting. Incremental techniques have been proposed in the past [43, 65], which can efficiently process the new streaming points, without affecting the quality of the embedding significantly.

The S-Isomap algorithm relies on the observation that a stable manifold can be learnt using only a fraction of the stream (denoted as the batch dataset $\boldsymbol{\mathcal{B}}$), and the remaining part of stream (denoted as the stream dataset $\boldsymbol{\mathcal{S}}$) can be mapped to the manifold in a significantly less costly manner. This can be justified by considering the convergence of eigenvectors and eigenvalues of $\mathbf{B}$, as the number of points in the batch increase [66]. In particular, the bounds on the convergence error for a similar NLSDR method, i.e., kernel PCA, is shown to be inversely proportional to the batch size [66]. Similar arguments can be made for Isomap, by considering the equivalence between Isomap and Kernel PCA [25, 6]. This relationship has also been empirically shown for multiple data sets [65].

The S-Isomap algorithm computes the low-dimensional representation for each new point i.e. $\mathbf{x}_{n+1} \in \mathbb{R}^{\mathbf{d}}$ by solving a least-squares problem formulated by matching the dot product of the new point with the low-dimensional embedding of the points in the batch dataset $\mathbf{X}$, computed using Isomap, to the normalized squared geodesic distances vector $\mathbf{f}$. The least-squares problem has

the following form:

$$\mathbf{X}^{\mathbf{T}}\mathbf{x}_{n+1} = \mathbf{f}$$

where[1]

$$\mathbf{f}_i \simeq \frac{1}{2}\left(\frac{1}{\mathbf{n}}\sum_j \mathbf{g}_{i,j}^2 - \mathbf{g}_{i,n+1}^2\right) \tag{5.2}$$

5.2.3 Handling Multiple Manifolds

In the ideal case, when manifolds are densely sampled and sufficiently separated, clustering can be performed before applying NLSDR techniques [56, 20], by choosing an appropriate local neighborhood size so as not to include points from other manifolds and still be able to capture the local geometry of the manifold. However, if the manifolds are close or intersecting, such methods typically fail.

The S-Isomap++ [50] algorithm overcomes limitations of the S-Isomap algorithm and extends it to be able to deal with multiple manifolds. It uses the notion of Multi-scale SVD [48] to define tangent manifold planes at each data point, computed at the appropriate scale, and compute similarity in a local neighborhood. Additionally, it includes a novel manifold tangent clustering algorithm to be able to deal with the above issue of clustering manifolds which are close and in certain scenarios, intersecting, using these tangent manifold planes. After initially clustering the high-dimensional batch dataset, the algorithm applies NLSDR on each manifold individually and eventually "stitches" them together in a global ambient space by defining transformations which can map points from the individual low-dimensional manifolds to the global space.

[1]Note that the Incremental Isomap algorithm [43] has a slightly different formulation where

$$\mathbf{f}_i \simeq \frac{1}{2}\left(\frac{1}{\mathbf{n}}\sum_j \mathbf{g}_{i,j}^2 - \frac{1}{\mathbf{n}^2}\sum_{l,m} \mathbf{g}_{l,m}^2\right) + \frac{1}{2}\left(\frac{1}{\mathbf{n}}\sum_j \mathbf{g}_{j,n+1}^2 - \mathbf{g}_{i,n+1}^2\right) \tag{5.1}$$

The S-Isomap algorithm assumes that the data stream draws from an uniformly sampled, unimodal distribution $p(\mathbf{x})$ and that the stream $\boldsymbol{\mathcal{S}}$ and the batch $\boldsymbol{\mathcal{B}}$ datasets get generated from $p(\mathbf{x})$. Additionally it assumes that the manifold has stabilized i.e. $|\boldsymbol{\mathcal{B}}| = \mathbf{n}$ is large enough. Using these assumptions in (5.1) above, we have that $\left(\frac{1}{\mathbf{n}}\sum_j \mathbf{g}_{j,n+1}^2 - \frac{1}{\mathbf{n}^2}\sum_{l,m} \mathbf{g}_{l,m}^2\right) = \boldsymbol{\epsilon} \simeq 0$ i.e. the expectation of squared geodesic distances for points in the batch dataset $\boldsymbol{\mathcal{B}}$ is close to those for points in the stream dataset $\boldsymbol{\mathcal{S}}$. The line of reasoning for this follows from [30]. Thus (5.1) simplifies to (5.2).

However, S-Isomap++ can only detect manifolds which it encounters in its batch learning phase and not those which it might encounter in the streaming phase.

5.2.4 Gaussian Process Regression

Let us assume that we are learning a probabilistic regression model to obtain the prediction at a given test input, $\mathbf{y}$, using a nonlinear and latent function, $\phi()$. Assuming[2] $\mathbf{d} = 1$, the observed output, $\mathbf{x}$, is related to the input as:

$$\mathbf{x} = \phi(\mathbf{y}) + \varepsilon, \text{ where, } \varepsilon \sim \mathcal{N}(0, \sigma_n^2) \tag{5.3}$$

Given a training set of inputs, $\mathbf{Y} = \{\mathbf{y}_i\}_{i=1\ldots n}$ and corresponding outputs, $\mathbf{X} = \{\mathbf{x}_i\}_{i=1\ldots n}$, the Gaussian Process Regression (GPR) model assumes a GP prior on the latent function values, i.e., $\phi(\mathbf{y}) \sim GP(\mathbf{m}(\mathbf{y}), \mathbf{k}(\mathbf{y}, \mathbf{y}'))$, where $\mathbf{m}(\mathbf{y})$ is the mean of $\phi(\mathbf{y})$ and $\mathbf{k}(\mathbf{y}, \mathbf{y}')$ is the covariance between any two evaluations of $\phi()$, i.e, $m(\mathbf{y}) = \mathbb{E}[\phi(\mathbf{y})]$ and $\mathbf{k}(\mathbf{y}, \mathbf{y}') = \mathbb{E}[(\phi(\mathbf{y}) - \mathbf{m}(\mathbf{y}))(\phi(\mathbf{y}') - \mathbf{m}(\mathbf{y}'))]$. Here we use a zero-mean function ($\mathbf{m}(\mathbf{y}) = 0$), though other functions could be used as well. The GP prior states that any finite collection of the latent function evaluations are jointly Gaussian, i.e.,

$$\phi(\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n) \sim \mathcal{N}(\mathbf{0}, \mathbf{K}) \tag{5.4}$$

where the ij^{th} entry of the $\mathbf{n} \times \mathbf{n}$ covariance matrix, $\mathbf{K}$, is given by $\mathbf{k}(\mathbf{y}_i, \mathbf{y}_j)$. The GPR model uses (5.3) and (5.4) to obtain the predictive distribution at a new test input, $\mathbf{y}_{n+1}$, as a Gaussian distribution with following mean and variance:

$$\mathbb{E}[\mathbf{x}_{n+1}] = \mathbf{k}_{n+1}^\top (K + \sigma_n^2 \mathbf{I})^{-1} \mathbf{X} \tag{5.5}$$

$$var[\mathbf{x}_{n+1}] = \mathbf{k}(\mathbf{y}_{n+1}, \mathbf{y}_{n+1}) - \mathbf{k}_{n+1}^\top (\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1} \mathbf{k}_{n+1} + \sigma_n^2 \tag{5.6}$$

where $\mathbf{k}_{n+1}$ is a $n \times 1$ vector with $\mathbf{i}^{th}$ value as $\mathbf{k}(\mathbf{y}_{n+1}, \mathbf{y}_i)$.

The kernel function, $\mathbf{k}()$, specifies the covariance between function values, $\phi(\mathbf{y}_i)$ and $\phi(\mathbf{y}_j)$, as a function of the corresponding inputs, $\mathbf{y}_i$ and $\mathbf{y}_j$. A popular

[2]For vector-valued outputs, i.e., $\mathbf{x} \in \mathbb{R}^{\mathbf{d}}$, one can consider $\mathbf{d}$ independent models.

choice is the *squared exponential* kernel, which has been used in this work:

$$\mathbf{k}(\mathbf{y}_i, \mathbf{y}_j) = \boldsymbol{\sigma}_s^2 \exp\left[-\frac{(\mathbf{y}_i - \mathbf{y}_j)^2}{2\boldsymbol{\ell}^2}\right] \tag{5.7}$$

where $\boldsymbol{\sigma}_s^2$ is the signal variance and $\boldsymbol{\ell}$ is the length scale. The quantities $\boldsymbol{\sigma}_s^2$, $\boldsymbol{\ell}$, and $\boldsymbol{\sigma}_n^2$ (from (5.3)) are the hyper-parameters of the model and can be estimated by maximizing the marginal log-likelihood of the observed data ($\mathbf{Y}$ and $\mathbf{X}$) under the GP prior assumption.

One can observe that predictive mean, $\mathbb{E}[\mathbf{x}_{n+1}]$ in (5.5) can be written as an inner product, i.e.,:

$$\mathbb{E}[\mathbf{x}_{n+1}] = \boldsymbol{\beta}^\top \mathbf{k}_{n+1} \tag{5.8}$$

where $\boldsymbol{\beta} = (\mathbf{K} + \boldsymbol{\sigma}_n^2 \mathbf{I})^{-1}\mathbf{X}$. We will utilize this form in subsequent proofs.

5.3 Methodology

The proposed GP-Isomap algorithm follows a two-phase strategy (similar to the S-Isomap and S-Isomap++), where exact manifolds are learnt from an initial batch $\boldsymbol{\mathcal{B}}$, and subsequently a computationally inexpensive mapping procedure processes the remainder of the stream. To handle multiple manifolds, the batch data $\boldsymbol{\mathcal{B}}$ is first clustered via manifold tangent clustering or other standard techniques. Exact Isomap is applied on each cluster. The resulting low-dimensional data for the clusters is then "stitched" together to obtain the low-dimensional representation of the input data. The difference from the past methods is the mapping procedure which uses GPR to obtain the predictions for the low-dimensional mapping (See (5.5)). At the same time, the associated predictive variance (See (5.6)) is used to detect changes in the underlying distribution.

The overall GP-Isomap algorithm is outlined in Algorithm 6 and takes a batch data set, $\boldsymbol{\mathcal{B}}$ and the streaming data, $\boldsymbol{\mathcal{S}}$ as inputs, along with other parameters. The processing is split into two phases: a batch learning phase (Lines 1–15) and a streaming phase (Lines 16–32), which are described later in this section.

Algorithm 6 GP-Isomap

Require: Batch dataset: $\boldsymbol{\mathcal{B}}$, Streaming dataset: $\boldsymbol{\mathcal{S}}$; Parameters: $\boldsymbol{\epsilon}$, $\mathbf{k}$, $\mathbf{l}$, $\boldsymbol{\lambda}$, $\boldsymbol{\sigma}_t$, $\mathbf{n}_s$

Ensure: $\boldsymbol{\mathcal{Y}}_{\mathcal{S}}$: low-dimensional representation for $\boldsymbol{\mathcal{S}}$

▷ Batch Phase

$\boldsymbol{\mathcal{C}}_{i=1,2\ldots p} \leftarrow$ FIND_CLUSTERS($\boldsymbol{\mathcal{B}}$, $\boldsymbol{\epsilon}$)

$\boldsymbol{\xi}_s \leftarrow \emptyset$

for $1 \leq \mathbf{i} \leq \mathbf{p}$ **do**

  $\boldsymbol{\mathcal{LDE}}_i, \boldsymbol{\mathcal{G}}_i \leftarrow$ ISOMAP($\boldsymbol{\mathcal{C}}_i$)

end for

for $1 \leq \mathbf{i} \leq \mathbf{p}$ **do**

  $\phi_i^{\mathcal{GP}} \leftarrow$ ESTIMATE($\boldsymbol{\mathcal{LDE}}_i, \boldsymbol{\mathcal{G}}_i$)

end for

$\boldsymbol{\xi}_s \leftarrow \bigcup_{i=1}^{p} \bigcup_{j=i+1}^{p} \text{NN}(\boldsymbol{\mathcal{C}}_i, \boldsymbol{\mathcal{C}}_j, \mathbf{k}) \cup \text{FN}(\boldsymbol{\mathcal{C}}_i, \boldsymbol{\mathcal{C}}_j, \mathbf{l})$

$\boldsymbol{\mathcal{GE}}_s \leftarrow \text{MDS}(\boldsymbol{\xi}_s)$

for $1 \leq \mathbf{j} \leq \mathbf{p}$ **do**

  $\boldsymbol{\mathcal{I}} \leftarrow \boldsymbol{\xi}_s \cap \boldsymbol{\mathcal{C}}_j$

  $\boldsymbol{\mathcal{A}} \leftarrow \begin{bmatrix} \boldsymbol{\mathcal{LDE}}_j^{\mathcal{I}} \\ \mathbf{e}^T \end{bmatrix}$

  $\boldsymbol{\mathcal{R}}_i, \mathbf{t}_i \leftarrow \boldsymbol{\mathcal{GE}}_{\mathcal{I},s} \times \boldsymbol{\mathcal{A}}^T \left(\boldsymbol{\mathcal{A}}\boldsymbol{\mathcal{A}}^T + \boldsymbol{\lambda}\mathbf{I}\right)^{-1}$

end for

▷ Streaming Phase

$\boldsymbol{\mathcal{S}}_u \leftarrow \emptyset$

for $\mathbf{s} \in \boldsymbol{\mathcal{S}}$ **do**

  if $|\boldsymbol{\mathcal{S}}_u| \geq \mathbf{n}_s$ **then**

    $\boldsymbol{\mathcal{Y}}_u \leftarrow$ Re-run Batch Phase with $\boldsymbol{\mathcal{B}} \leftarrow \boldsymbol{\mathcal{B}} \cup \boldsymbol{\mathcal{S}}_u$

    $\boldsymbol{\mathcal{Y}}_{\mathcal{S}} \leftarrow \boldsymbol{\mathcal{Y}}_{\mathcal{S}} \cup \boldsymbol{\mathcal{Y}}_u$

  end if

  for $1 \leq \mathbf{i} \leq \mathbf{p}$ **do**

    $\boldsymbol{\mu}_i, \boldsymbol{\sigma}_i \leftarrow$ GP_REG($\mathbf{s}, \boldsymbol{\mathcal{LDE}}_i, \boldsymbol{\mathcal{G}}_i, \phi_i^{GP}$)

  end for

  $\mathbf{j} \leftarrow \text{argmin}_i \, |\boldsymbol{\sigma}_i|$

  if $\boldsymbol{\sigma}_j \leq \boldsymbol{\sigma}_t$ **then**

    $\mathbf{y}_s \leftarrow \boldsymbol{\mathcal{R}}_j \boldsymbol{\mu}_j + \mathbf{t}_j$

    $\boldsymbol{\mathcal{Y}}_{\mathcal{S}} \leftarrow \boldsymbol{\mathcal{Y}}_{\mathcal{S}} \cup \mathbf{y}_s$

  else

    $\boldsymbol{\mathcal{S}}_u \leftarrow \boldsymbol{\mathcal{S}}_u \cup \mathbf{s}$

  end if

end for

return $\boldsymbol{\mathcal{Y}}_{\mathcal{S}}$

5.3.1 Kernel Function

The proposed GP-Isomap algorithm uses a novel geodesic distance based kernel function defined as:

$$\mathbf{k}(\mathbf{y}_i, \mathbf{y}_j) = \boldsymbol{\sigma}_s^2 \exp\left(-\frac{\mathbf{b}_{i,j}}{2\boldsymbol{\ell}^2}\right) \tag{5.9}$$

where $\mathbf{b}_{i,j}$ is the ij^{th} entry of the normalized geodesic distance matrix, $\mathbf{B}$, as discussed in Sec. 5.2.1, $\boldsymbol{\sigma}_s^2$ is the signal variance (whose value we fix as **1.0** in this work) and $\boldsymbol{\ell}$ is the length scale hyper-parameter. Thus the kernel matrix $\mathbf{K}$

can be written as:

$$\mathbf{K} = \exp\left(-\frac{\mathbf{B}}{2\ell^2}\right) \tag{5.10}$$

This kernel function plays a key role in using the GPR model for mapping streaming points on the learnt manifold, by measuring similarity along the low-dimensional manifold, instead of the original space ($\mathbb{R}^{\mathbf{D}}$), as is typically done in GPR based solutions. The matrix, $\mathbf{B}$, is positive semi-definite[3] (PSD), due to the double mean centering done to squared geodesic distance matrix $\widetilde{\mathbf{G}}$. Consequently, we note that the kernel matrix, $\mathbf{K}$, is positive definite (refer (5.11) below).

Using lemmas 5.8.1 and 5.8.2 defined in Sec. 5.8, the novel kernel we propose can be written as

$$\mathbf{K}(\mathbf{x},\mathbf{y}) = \mathbf{I} + \sum_{i=1}^{d}\left[\exp\left(-\frac{\boldsymbol{\lambda}_i}{2\ell^2}\right) - 1\right]\mathbf{q}_i\mathbf{q}_i^T = \mathbf{I} + \mathbf{Q}\widetilde{\boldsymbol{\Lambda}}\mathbf{Q}^{\mathbf{T}} \tag{5.11}$$

where $\widetilde{\boldsymbol{\Lambda}} = \begin{bmatrix} \left[\exp\left(-\frac{\boldsymbol{\lambda}_1}{2\ell^2}\right) - 1\right] & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \left[\exp\left(-\frac{\boldsymbol{\lambda}_d}{2\ell^2}\right) - 1\right] \end{bmatrix}$ and $\{\boldsymbol{\lambda}_i, \mathbf{q}_i\}_{i=1\ldots d}$ are eigenvalue/eigenvector pairs of $\mathbf{B}$ as discussed in Sec. 5.2.1.

5.3.2 Batch Learning

The batch learning phase consists of these tasks :

5.3.2.1 Clustering.

The first step in the batch phase involves clustering of the batch dataset $\boldsymbol{\mathcal{B}}$ into $\mathbf{p}$ individual clusters which represent the manifolds. In case, $\boldsymbol{\mathcal{B}}$ contains a single cluster, the algorithm can correctly detect it. (Line 1)

[3] Actually $\mathbf{B}$ is not always guaranteed to be PSD. [12] use a additive constant to make it PSD. Equation (5.11) via exponentiation introduces the identity which functions similarly to an additive constant.

5.3.2.2 Dimension Reduction.

Subsequently, full Isomap is executed on each of the **p** individual clusters to get low-dimensional representations $\boldsymbol{\mathcal{LDE}}_{i=1,2...p}$ of the data points belonging to each individual cluster. (Lines 3–5)

5.3.2.3 Hyper-parameter Estimation.

The geodesic distance matrix for the points in the $\mathbf{i}^{\text{th}}$ manifold $\boldsymbol{\mathcal{G}}_i$ and the corresponding low-dimensional representation $\boldsymbol{\mathcal{LDE}}_i$, are fed to the GP model for each of the **p** manifolds, to perform hyper-parameter estimation, which outputs $\{\phi_i^{GP}\}_{i=1,2...p}$. (Lines 6–8)

5.3.2.4 Learning Mapping to Global Space.

The low-dimensional embedding uncovered for each of the manifolds can be of different dimensionalities. Consequently, a mapping to a unified global space is needed. To learn this mapping, a support set $\boldsymbol{\xi}_s$ is formulated, which contains the **k** pairs of nearest points and **l** pairs of farthest points, between each pair of manifolds. Subsequently, MDS is executed on this support set $\boldsymbol{\xi}_s$ to uncover its low-dimensional representation $\boldsymbol{\mathcal{GE}}_s$. Individual scaling and translation factors $\{\boldsymbol{\mathcal{R}}_i, \mathbf{t}_i\}_{i=1,2...p}$ are learnt via solving a least squares problem involving $\boldsymbol{\xi}_s$, which map points from each of the individual manifolds to the global space. (Lines 9–15)

5.3.3 Stream Processing

In the streaming phase, each sample **s** in the stream set $\boldsymbol{\mathcal{S}}$ is embedded using each of the **p** GP models to evaluate the prediction $\boldsymbol{\mu}_i$, along with the variance $\boldsymbol{\sigma}_i$ (Lines 22–24). The manifold with the smallest variance get chosen to embed the sample **s** into, using the corresponding scaling $\boldsymbol{\mathcal{R}}_j$ and translation factor $\mathbf{t}_j$, provided $min_i\,|\boldsymbol{\sigma}_i|$ is within the allowed threshold $\boldsymbol{\sigma}_t$ (Lines 25–28), otherwise sample **s** is added to the unassigned set $\boldsymbol{\mathcal{S}}_u$ (Lines 29–31). When the size of unassigned set $\boldsymbol{\mathcal{S}}_u$ exceeds certain threshold $\mathbf{n}_s$, we add them to the batch dataset and

re-learn the base manifold (Line 18–21). The assimilation of the new points in the batch maybe done more efficiently in an incremental manner.

5.3.4 Complexity

The runtime complexity of our proposed algorithm is dominated by the GP regression step as well as the Isomap execution step, both of which have $\mathcal{O}(\mathbf{n}^3)$ complexity, where $\mathbf{n}$ is the size of the batch dataset $\boldsymbol{\mathcal{B}}$. This is similar to the S-Isomap and S-Isomap++ algorithms, that also have a runtime complexity of $\mathcal{O}(\mathbf{n}^3)$. The stream processing step is $\mathcal{O}(\mathbf{n})$ for each incoming streaming point. The space complexity of GP-Isomap is dominated by $\mathcal{O}(\mathbf{n}^2)$. This is because each of the samples of the stream set $\boldsymbol{\mathcal{S}}$ get processed separately. Thus, the space requirement as well as runtime complexity does not grow with the size of the stream, which makes the algorithm appealing for handling high-volume streams.

5.4 Theoretical Analysis

In this section, we first state the main result and subsequently prove it using results from lemmas stated later in Sec. 5.8.

Theorem 5.4.1. *The prediction $\boldsymbol{\tau}_{GP}$ of our proposed approach, GP-Isomap is equivalent to the prediction $\boldsymbol{\tau}_{ISO}$ of S-Isomap i.e. the Procrustes Error $\boldsymbol{\epsilon}_{Proc}(\boldsymbol{\tau}_{GP}, \boldsymbol{\tau}_{ISO})$ between $\boldsymbol{\tau}_{GP}$ and $\boldsymbol{\tau}_{ISO}$ is $\mathbf{0}$.*

Proof. The prediction of GP-Isomap is given by (5.8). Using Lemma 5.8.5, we demonstrated that

$$\boldsymbol{\beta} = \{\frac{\boldsymbol{\alpha}\sqrt{\boldsymbol{\lambda}}_1\mathbf{q}_1}{1+\boldsymbol{\alpha}\mathbf{c}_1}\frac{\boldsymbol{\alpha}\sqrt{\boldsymbol{\lambda}}_2\mathbf{q}_2}{1+\boldsymbol{\alpha}\mathbf{c}_2}\cdots\frac{\boldsymbol{\alpha}\sqrt{\boldsymbol{\lambda}}_d\mathbf{q}_d}{1+\boldsymbol{\alpha}\mathbf{c}_d}\} \tag{5.12}$$

The term $\mathbf{K}_*$ for GP-Isomap, using our novel kernel function evaluates to

$$\mathbf{K}_* = \exp\left(-\frac{\mathbf{G}_*^2}{2\ell^2}\right) \tag{5.13}$$

where $\mathbf{G}_*^2$ represents the vector containing the squared geodesic distances of $\mathbf{x_{n+1}}$ to $\mathbf{X}$ containing $\{\mathbf{x}_i\}_{i=1,2\ldots n}$.

Considering the above equation element-wise, we have that the $\mathbf{i}^{\text{th}}$ term of $\mathbf{K}_*$ equates to $\exp\left[-\frac{\mathbf{g}_{i,n+1}^2}{2\ell^2}\right]$. Using Taylor's series expansion we have,

$$\exp\left[-\frac{\mathbf{g}_{i,n+1}^2}{2\ell^2}\right] \simeq \left(1 - \frac{\mathbf{g}_{i,n+1}^2}{2\ell^2}\right) \text{ for large } \ell \tag{5.14}$$

The prediction by the S-Isomap is given by (5.2) as follows :-

$$\boldsymbol{\tau}_{\text{ISO}} = \{\sqrt{\boldsymbol{\lambda}}_1\mathbf{q}_1^T\mathbf{f}\ \sqrt{\boldsymbol{\lambda}}_2\mathbf{q}_2^T\mathbf{f}\ \ldots\ \sqrt{\boldsymbol{\lambda}}_d\mathbf{q}_d^T\mathbf{f}\}^T \tag{5.15}$$

where $\mathbf{f} = \{\mathbf{f}_i\}$ is as defined by (5.2).

Rewriting (5.2) we have,

$$\mathbf{f}_i \simeq \frac{1}{2}\left(\boldsymbol{\gamma} - \mathbf{g}_{i,n+1}^2\right) \tag{5.16}$$

where $\boldsymbol{\gamma} = \left(\frac{1}{n}\sum_j \mathbf{g}_{i,j}^2\right)$ is a constant with respect to $\mathbf{x}_{n+1}$, since it depends only on squared geodesic distance values associated within the batch dataset $\mathcal{B}$ and $\mathbf{x}_{n+1}$ is part of the stream dataset $\mathcal{S}$.

We now consider the 1^{st} dimension of the predictions for GP-Isomap and S-Isomap only and demonstrate their equivalence via Procrustes Error. The analysis for the remaining dimensions follows a similar line of reasoning.

Thus for the 1^{st} dimension, using (5.16) the S-Isomap prediction is

$$\begin{aligned}
\boldsymbol{\tau}_{\text{ISO1}} &= \sqrt{\boldsymbol{\lambda}}_1\mathbf{q}_1^T\mathbf{f} \\
&= \sqrt{\boldsymbol{\lambda}}_1\sum_{i=1}^{n}\mathbf{q}_{1,i}\left(\frac{1}{2}\left(\boldsymbol{\gamma} - \mathbf{g}_{i,n+1}^2\right)\right) \\
&= \frac{\sqrt{\boldsymbol{\lambda}}_1}{2}\sum_{i=1}^{n}\mathbf{q}_{1,i}\left(\boldsymbol{\gamma} - \mathbf{g}_{i,n+1}^2\right)
\end{aligned} \tag{5.17}$$

Similarly using Lemma 5.8.5, (5.13) and (5.14), we have that the $\mathbf{1}^{\text{st}}$ dimension

for GP-Isomap prediction is given by,

$$\begin{aligned} \boldsymbol{\tau}_{\mathrm{GP1}} &= \frac{\boldsymbol{\alpha}\sqrt{\boldsymbol{\lambda}_1}\mathbf{q}_1^T}{1+\boldsymbol{\alpha}\mathbf{c}_1}\mathbf{K}_* \\ &= \frac{\boldsymbol{\alpha}\sqrt{\boldsymbol{\lambda}_1}}{1+\boldsymbol{\alpha}\mathbf{c}_1}\sum_{i=1}^{n}\mathbf{q}_{1,i}\big(1-\frac{\mathbf{g}_{i,n+1}^2}{2\boldsymbol{\ell}^2}\big) \end{aligned} \tag{5.18}$$

We can observe that $\boldsymbol{\tau}_{\mathrm{GP1}}$ is a scaled and translated version of $\boldsymbol{\tau}_{\mathrm{ISO1}}$. Similarly for each of the dimensions ($1 \leq \mathbf{i} \leq \mathbf{d}$), the prediction for the GP-Isomap $\boldsymbol{\tau}_{\mathrm{GP}i}$ can be shown to be a scaled, rotated and translated version of the prediction for the S-Isomap $\boldsymbol{\tau}_{\mathrm{ISO}i}$. These individual scaling $\mathbf{s}_i$, rotation $\mathbf{r}_i$ and translation $\mathbf{t}_i$ factors can be represented together by single collective scaling $\mathbf{S}$, rotation $\mathbf{R}$ and translation $\mathbf{T}$ factors. Consequently, the Procrustes Error $\boldsymbol{\epsilon}_{\mathrm{Proc}}\big(\boldsymbol{\tau}_{\mathrm{GP}}, \boldsymbol{\tau}_{\mathrm{SI}}\big)$ is $\mathbf{0}$. (refer Sect. 5.2.1). ■

5.5 Results and Analysis

In this section, we demonstrate the ability of the predictive variance within GP-Isomap to identify changes in the underlying distribution in the data stream on synthetically generated datasets as well as on benchmark sensor data sets.

5.5.1 Results on Synthetic Data Sets

Swiss Roll datasets are typically used for evaluating manifold learning algorithms. To evaluate our method on sudden *concept-drift*, we use the *Euler Isometric Swiss Roll* dataset [65] consisting of four $\mathbb{R}^{\mathbf{2}}$ Gaussian patches having $\mathbf{n} = \mathbf{2000}$ points from each patch, chosen at random, which are embedded into $\mathbb{R}^{\mathbf{3}}$ using a nonlinear function $\boldsymbol{\psi}(\cdot)$. The points for each of the Gaussian modes are divided equally into training and test sets. To test incremental *concept-drift*, we use the single patch dataset, which consists of a single patch borrowed from the above, which we use as the training data, along with a uniform distribution of points for testing. Figures 5.1, 5.3 and 5.4 demonstrates our results on these datasets.

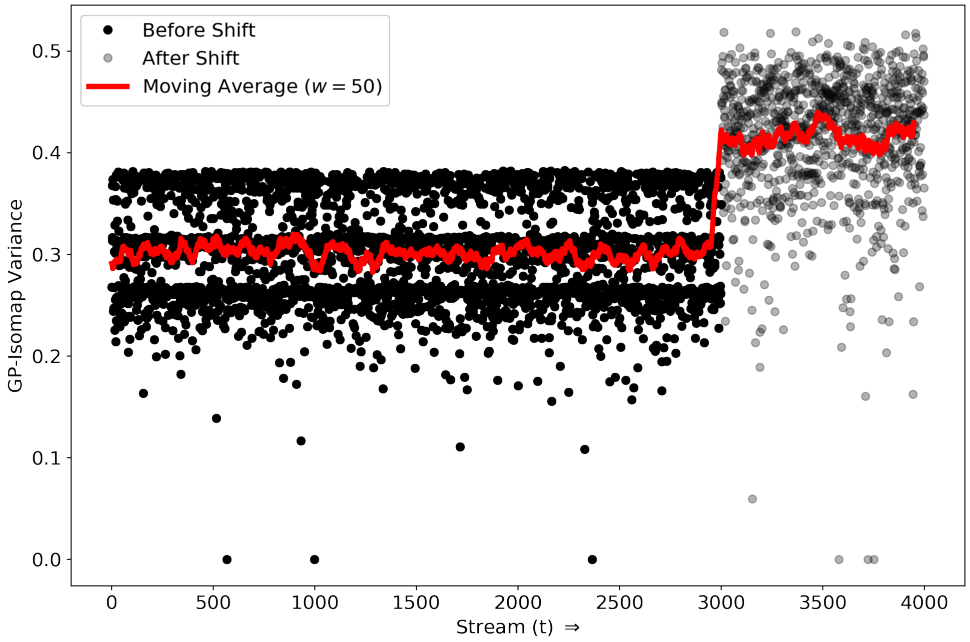


Figure 5.3: Using variance to detect *concept-drift* using the four patches dataset. The horizontal axis represents time and the vertical axis represents variance of the stream. Initially, when stream consists of samples generated from known modes, variance is low, later when samples from an unrecognized mode appear, variance shoots up. We can also observe the three variance "bands" above corresponding to the variance levels of the three modes for $\mathbf{t} \leq \mathbf{3000}$.

5.5.1.1 Gaussian patches on *Isometric Swiss Roll*

To evaluate our method on sudden *concept-drift*, we trained our GP-Isomap model using **3** out of **4** training sets of the *Isometric Swiss Roll* dataset. Subsequently we stream points randomly from the test sets of only these **3** classes initially and later stream points from the test set of the fourth class, keeping track of the predictive variance all the while. Fig. 5.3 demonstrates the sudden increase (see red line) in the variance of the stream when streaming points are from the fourth class i.e. unknown mode. Thus GP-Isomap is able to detect *concept-drift* correctly. The bottom panel of Fig. 5.1 demonstrates the performance of S-Isomap++ on this dataset. It fails to map the streaming points of the unknown mode correctly, given it had not encountered the unknown mode in

its batch training phase.

To test our proposed approach for detecting incremental *concept-drift*, we train our model using the single patch dataset and subsequently observe how the variance of the stream behaves on the test streaming dataset. The top panel of Fig. 5.1 shows how gradually variance increases smoothly as the stream gradually drifts away from the Gaussian patch. This shows that GP-Isomap maps incremental drift correctly. In Sect. 5.4, we proved the equivalence between the prediction of S-Isomap with that of GP-Isomap, using our novel kernel. In Fig. 5.4, we show empirically via Procrustes Error (PE) that indeed the prediction of S-Isomap matches that of GP-Isomap, irrespective of size of batch used. PE for GP-Isomap with the Euclidean distance based kernel remains high irrespective of the size of the batch, which clearly demonstrates the unsuitability of this kernel to adequately learn mappings in the low-dimensional space.

5.5.2 Results on Sensor Data Set

The *Gas Sensor Array Drift* (GSAD) [80] is a benchmark dataset ($\mathbf{n = 13910}$) available to research communities to develop strategies to dealing with *concept-drift* and uses measurements from **16** chemical sensors used to discriminate between **6** gases (class labels) at various concentrations. We demonstrate the performance of our proposed method on this dataset.

The data was first *mean normalized*. Data points from the first five classes were divided into training and test sets. We train our model using the training data from four out of these five classes. While testing, we stream points randomly from the test sets of these four classes first and later stream points from the test set of the fifth class. Figure 5.5 demonstrates our results. Our model can clearly detect *concept-drift* due to the unknown fifth class by tracking the variance of the stream, using the running average (red line).

5.6 Related Work

Processing data streams efficiently using standard approaches is challenging in general, given streams require real-time processing and cannot be stored per-

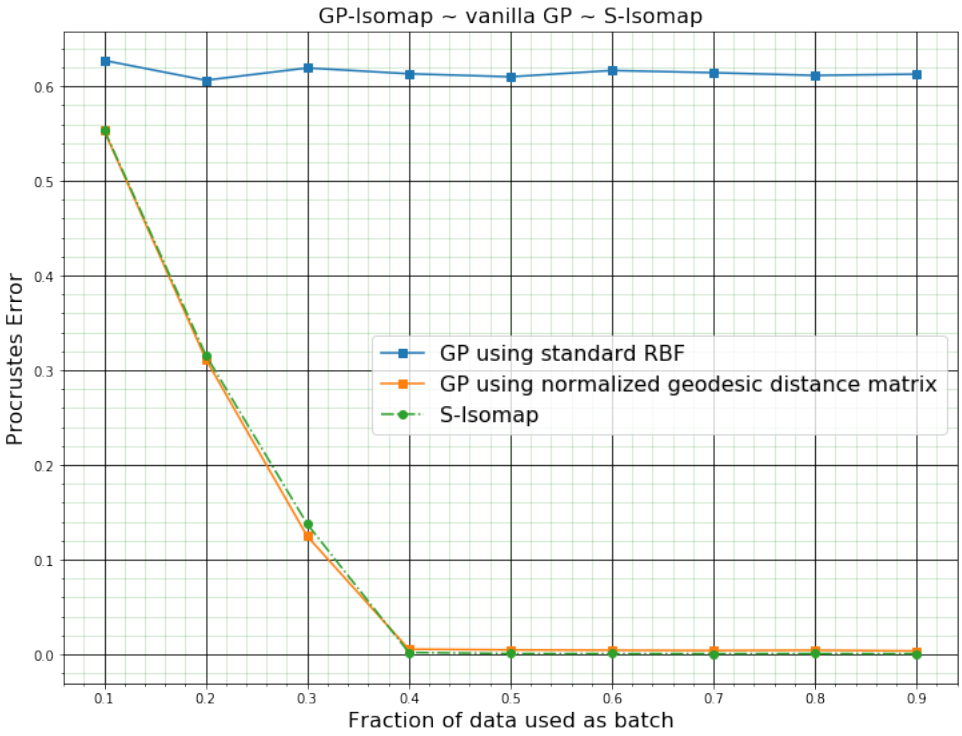


Figure 5.4: Procrustes error (PE) between the ground truth with a) GP-Isomap (orange line) with the geodesic distance based kernel, b) S-Isomap (dashed green line with dots) and c) GP-Isomap (blue line) using the Euclidean distance based kernel, for different fractions ($\mathbf{f}$) of data used in the batch $\boldsymbol{\mathcal{B}}$. The behavior of PE for a) closely matches that for b). However, the PE for GP-Isomap using the Euclidean distance kernel remains high irrespective of $\mathbf{f}$ demonstrating its unsuitability for manifolds.

manently. Any form of analysis, including detecting *concept-drift* requires adequate summarization which can deal with the inherent constraints and that can approximate the characteristics of the stream well. Sampling based strategies include *random sampling* [81, 9] as well as decision-tree based approaches [15] which have been used in this context. To identify *concept-drift*, maintaining statistical summaries on a streaming "window" is a typical strategy [1, 34, 14]. However, none of these are applicable in the setting of learning a latent representation from the data, e.g., manifolds, in the presence of changes in the stream distribution.

We discuss limitations of existing incremental and streaming solutions that have been specifically developed in the context of manifold learning, specifi-

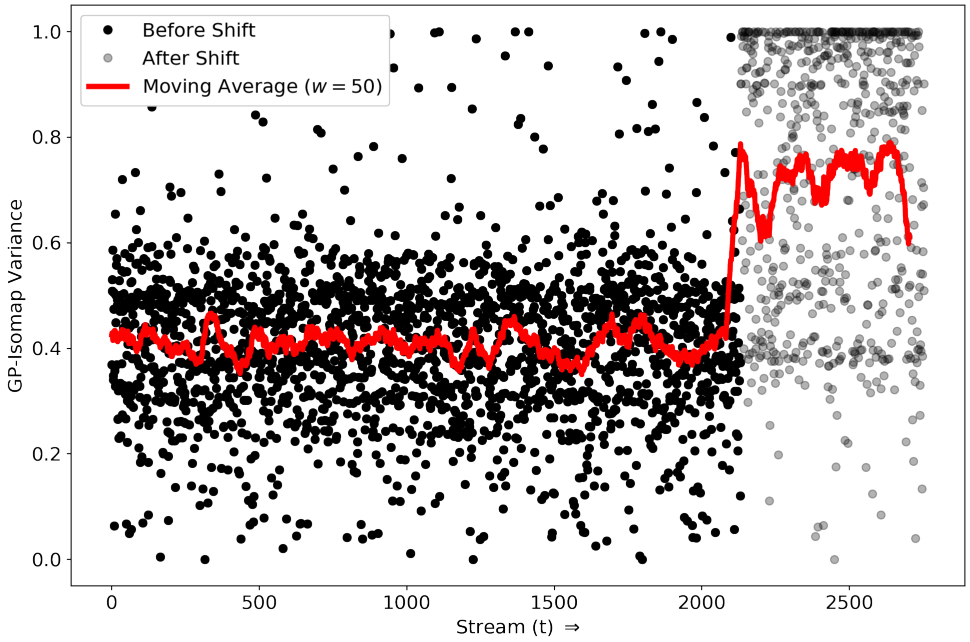


Figure 5.5: Using variance to identify *concept-drift* for the *Gas Sensor Array Drift* dataset. Similar to Fig. 5.3, the introduction of points from an unknown mode in the stream results in variance increasing drastically as demonstrated by the mean (red line). The spread of variances for points from known modes ($\mathbf{t} \precsim \mathbf{2000}$) is also smaller, compared to the spread for the points from the unknown mode ($\mathbf{t} \succsim \mathbf{2000}$).

cally in the context of the Isomap algorithm in Section 5.2. Coupling Isomap with GP Regression (GPR) has been explored in the past [12, 87], though not in the context of streaming data. For instance, a Mercer kernel-based Isomap technique has been proposed [12]. Similarly [87] presented an emulator pipeline using Isomap to determine a low-dimensional representation, whose output is fed to a GPR model. The intuition to use GPR for detecting *concept-drift* is novel even though the Bayesian non-parametric approach [3], primarily intended for anomaly detection, comes close to our work in a single manifold setting. Their choice of the Euclidean distance (in original $\mathbb{R}^D$ space) based kernel for its covariance matrix, can result in high Procrustes error, as shown in Fig. 5.4. Additionally, their approach does not scale, given it does not use any approximation

to be able to process the new streaming points "cheaply".

5.7 Conclusion

We have proposed a streaming Isomap algorithm (GP-Isomap) that can be used to learn nonlinear low-dimensional representation of high-dimensional data arriving in a streaming fashion. We prove that using a GPR formulation to map incoming data instances onto an existing manifold is equivalent to using existing geometric strategies [65, 50]. Moreover, by utilizing a small batch for exact learning of the Isomap as well as training the GPR model, the method scales linearly with the size of the stream, thereby ensuring its applicability for practical problems. Using the Bayesian inference of the GPR model allows us to estimate the variance associated with the mapping of the streaming instances. The variance is shown to be a strong indicator of changes in the underlying stream properties on a variety of data sets. By utilizing the variance, one can devise re-training strategies that can include expanding the batch data set. While we have focused on Isomap algorithm in this chapter, similar formulations can be applied for other NLSDR methods such as LLE [60], etc., and will be explored as future research.

5.8 Theoretical results related to GP-Isomap

Lemma 5.8.1. *The matrix exponential for* $\mathbf{M}$ *for rank*$(\mathbf{M}) = 1$ *and symmetric* $\mathbf{M}$ *is given by*

$$e^{\mathbf{M}} = \mathbf{I} + \left(e^{\boldsymbol{\lambda}_1} - 1\right)\mathbf{q}_1\mathbf{q}_1^T$$

where $\mathbf{q}_1$ *is the first eigenvector of* $\mathbf{M}$ *such that* $\mathbf{q}_1^T\mathbf{q}_1 = 1$ *and* $\boldsymbol{\lambda}_1$ *is the corresponding eigenvalue.*

Proof. Given $\mathbf{M}$ is symmetric and rank one, $\mathbf{M}$ can be written as $\boldsymbol{\lambda}_1 \mathbf{q}_1 \mathbf{q}_1^T$.

$$\begin{aligned} e^{\mathbf{M}} &= \mathbf{I} + \sum_{k=1}^{\infty} \frac{1}{\mathbf{k}\,!} \mathbf{M}^k \\ &= \mathbf{I} + \frac{\boldsymbol{\lambda}_1}{1\,!} \mathbf{q}_1 \mathbf{q}_1^T + \frac{\boldsymbol{\lambda}_1^2}{2\,!} \mathbf{q}_1 \mathbf{q}_1^T \mathbf{q}_1 \mathbf{q}_1^T + \cdots \\ &= \mathbf{I} + \big(\frac{\boldsymbol{\lambda}_1}{1\,!} + \frac{\boldsymbol{\lambda}_1^2}{2\,!} + \dots\big) \mathbf{q}_1 \mathbf{q}_1^T \\ &= \mathbf{I} + \big(e^{\boldsymbol{\lambda}_1} - 1\big) \mathbf{q}_1 \mathbf{q}_1^T \end{aligned} \tag{5.19}$$

■

Lemma 5.8.2. *The matrix exponential for* $\mathbf{M}$ *for rank*$\big(\mathbf{M}\big) = d$ *and symmetric* $\mathbf{M}$ *is given by*

$$e^{\mathbf{M}} = \mathbf{I} + \sum_{i=1}^{d} \big(e^{\boldsymbol{\lambda}_i} - 1\big) \mathbf{q}_i \mathbf{q}_i^{\top}$$

where $\{\boldsymbol{\lambda}_i\}_{i=1,2\dots d}$ *are the* d *largest eigenvalues of* $\mathbf{M}$ *and* $\{\mathbf{q}_i\}_{i=1,2\dots d}$ *are the corresponding eigenvectors such that* $\mathbf{q}_i^{\top} \mathbf{q}_j = \boldsymbol{\delta}_{i,j}$.

Proof. Let $\mathbf{M}$ be an $n \times n$ real matrix. The exponential $e^{\mathbf{M}}$ is given by

$$e^{\mathbf{M}} = \sum_{k=0}^{\infty} \frac{1}{\mathbf{k}\,!} \mathbf{M}^{\mathbf{k}} = \mathbf{I} + \sum_{k=1}^{\infty} \frac{1}{\mathbf{k}\,!} \mathbf{M}^{\mathbf{k}}$$

where $\mathbf{I}$ is the identity. Real, symmetric $\mathbf{M}$ has real eigenvalues and mutually orthogonal eigenvectors i.e. $\mathbf{M} = \sum_{i=1}^{n} \boldsymbol{\lambda}_i \mathbf{q}_i \mathbf{q}_i^{\top}$ where $\{\boldsymbol{\lambda}_i\}_{i=1\dots n}$ are real and $\mathbf{q}_i^{\top} \mathbf{q}_j = \boldsymbol{\delta}_{i,j}$.

Given $\mathbf{M}$ has rank d, we have $\mathbf{M} = \sum_{i=1}^{d} \boldsymbol{\lambda}_i \mathbf{q}_i \mathbf{q}_i^\top$.

$$
\begin{aligned}
e^{\mathbf{M}} &= \mathbf{I} + \sum_{i=1}^{\infty} \frac{1}{\mathbf{i}\,!} \mathbf{M}^i \\
&= \mathbf{I} + \frac{1}{1\,!}(\boldsymbol{\lambda}_1 \mathbf{q}_1 \mathbf{q}_1^\top + \boldsymbol{\lambda}_2 \mathbf{q}_2 \mathbf{q}_2^\top + \ldots + \boldsymbol{\lambda}_d \mathbf{q}_d \mathbf{q}_d^\top) \\
&+ \frac{1}{2\,!}(\boldsymbol{\lambda}_1 \mathbf{q}_1 \mathbf{q}_1^\top + \boldsymbol{\lambda}_2 \mathbf{q}_2 \mathbf{q}_2^\top + \ldots + \boldsymbol{\lambda}_d \mathbf{q}_d \mathbf{q}_d^\top)^2 + \ldots \\
&= \mathbf{I} + (\frac{\boldsymbol{\lambda}_1}{1\,!} + \frac{\boldsymbol{\lambda}_1^2}{2\,!} + \ldots)\mathbf{q}_1 \mathbf{q}_1^\top + (\frac{\boldsymbol{\lambda}_2}{1\,!} + \frac{\boldsymbol{\lambda}_2^2}{2\,!} + \ldots)\mathbf{q}_2 \mathbf{q}_2^\top + \ldots \\
&+ (\frac{\boldsymbol{\lambda}_d}{1\,!} + \frac{\boldsymbol{\lambda}_d^2}{2\,!} + \ldots)\mathbf{q}_d \mathbf{q}_d^\top \\
&= \mathbf{I} + \left(e^{\boldsymbol{\lambda}_1} - 1\right)\mathbf{q}_1 \mathbf{q}_1^\top + \left(e^{\boldsymbol{\lambda}_2} - 1\right)\mathbf{q}_2 \mathbf{q}_2^\top + \ldots + \left(e^{\boldsymbol{\lambda}_d} - 1\right)\mathbf{q}_d \mathbf{q}_d^\top \\
&= \mathbf{I} + \sum_{i=1}^{d} \left(e^{\boldsymbol{\lambda}_i} - 1\right)\mathbf{q}_i \mathbf{q}_i^\top
\end{aligned}
\tag{5.20}
$$

■

Lemma 5.8.3. *The inverse of the Gaussian kernel for rank*$(\mathbf{M}) = 1$ *and symmetric* $\mathbf{M}$ *is given by*

$$
\left(\mathbf{K} + {\boldsymbol{\sigma}_n}^2 \mathbf{I}\right)^{-1} = \boldsymbol{\alpha}\mathbf{I} - \frac{\boldsymbol{\alpha}^2 \mathbf{c_1} \mathbf{q_1} \mathbf{q_1}^\top}{1 + \boldsymbol{\alpha}\mathbf{c_1}}
$$

where $\mathbf{q}_1$ *is the first eigenvector of* $\mathbf{M}$ *i.e.* $\mathbf{q}_1^\top \mathbf{q}_1 = 1$, $\boldsymbol{\lambda}_1$ *is the corresponding eigenvalue and* $\boldsymbol{\alpha} = \frac{1}{\left(1 + {\boldsymbol{\sigma}_n}^2\right)}$ *and* $\mathbf{c}_1 = \left[\exp\left(-\frac{\boldsymbol{\lambda}_1}{2\ell^2}\right) - 1\right]$.

Proof. Using (5.11) for $\mathbf{d} = 1$, we have

$$
\begin{aligned}
\left(\mathbf{K} + {\boldsymbol{\sigma}_n}^2 \mathbf{I}\right)^{-1} &= \left(\mathbf{I} + \left[\exp\left(-\frac{\boldsymbol{\lambda}_1}{2\ell^2}\right) - 1\right]\mathbf{q}_1 \mathbf{q}_1^\top + {\boldsymbol{\sigma}_n}^2 \mathbf{I}\right)^{-1} \\
&= \left(\left(1 + {\boldsymbol{\sigma}_n}^2\right)\mathbf{I} + \left[\exp\left(-\frac{\boldsymbol{\lambda}_1}{2\ell^2}\right) - 1\right]\mathbf{q}_1 \mathbf{q}_1^\top\right)^{-1}
\end{aligned}
\tag{5.21}
$$

Representing $\frac{1}{\left(1 + {\boldsymbol{\sigma}_n}^2\right)}$ as $\boldsymbol{\alpha}$ and $\left[\exp\left(-\frac{\boldsymbol{\lambda}_1}{2\ell^2}\right) - 1\right]$ as $\mathbf{c}_1$ and using $\left(1 + {\boldsymbol{\sigma}_n}^2\right)\mathbf{I}$ as

$\mathbf{A}$, $\mathbf{c}_1\mathbf{q}_1$ as $\mathbf{u}$ and $\mathbf{q}_1$ as $\mathbf{v}$ in the Sherman-Morrison identity [57], we have

$$\begin{aligned}\left(\mathbf{K}+{\boldsymbol{\sigma}_n}^2\mathbf{I}\right)^{-1} &= \boldsymbol{\alpha}\mathbf{I}-\frac{\boldsymbol{\alpha}\mathbf{I}\mathbf{c}_1\mathbf{q}_1\mathbf{q}_1^\top\boldsymbol{\alpha}\mathbf{I}}{1+\boldsymbol{\alpha}\mathbf{c}_1}\\ &= \boldsymbol{\alpha}\mathbf{I}-\frac{\boldsymbol{\alpha}^2\mathbf{c}_1\mathbf{q}_1\mathbf{q}_1^\top}{1+\boldsymbol{\alpha}\mathbf{c}_1}\end{aligned} \tag{5.22}$$

■

Lemma 5.8.4. *The inverse of the Gaussian kernel for rank*$(\mathbf{M})=\mathbf{d}$ *and symmetric* $\mathbf{M}$ *is given by*

$$\left(\mathbf{K}+{\boldsymbol{\sigma}_n}^2\mathbf{I}\right)^{-1}=\boldsymbol{\alpha}\mathbf{I}-\boldsymbol{\alpha}^2\sum_{i=1}^{d}\frac{\mathbf{c}_i\mathbf{q}_i\mathbf{q}_i^\top}{1+\boldsymbol{\alpha}\mathbf{c}_i}$$

where $\{\boldsymbol{\lambda}_i\}_{i=1,2\ldots d}$ *are the* $\mathbf{d}$ *largest eigenvalues of* $\mathbf{M}$ *and* $\{\mathbf{q}_i\}_{i=1,2\ldots d}$ *are the corresponding eigenvectors such that* $\mathbf{q}_i^\top\mathbf{q}_j=\boldsymbol{\delta}_{i,j}$.

Proof. Using the result of previous lemma iteratively, we get the required result

$$\left(\mathbf{K}+{\boldsymbol{\sigma}_n}^2\mathbf{I}\right)^{-1}=\boldsymbol{\alpha}\mathbf{I}-\boldsymbol{\alpha}^2\sum_{i=1}^{d}\frac{\mathbf{c}_i\mathbf{q}_i\mathbf{q}_i^\top}{1+\boldsymbol{\alpha}\mathbf{c}_i} \tag{5.23}$$

where $\boldsymbol{\alpha}=\frac{1}{\left(1+{\boldsymbol{\sigma}_n}^2\right)}$ and $\mathbf{c}_i=\left[\exp\left(-\frac{\boldsymbol{\lambda}_i}{2\boldsymbol{\ell}^2}\right)-1\right]$. ■

Lemma 5.8.5. *The solution for Gaussian Process regression system, for the scenario when rank*$(\mathbf{M})=1$ *and for symmetric* $\mathbf{M}$ *is given by*

$$\left(\mathbf{K}+{\boldsymbol{\sigma}_n}^2\mathbf{I}\right)^{-1}\mathbf{y}=\frac{\boldsymbol{\alpha}\sqrt{\boldsymbol{\lambda}}_1\mathbf{q}_1}{1+\boldsymbol{\alpha}\mathbf{c_1}}$$

Proof. Assuming the intrinsic dimensionality of the low-dimensional manifold to be 1 implies that the inverse of the Gaussian kernel is as defined as in (5.23).

$\mathbf{y}$ is $\sqrt{\boldsymbol{\lambda}_1}\mathbf{q}_1$ in this case (refer Sect. 5.2.1). Thus we have

$$\begin{aligned}\left(\mathbf{K}+{\boldsymbol{\sigma}_n}^2\mathbf{I}\right)^{-1}\mathbf{y} &= \left(\boldsymbol{\alpha}\mathbf{I}-\frac{\boldsymbol{\alpha}^2\mathbf{c}_1\mathbf{q}_1\mathbf{q}_1^\top}{1+\boldsymbol{\alpha}\mathbf{c}_1}\right)\left(\sqrt{\boldsymbol{\lambda}_1}\mathbf{q}_1\right)\\ &= \boldsymbol{\alpha}\sqrt{\boldsymbol{\lambda}_1}\mathbf{q}_1-\frac{\boldsymbol{\alpha}^2\sqrt{\boldsymbol{\lambda}_1}\mathbf{c}_1\mathbf{q}_1}{1+\boldsymbol{\alpha}\mathbf{c}_1}\\ &= \frac{\boldsymbol{\alpha}\sqrt{\boldsymbol{\lambda}_1}\mathbf{q}_1}{1+\boldsymbol{\alpha}\mathbf{c}_1}\end{aligned} \tag{5.24}$$

■

Lemma 5.8.6. *The solution for Gaussian Process regression system, for the scenario when rank*$\left(\mathbf{M}\right)=\mathbf{d}$ *and for symmetric* $\mathbf{M}$ *is given by*

$$\left(\mathbf{K}+{\boldsymbol{\sigma}_n}^2\mathbf{I}\right)^{-1}\mathbf{y}=\{\frac{\boldsymbol{\alpha}\sqrt{\boldsymbol{\lambda}_1}\mathbf{q}_1}{1+\boldsymbol{\alpha}\mathbf{c}_1}\ \frac{\boldsymbol{\alpha}\sqrt{\boldsymbol{\lambda}_2}\mathbf{q}_2}{1+\boldsymbol{\alpha}\mathbf{c}_2}\ \ldots\ \frac{\boldsymbol{\alpha}\sqrt{\boldsymbol{\lambda}_d}\mathbf{q}_d}{1+\boldsymbol{\alpha}\mathbf{c}_d}\}$$

Proof. Assuming the intrinsic dimensionality of the low-dimensional manifold to be $\mathbf{d}$ implies that the inverse of the Gaussian kernel is as defined as in (5.24). $\mathbf{y}$ is $\{\sqrt{\boldsymbol{\lambda}_1}\mathbf{q}_1\ \sqrt{\boldsymbol{\lambda}_2}\mathbf{q}_2\ \ldots\ \sqrt{\boldsymbol{\lambda}_d}\mathbf{q}_d\}$ in this case (refer Sect. 5.2.1), where $\mathbf{q}_i^\top\mathbf{q}_j=\boldsymbol{\delta}_{i,j}$.

Each of the $\mathbf{k}$ dimensions of $\left(\mathbf{K}+{\boldsymbol{\sigma}_n}^2\mathbf{I}\right)^{-1}\mathbf{y}$ can be processed independently, similar to the previous lemma. For the $\mathbf{i}^{\text{th}}$ dimension, we have,

$$\begin{aligned}\left(\mathbf{K}+{\boldsymbol{\sigma}_n}^2\mathbf{I}\right)^{-1}\mathbf{y_i} &= \left(\boldsymbol{\alpha}\mathbf{I}-\boldsymbol{\alpha}^2\sum_{j=1}^{d}\frac{\mathbf{c}_j\mathbf{q}_j\mathbf{q}_j^\top}{1+\boldsymbol{\alpha}\mathbf{c}_j}\right)\left(\sqrt{\boldsymbol{\lambda}_i}\mathbf{q}_i\right)\\ &= \boldsymbol{\alpha}\sqrt{\boldsymbol{\lambda}_i}\mathbf{q}_i-\boldsymbol{\alpha}^2\sum_{j=1}^{d}\frac{\mathbf{c}_j\mathbf{q}_j\mathbf{q}_j^\top\mathbf{q}_i\left(\sqrt{\boldsymbol{\lambda}_i}\right)}{1+\boldsymbol{\alpha}\mathbf{c}_j}\\ &= \boldsymbol{\alpha}\sqrt{\boldsymbol{\lambda}_i}\mathbf{q}_i-\frac{\boldsymbol{\alpha}^2\sqrt{\boldsymbol{\lambda}_i}\mathbf{c}_i\mathbf{q}_i}{1+\boldsymbol{\alpha}\mathbf{c}_i}\\ &= \frac{\boldsymbol{\alpha}\sqrt{\boldsymbol{\lambda}_i}\mathbf{q}_i}{1+\boldsymbol{\alpha}\mathbf{c}_i}\end{aligned} \tag{5.25}$$

Thus we get the result,

$$\left(\mathbf{K}+{\boldsymbol{\sigma}_n}^2\mathbf{I}\right)^{-1}\mathbf{y}=\{\frac{\boldsymbol{\alpha}\sqrt{\boldsymbol{\lambda}_1}\mathbf{q}_1}{1+\boldsymbol{\alpha}\mathbf{c}_1}\ \frac{\boldsymbol{\alpha}\sqrt{\boldsymbol{\lambda}_2}\mathbf{q}_2}{1+\boldsymbol{\alpha}\mathbf{c}_2}\ \ldots\ \frac{\boldsymbol{\alpha}\sqrt{\boldsymbol{\lambda}_d}\mathbf{q}_d}{1+\boldsymbol{\alpha}\mathbf{c}_d}\} \tag{5.26}$$

■

Chapter 6

A generalized Out-of-Sample Extension (OOSE) framework for streaming NLSDR

In this chapter, we discuss a generalized Out-of-Sample Extension framework for streaming NLSDR discussed which can be formulated with different manifold learning algorithms as necessary, and which has the ability to effectively deal with multiple manifolds. Additionally we discuss a novel manifold learning algorithm, based on the above framework, apart from that described in Chapter 4. The rest of the chapter is organized as follows: we describe and discuss the generalized Out-of-Sample Extension framework in Section 6.1 and discuss related work in Section 6.2. We describe Streaming-LLE, the proposed algorithm based on LLE, which is a specific instance of the above framework, in Section 6.3. We conclude the chapter by demonstrating results for Streaming-LLE for both synthetic as well as benchmark datasets in Section 6.4.

6.1 A Standard generalized OOSE framework

The skeleton of the S-Isomap++ algorithm (see Algorithm 2 in Section 4.4) motivates a generalized Out-of-Sample Extension framework for streaming NLSDR for multiple manifolds wherein the main framework remains fixed and the dif-

ferent constituents which are specific to the manifold learning algorithm can be plugged in as is appropriate. This generalized framework is outlined in Algorithm 7 below. The algorithm can be said to be parametrized by $\mathcal{A}$, the NLSDR technique used in the batch phase as well as by $\mathcal{OOS}_{\mathcal{A}}$, the possibly "cheap" Out-of-Sample Extension technique specific to the NLSDR technique $\mathcal{A}$. The S-Isomap++ algorithm can be thought of as a specific instantiation of Algorithm 7 wherein $\mathcal{A}$ is Isomap and $\mathcal{OOS}_{\mathcal{A}}$ is S-Isomap. In the subsequent section 6.3, we define a instantiation specific to LLE as well. All methods referred to in the Algorithm 7 below can be found in Section 4.4 since they are constituents of the main framework which work similarly for different NLSDR techniques that can be plugged into the below framework as needed.

The proposed generalized OOSE framework follows a two-phase strategy similar to that proposed in our earlier work [50], where we first learn exact manifolds from an initial batch, and subsequently employ a computationally inexpensive mapping procedure to process the rest of the stream. An error metric is used to decide on when to *switch* from expensive and exact learning to inexpensive and approximate mapping [65]. We first cluster the batch data and then apply the NLSDR technique $\mathcal{A}$ on each cluster. The resulting low-dimensional data for the clusters is then *stitched* together to obtain the data reduced to a low (and closer to true) dimensionality.

The generalized approach takes a batch data set, $\mathcal{B}$ and the streaming data, $\mathcal{S}$ as inputs where $\mathcal{B}, \mathcal{S} \in \mathbb{R}^{\mathbf{D}}$. Note that in practical applications, one might not have data split into batch and streaming parts. In that scenario, one may track the quality of the output of the batch phase using suitable error metrics [65], and switch when a reliable solution for the batch is obtained. For simplicity, we will assume that the optimal batch size has been pre-determined. The processing is split into two phases: a batch learning phase (Lines 1–12) and a streaming phase (Lines 13–20). The batch phase can be summarized as :

- Step 1: Cluster samples in $\mathcal{B}$ into $\mathbf{p}$ clusters (Line 1).
- Step 2: Learn $\mathbf{p}$ individual manifolds corresponding to each cluster using $\mathcal{A}$, the specific manifold learning algorithm and map samples within each cluster to a low-dimensional representation. (Lines 6–7).

- Step 3: Map reduced samples from individual manifolds into a global reduced space (Lines 8–12).

Clustering separates the batch samples into different individual clusters, such that each cluster corresponds to one of the multiple manifolds present in the data. Our intuition for clustering and subsequently processing each of the clusters separately is based on our thinking that the observed data was generated by first sampling points from multiple $\boldsymbol{\mathcal{V}}_{i=1,2,\ldots p}$ i.e. convex domains in $\mathbb{R}^{\mathbf{d}}$ Euclidean space[1] and subsequently mapping those points nonlinearly using possibly different $\phi_{i=1,2,\ldots p}$ to a $\mathbb{R}^{\mathbf{D}}$ space. Thus to be able to learn the different inverse mappings effectively i.e. the different $\phi^{-1}_{i=1,2,\ldots p}$ which manifold learning algorithms strive to achieve, we need to be able to cluster the data appropriately. Also note that we do not assume that the number of manifolds ($\mathbf{p}$) is specified; it is automatically inferred by the clustering framework. In cases of uneven/low density sampling, the clustering strategy discussed might possibly generate many small clusters. In such cases, one can try to merge clusters, based on their affinity/closeness to allow the number of clusters to remain within required limits. Given that the batch samples lie on low-dimensional and potentially intersecting manifolds, it is evident that the standard clustering methods, such as K-Means [35], that operate on the observed data in $\mathbb{R}^{\mathbf{D}}$, will fail in correctly identifying the clusters.

The streaming phase, maps each sample in the stream set $\boldsymbol{\mathcal{S}}$ onto each of the $\mathbf{p}$ manifolds by using $\mathcal{OOS}_{\mathcal{A}}$ (Lines 14-17). The nearest manifold is identified by comparing each reduced representation of the sample to the "center" of each manifold (Line 18), and choosing the corresponding reduced representation for the stream sample (Line 19).

6.2 Related Work

[60] proposed the original LLE algorithm for NLSDR and subsequently [64] in

[1] It is possible that the low-dimensional Euclidean space specific to each manifold is different i.e. $\mathcal{V}_i$ is a convex domain in $\mathbb{R}^{\mathbf{d}_i}$ space, where $\mathbf{d}_i \neq \mathbf{d}_j$. However we can imagine a scenario where we choose a $\mathbb{R}^{\mathbf{d}}$ global space, where $\mathbf{d} = \max_i \mathbf{d}_i$ from which the different convex $\boldsymbol{\mathcal{V}}_i$ were sampled from. Additionally note that convexity is preserved by linear projections to higher dimensional spaces thus the convex domains $\boldsymbol{\mathcal{V}}_{i=1,2,\ldots p}$ remain convex in this new space.

Algorithm 7 A generalized Out-of-Sample Extension framework for streaming NLSDR for multiple manifolds.

Require: Batch dataset: $\boldsymbol{\mathcal{B}}$, Streaming dataset: $\boldsymbol{\mathcal{S}}$; Parameters: ϵ, $\mathbf{k}$, $\mathbf{l}$, $\boldsymbol{\lambda}$
Ensure: $\boldsymbol{\mathcal{Y}}_{\mathcal{S}}$: low-dimensional representation for $\boldsymbol{\mathcal{S}}$

▷ Batch Phase

$\boldsymbol{\mathcal{C}}_{i=1,2\ldots\mathbf{p}} \leftarrow$ FIND_CLUSTERS($\boldsymbol{\mathcal{B}}$, ϵ)
$\boldsymbol{\xi}_s \leftarrow \emptyset$

for $1 \leq \mathbf{i} \leq \mathbf{p}$ **do**
    $\boldsymbol{\mathcal{LDE}}_i \leftarrow \boldsymbol{\mathcal{A}}(\boldsymbol{\mathcal{C}}_i)$
end for

$\boldsymbol{\xi}_s \leftarrow \bigcup_{i=1}^{p} \bigcup_{j=i+1}^{p} \mathrm{NN}(\boldsymbol{\mathcal{C}}_i, \boldsymbol{\mathcal{C}}_j, \mathbf{k}) \cup \mathrm{FN}(\boldsymbol{\mathcal{C}}_i, \boldsymbol{\mathcal{C}}_j, \mathbf{l})$
$\boldsymbol{\mathcal{GE}}_s \leftarrow \mathrm{MDS}(\boldsymbol{\xi}_s)$

for $1 \leq \mathbf{j} \leq \mathbf{p}$ **do**
    $\boldsymbol{\mathcal{I}} \leftarrow \boldsymbol{\xi}_s \cap \boldsymbol{\mathcal{C}}_j$
    $\boldsymbol{\mathcal{A}} \leftarrow \begin{bmatrix} \boldsymbol{\mathcal{LDE}}_j^{\mathcal{I}} \\ \mathbf{e}^T \end{bmatrix}$
    $\boldsymbol{\mathcal{R}}_i, \mathbf{t}_i \leftarrow \boldsymbol{\mathcal{GE}}_{\mathcal{I},s} \times \boldsymbol{\mathcal{A}}^T \left(\boldsymbol{\mathcal{A}}\boldsymbol{\mathcal{A}}^T + \boldsymbol{\lambda}\mathbf{I}\right)^{-1}$
end for

▷ Streaming Phase

for $\mathbf{s} \in \boldsymbol{\mathcal{S}}$ **do**
    for $1 \leq \mathbf{i} \leq \mathbf{p}$ **do**
        $\mathbf{y}_s^i \leftarrow \mathcal{OOS}_{\mathcal{A}}(\mathbf{s}, \boldsymbol{\mathcal{C}}_i)$
        $\boldsymbol{\mathcal{GE}}_s^i \leftarrow \boldsymbol{\mathcal{R}}_i \mathbf{y}_s^i + \mathbf{t}_i$
    end for

    $\mathbf{m} \leftarrow \mathrm{argmin}_i \left| \mathbf{y}_s^i - \mu(\boldsymbol{\mathcal{C}}_i, \boldsymbol{\mathcal{R}}_i, \mathbf{t}_i) \right|$
    $\boldsymbol{\mathcal{Y}}_{\mathcal{S}} \leftarrow \boldsymbol{\mathcal{Y}}_{\mathcal{S}} \cup \mathbf{y}_s^m$
end for

return $\boldsymbol{\mathcal{Y}}_{\mathcal{S}}$

their work, proposed two approaches for Out-of-Sample Extensions for LLE. The first was a simple approach of first determining the local neighborhood for the new sample and subsequently computing a set of locally linear weights for the new sample with respect to these set of points in the local neighborhood by solving a simple least-squares problem. The low-dimensional embedding of the new sample subsequently gets computed as the linear combination of the low-dimensional embeddings of the set of points in the local neighborhood using the weights determined previously. The other approach was learning the function which maps from the high-dimensional $\mathbb{R}^{\mathbf{D}}$ Euclidean space of the samples to the low-dimensional $\mathbb{R}^{\mathbf{d}}$ manifold space by conditioning on already encountered data and their corresponding low-dimensional representations. These approaches were applicable in general for single manifold scenarios. [27] proposed a technique for dealing with multiple manifolds, however the technique only works for supervised data for which labels for each training sample was known beforehand. Additionally, the technique was meant for batch processing and not for Out-of-Sample Extensions. [71, 90] have also proposed Out-of-Sample Extension approaches for LLE, however again like [64] they are applicable for single manifold scenarios only and not for multiple, possibly intersecting manifolds arising due to possibly non-uniform sampling and/or from multi-modal distributions.

6.3 Methodology

We describe the Streaming-LLE algorithm in this section, which is an specific instantiation of the generalized methodology discussed in the previous section.

6.3.1 Streaming-LLE or S-LLE

The Streaming-LLE or S-LLE algorithm is described in Algorithm 8. Given it is a specific instantiation of the generalized Algorithm 7, described in Section 6.1, we describe below only the sections of the algorithm which are specific to LLE here. We describe the Out-of-Sample Extension strategy for LLE in Algorithm 9 subsequently.

The sections specific to LLE in the Streaming LLE algorithm consists of :

6.3.1.1 Dimensionality Reduction

The LLE algorithm is executed on each of the $\mathbf{p}$ individual clusters to get low-dimensional representations $\boldsymbol{\mathcal{LDE}}_{i=1,2...p}$ of the data points belonging to each individual cluster. (Lines 3–5)

6.3.1.2 Out-of-Sample Extension methodology for LLE

The Out-of-Sample Extension methodology described in Algorithm 9 is called for each streaming sample $\mathbf{s} \in \boldsymbol{\mathcal{S}}$ for each of the $\mathbf{p}$ individual manifolds to determine $\mathbf{y}_s^i$, the predicted local embedding for $\mathbf{s}$ for the $\mathbf{i}^{\text{th}}$ manifold. After which, the rotation and translation factors $\boldsymbol{\mathcal{R}}_i$ and $\mathbf{t}_i$ learnt in the batch learning phase for the $\mathbf{i}^{\text{th}}$ manifold are used to map the learnt local embedding into the global space, where all manifolds reside. (Lines 14–17).

The Out-of-Sample Extension strategy for LLE (refer Algorithm 9), is described below :

6.3.1.3 Nearest neighbor selection

The $\mathbf{k}$-nearest neighbors $\boldsymbol{\zeta}_s$ of the streaming sample $\mathbf{s}$ in the $\mathbf{i}^{\text{th}}$ manifold, $\boldsymbol{\mathcal{C}}_i$ is determined in this step, based on the underlying $\mathbb{R}^{\mathbf{D}}$ Euclidean space. (Line 2)

6.3.1.4 Learn optimal locally linear weights

$\mathbf{w}^*$, the optimal locally linear weights are determined which minimizes the objective $\left\| \left(\mathbf{s} - \sum_{x_j \in \zeta_s} \mathbf{w}_j \mathbf{x}_j \right) \right\|^2$ best as possible is learnt i.e. the best linear combination of the $\mathbf{k}$-nearest neighbors $\boldsymbol{\zeta}_s$ which can explain the streaming sample $\mathbf{s}$. Subsequently, this optimal linear combination $\left(\sum_{x_j \in \zeta_s} \mathbf{w}_j^* \mathbf{y}_j \right)$ is returned. (Line 3–4).

6.4 Results

In this section, we present results related to the Streaming-LLE algorithm on synthetically generated datasets as well as on benchmark datasets. We use the *Euler Isometric Swiss Roll* dataset [65], a synthetically generated dataset both to evaluate our algorithm as well as understand the effect of the various parameters on its working. Additionally we use *MNIST* and the *Gas Sensor Array Drift* (GSAD) dataset [80] as benchmark datasets to evaluate our algorithm.

6.4.1 Results on the *Euler Isometric Swiss Roll* dataset

The *Euler Isometric Swiss Roll* dataset [65] is a synthetically generated dataset consisting of four $\mathbb{R}^{\mathbf{2}}$ Gaussian patches having $\mathbf{n} = \mathbf{2000}$ points from each patch, chosen at random, which are embedded into $\mathbb{R}^{\mathbf{3}}$ using a nonlinear function $\boldsymbol{\psi}(\cdot)$. The points for each of the Gaussian modes are divided equally into training and test sets. Figure 6.4 demonstrates the ground truth for the streaming samples present in the dataset. Figure 6.5 demonstrates results for the low-dimensional embeddings uncovered by Streaming-LLE algorithm on this dataset. Comparing with the ground-truth to see how well Streaming-LLE algorithm is able to recreate it, we can observe that the recreation by Streaming-LLE is pretty good. We can also compare Figure 6.5, the recreation result by Streaming-LLE with Figure 4.1e, the recreation result by the S-Isomap++ algorithm, which is another instantiation of the same generalized framework as Streaming-LLE, on the same dataset. The quality of the results for the S-Isomap++ algorithm seems to be better, compared to Streaming-LLE algorithm, which can be explained by the strong linearity assumption for the underlying NLSDR methodology i.e. LLE for the Streaming-LLE algorithm, which seems to hamper the quality of the recreation marginally.

6.4.1.1 Effect of different parameters

Here we present results of the effect of changing the different parameters of the Streaming-LLE algorithm, *while keeping all other parameters fixed at optimal values*. Figures 6.1, 6.2, 6.3, demonstrates the effect of parameter $\boldsymbol{\lambda}$, $\mathbf{k}$ and $\mathbf{l}$

on the embeddings uncovered by the Streaming-LLE algorithm. While smaller values of **k** seems to result in skinny manifolds, larger values of **k** seems to make the manifolds more uniform or rounded. Parameter **l** seems to control how stretched the manifolds are i.e. larger values of **l** seem to stretch the manifolds. Parameter $\boldsymbol{\lambda}$ seem to control the separability in between the manifolds. Larger values of $\boldsymbol{\lambda}$ seem to separate the manifolds more. This can be really useful while visualization of low-dimensional embedding results for Streaming-LLE i.e. we can use it to visualize manifolds better on account of their separability.

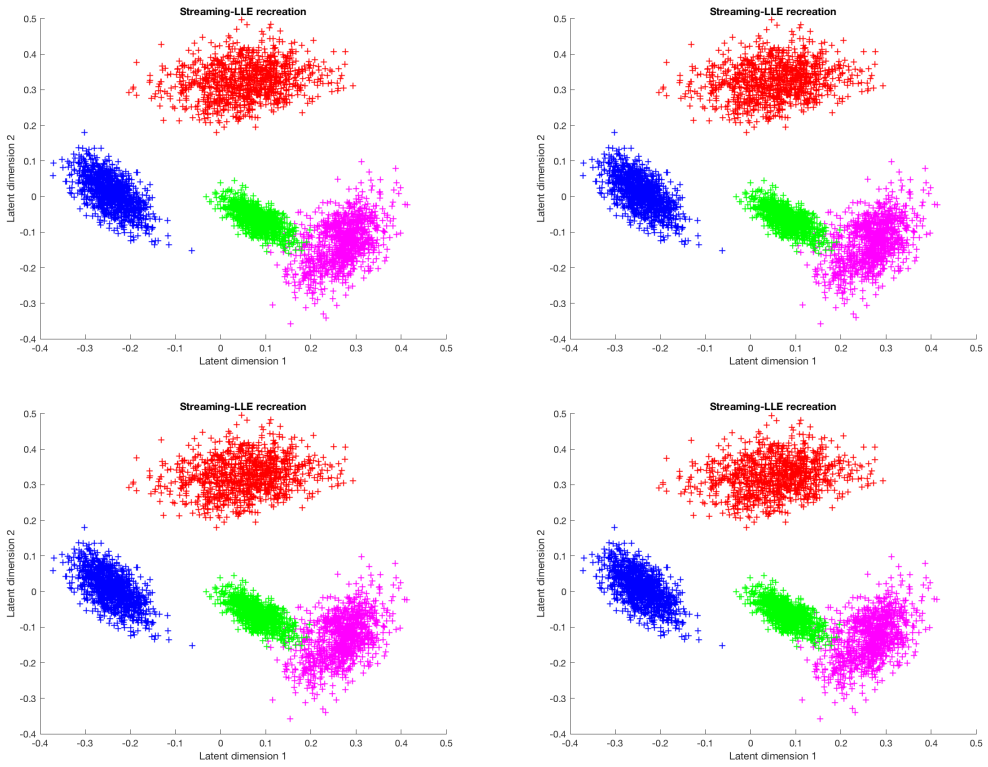


Figure 6.1: Effect of changing $\boldsymbol{\lambda}$ on Streaming-LLE. Top Left: $\boldsymbol{\lambda} = \mathbf{0.005}$, Top Right: $\boldsymbol{\lambda} = \mathbf{0.01}$, Bottom Left: $\boldsymbol{\lambda} = \mathbf{0.02}$, Bottom Right: $\boldsymbol{\lambda} = \mathbf{0.04}$

6.4.2 Results on *MNIST* dataset

The *MNIST* handwritten digit database consists of **70000** normalized, $\mathbf{28 \times 28}$ grayscale images for handwritten digits from **'0'** to **'9'**. Each image is represented by a **784-D** vector resulting from the normalized, grayscale image. Each

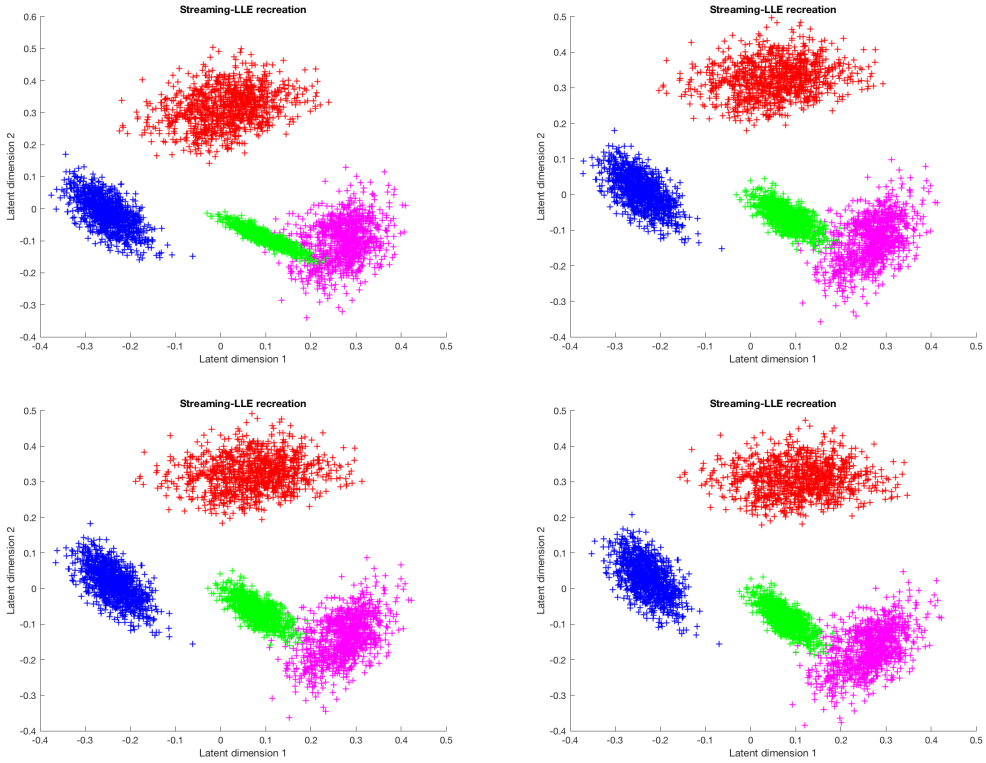


Figure 6.2: Effect of changing **k** on Streaming-LLE. Top Left: $\mathbf{k} = \mathbf{8}$, Top Right: $\mathbf{k} = \mathbf{16}$, Bottom Left: $\mathbf{k} = \mathbf{24}$, Bottom Right: $\mathbf{k} = \mathbf{32}$

of the ten digits has roughly **7000** samples. For our experiment, we considered the first **5** digits i.e. **'0'**, **'1'**, **'2'**, **'3'** and **'4'**. We shuffled the instances for each digit and used $\mathbf{n} = \mathbf{1000}$ instances for each of the **5** digits specified as our training dataset i.e. for the batch training phase and used $\mathbf{m} = \mathbf{1000}$ instances for same digits as our test dataset for the streaming phase and computed the low-dimensional embeddings for them as predicted by the Streaming-LLE algorithm. Figure 6.6 demonstrates the results of this experiment. The uncovered low-dimensional embeddings by Streaming-LLE for each of the digits are continuous and individual manifolds are smooth, which is desirable.

6.4.3 Results on *Gas Sensor Array Drift* dataset

The *Gas Sensor Array Drift* (GSAD) [80] is a benchmark dataset which contains **13910** measurements from **16** chemical sensors utilized in simulations for drift

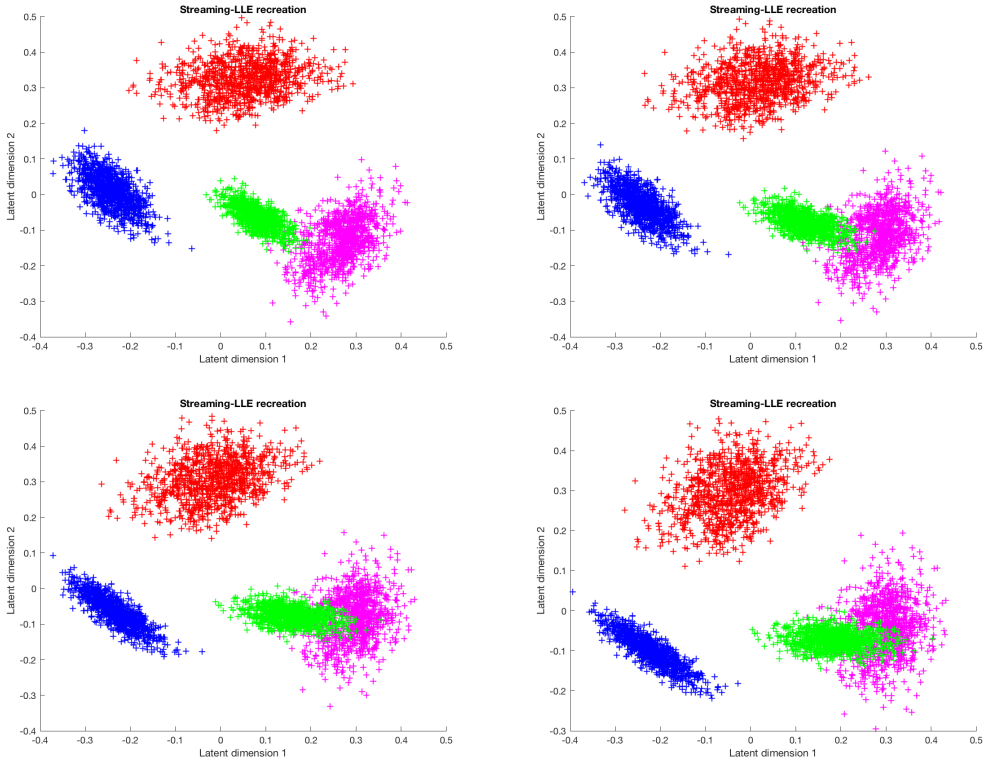


Figure 6.3: Effect of changing $\mathbf{l}$ on Streaming-LLE. Top Left: $\mathbf{l} = \mathbf{1}$, Top Right: $\mathbf{l} = \mathbf{2}$, Bottom Left: $\mathbf{l} = \mathbf{4}$, Bottom Right: $\mathbf{l} = \mathbf{8}$

compensation in a discrimination task of **6** gases at various levels of concentrations. The dataset comprises recordings from six distinct pure gaseous substances, namely *Ammonia*, *Acetaldehyde*, *Acetone*, *Ethylene*, *Ethanol*, and *Toluene*, each dosed at a wide variety of concentration values ranging from **5** to **1000** ppmv. We demonstrate the performance of our proposed method on this dataset. The data was first *mean normalized*. Subsequently, we randomly shuffled the dataset and used $\mathbf{n} = \mathbf{1000}$ instances from the first five classes as the training data and used $\mathbf{m} = \mathbf{1000}$ instances from the same five classes as the test data. We used the training data as the batch dataset for both the S-Isomap++ algorithm as well as the Streaming-LLE algorithm. Subsequently, we computed the low-dimensional embeddings for test dataset in the streaming phase for both algorithms. The results for S-Isomap++ are demonstrated in Figure 6.8 and the results for Streaming-LLE are demonstrated in Figure 6.7. Even though both

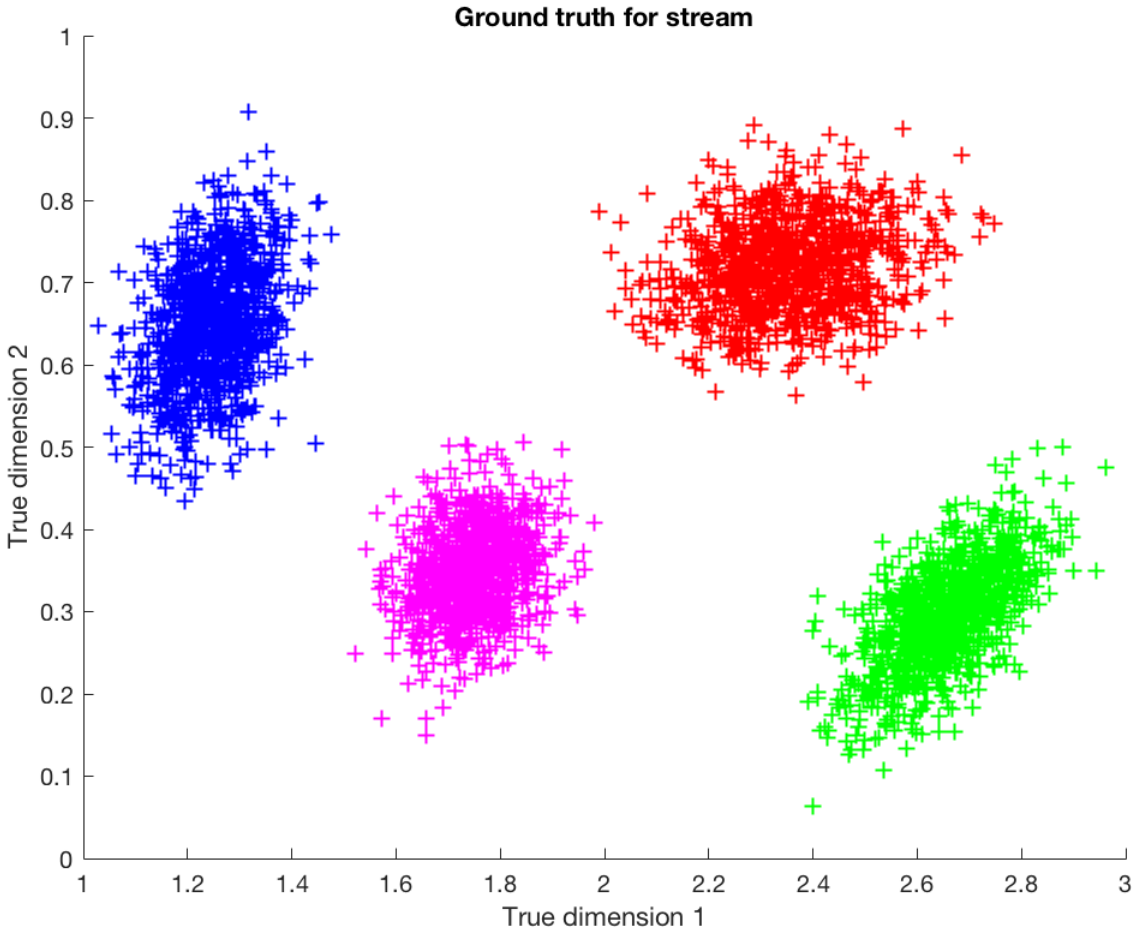


Figure 6.4: The ground truth in $\mathbb{R}^2$ for the streaming samples of the *Euler Isometric Swiss Roll* dataset [65]. The embedding uncovered by different manifold learning algorithms after mapping the streaming samples to $\mathbb{R}^3$ using the non-linear function $\phi()$ is compared to the above.

algorithms i.e. S-Isomap++ and Streaming-LLE are specific instantiations of the same generalized framework, the low-dimensional embedding uncovered by Streaming-LLE is very different from that uncovered by S-Isomap++ i.e. while S-Isomap++ seems to uncover embeddings whose manifolds have smooth surfaces, Streaming-LLE seems to uncover individual manifolds which are linear but disjoint and non-smooth. This is due to the fact that the underlying NLSDR methodologies for both these algorithms are completely different i.e. Isomap for the S-Isomap++ algorithm and LLE for the Streaming-LLE algorithm. Isomap

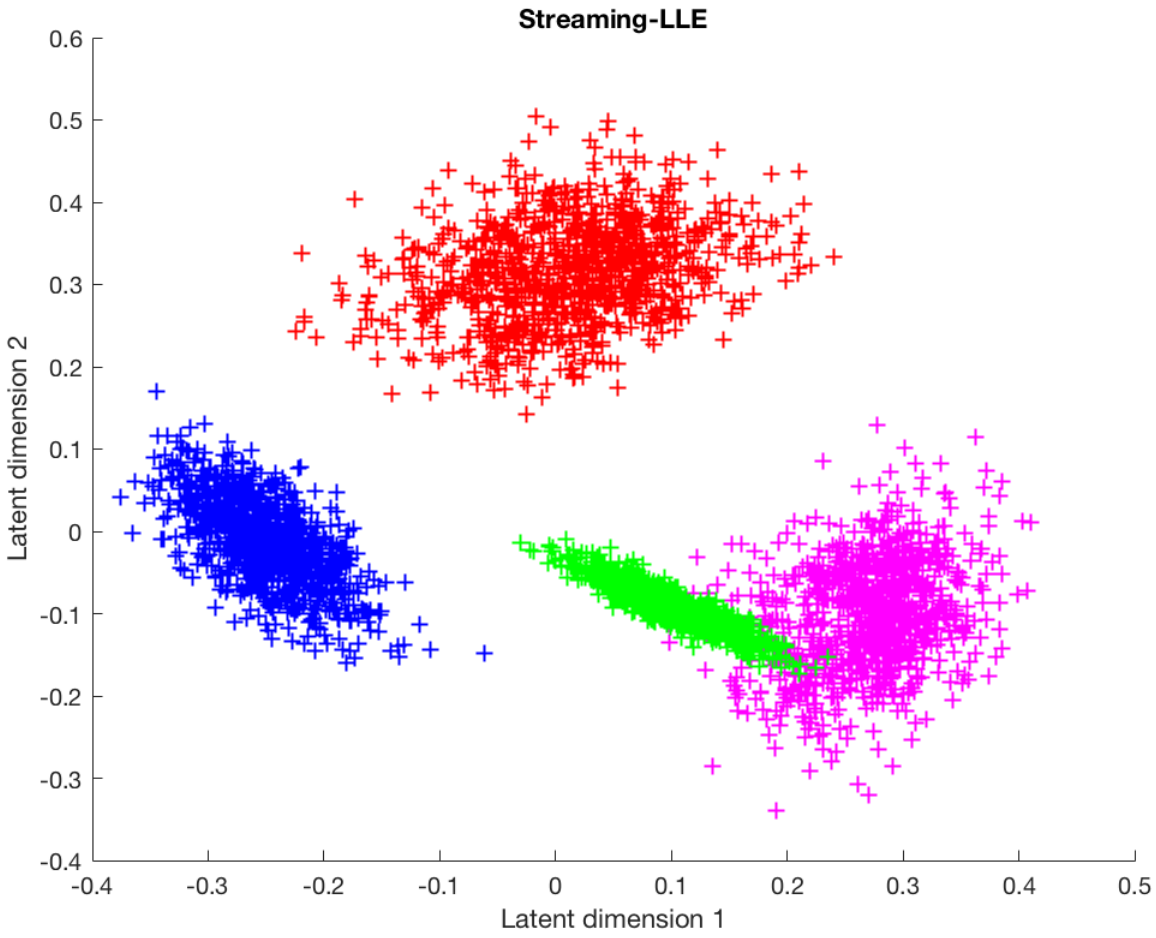


Figure 6.5: The embedding uncovered by the Streaming-LLE algorithm for the streaming samples of the *Euler Isometric Swiss Roll* dataset [65]. The ground truth is demonstrated in Figure 6.4. The results are pretty similar to the results for S-Isomap++ (refer Figure 4.1e)

and LLE work under different sets of assumptions and consequently they interpret the input datasets differently.

6.5 Conclusion

In this chapter, we proposed a generalized OOSE framework for streaming NLSDR, along with a specific instantiation for LLE, proposed as Streaming-LLE. We also demonstrated results of Streaming-LLE on various synthetic and

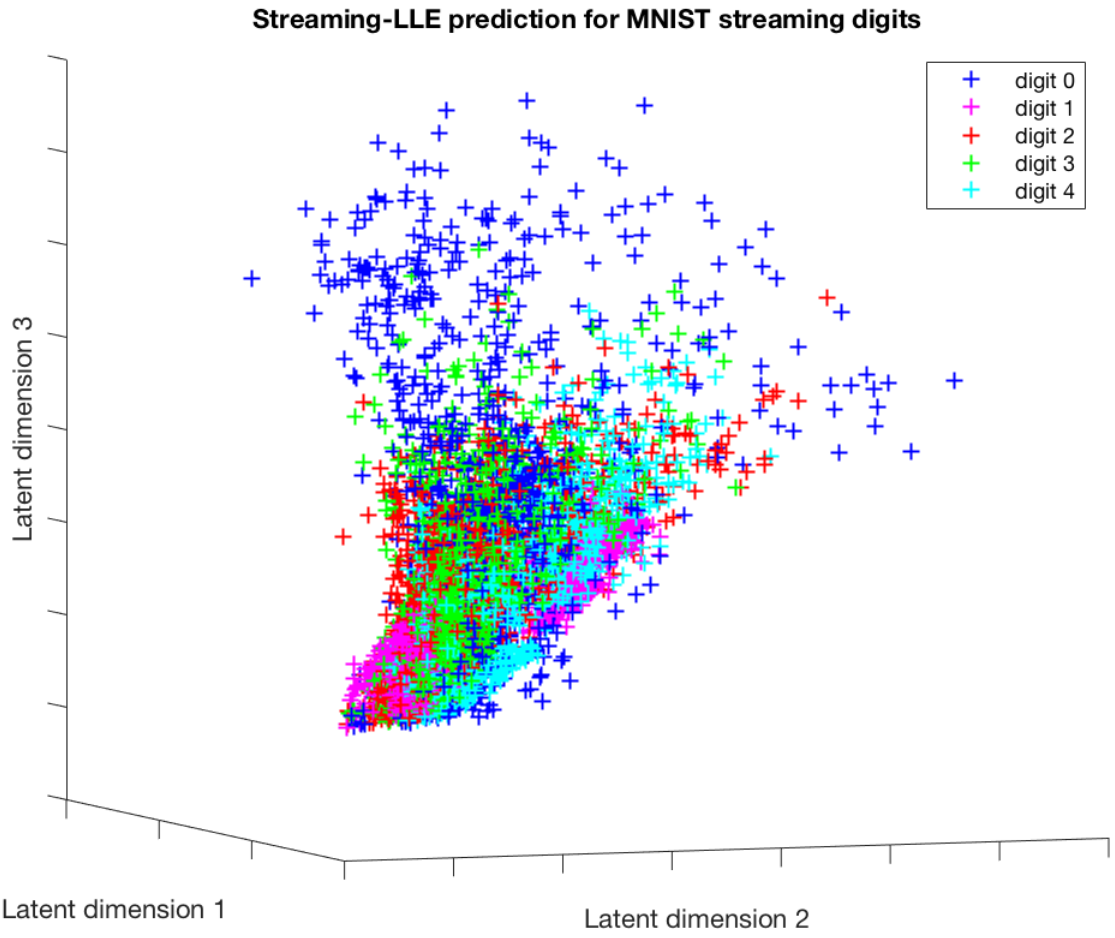


Figure 6.6: Low-dimensional embedding uncovered by the Streaming-LLE algorithm for *MNIST* digits **0–4**. The uncovered low-dimensional embeddings by Streaming-LLE for each of the digits are continuous and individual manifolds are smooth, which is desirable.

benchmark datasets and studied how the different parameters of Streaming-LLE affect embeddings uncovered by the algorithm. We showed how the low-dimensional embedding uncovered by Streaming-LLE is very different from that uncovered by S-Isomap++ even though both algorithms are specific instantiations of the same generalized NLSDR framework. We note here that the skeleton of the GP-Isomap algorithm (outlined in Algorithm 6) similarly motivates a non-parametric generalized OOSE framework for streaming NLSDR for multiple manifolds, wherein the main framework remains fixed and the different

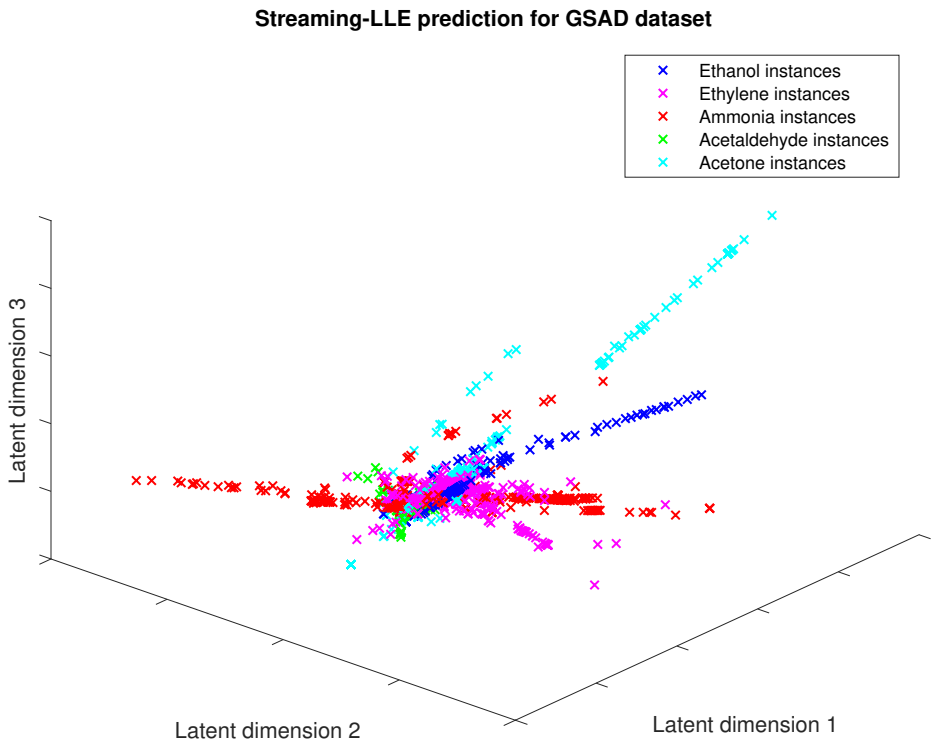


Figure 6.7: Low-dimensional embedding uncovered by the Streaming-LLE algorithm on the *Gas Sensor Array Drift* dataset [80]. Each labelled sample is a **128-D** vector consisting of chemical sensor readings for one of the different gases (*Ethanol*, *Ethylene*, *Ammonia*, *Acetaldehyde*, *Acetone*). S-Isomap++ seems to uncover embeddings whose manifolds have smooth surfaces, while Streaming-LLE seems to uncover individual manifolds which are linear but disjoint and non-smooth.

constituents specific to different manifold learning algorithms can be plugged in as is appropriate. This is part of future research for the current line of work mentioned in this chapter. We also note that the "manifold-stitching" mechanism proposed in both generalized frameworks via using transformation matrices $\{\boldsymbol{\mathcal{R}}_i, \mathbf{t}_i\}_{i=1,2...p}$ to map streaming samples from the "localized manifold spaces" to the "generalized global space", performed well in our study possibly since the underlying manifold learning techniques used were Isomap and LLE which are invariant to these forms of transformations. In future work, wherein we plan to include other NLSDR methods in our proposed generalized framework, we also plan to investigate "manifold-stitching" procedures which might be needed by other NLSDR methods to be able to successfully map stream-

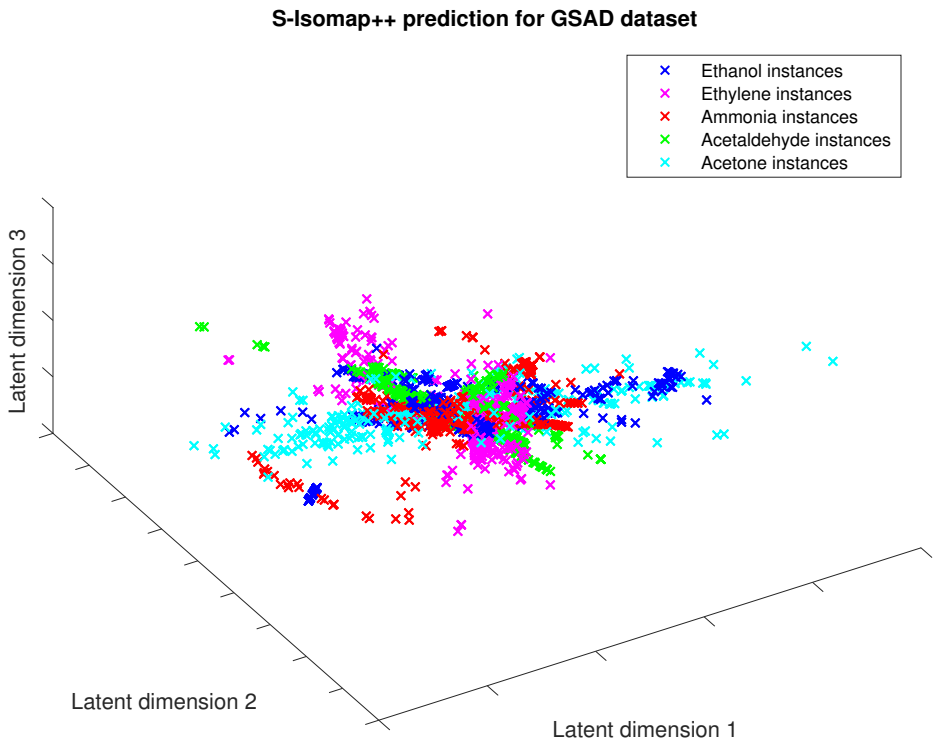


Figure 6.8: Low-dimensional embedding uncovered by the S-Isomap++ algorithm on the *Gas Sensor Array Drift* dataset [80]. Each labelled sample is a **128-D** vector consisting of chemical sensor readings for one of the different gases (*Ethanol*, *Ethylene*, *Ammonia*, *Acetaldehyde*, *Acetone*). The low-dimensional embedding uncovered by Streaming-LLE is very different from that uncovered by S-Isomap++ (refer Figure 6.7), even though both algorithms are specific instantiations of the same generalized NLSDR framework.

ing samples from their "localized manifold spaces" to the "generalized global space".

Algorithm 8 STREAMING-LLE

Require: Batch dataset: $\mathcal{B}$, Streaming dataset: $\mathcal{S}$; Parameters: ϵ, k, l, λ
Ensure: $\mathcal{Y}_{\mathcal{S}}$: low-dimensional representation for $\mathcal{S}$

▷ Batch Phase

$\mathcal{C}_{i=1,2...\mathbf{p}} \leftarrow$ FIND_CLUSTERS($\mathcal{B}$, ϵ)
$\xi_s \leftarrow \emptyset$

for $1 \leq \mathbf{i} \leq \mathbf{p}$ **do**
    $\mathcal{LDE}_i \leftarrow LLE(\mathcal{C}_i)$
end for

$\xi_s \leftarrow \bigcup_{i=1}^{p} \bigcup_{j=i+1}^{p} \mathrm{NN}(\mathcal{C}_i, \mathcal{C}_j, \mathbf{k}) \cup \mathrm{FN}(\mathcal{C}_i, \mathcal{C}_j, \mathbf{l})$
$\mathcal{GE}_s \leftarrow \mathrm{MDS}(\xi_s)$

for $1 \leq \mathbf{j} \leq \mathbf{p}$ **do**
    $\mathcal{I} \leftarrow \xi_s \cap \mathcal{C}_j$
    $\mathcal{A} \leftarrow \begin{bmatrix} \mathcal{LDE}_j^{\mathcal{I}} \\ \mathbf{e}^T \end{bmatrix}$
    $\mathcal{R}_i, \mathbf{t}_i \leftarrow \mathcal{GE}_{\mathcal{I},s} \times \mathcal{A}^T \left(\mathcal{A}\mathcal{A}^T + \lambda \mathbf{I}\right)^{-1}$
end for

▷ Streaming Phase

for $\mathbf{s} \in \mathcal{S}$ **do**
    for $1 \leq \mathbf{i} \leq \mathbf{p}$ **do**
        $y_s^i \leftarrow \mathcal{OOS}_{\mathcal{LLE}}(s, \mathcal{C}_i, \mathcal{LDE}_i)$
        $\mathcal{GE}_s^i \leftarrow \mathcal{R}_i \mathbf{y}_s^i + \mathbf{t}_i$
    end for

    $\mathbf{m} \leftarrow \mathrm{argmin}_i \left| \mathbf{y}_s^i - \mu(\mathcal{C}_i, \mathcal{R}_i, \mathbf{t}_i) \right|$
    $\mathcal{Y}_{\mathcal{S}} \leftarrow \mathcal{Y}_{\mathcal{S}} \cup \mathbf{y}_s^m$
end for

return $\mathcal{Y}_{\mathcal{S}}$

Algorithm 9 Out-of-Sample Extension Methodology for LLE

1: **function** $\mathcal{OOS}_{\mathcal{LLE}}(\mathbf{s}, \mathcal{C}_i, \mathcal{LDE}_i)$
2: $\quad \zeta_s \leftarrow \text{KNN}(\mathbf{s}, \mathcal{C}_i)$
3: $\quad \mathbf{w}^* \leftarrow \underset{w}{\text{argmin}} \left\| \left(\mathbf{s} - \sum_{x_j \in \zeta_s} \mathbf{w}_j \mathbf{x}_j \right) \right\|^2$
4: $\quad$ **return** $\left(\sum_{y_j \in \zeta_s} \mathbf{w}_j^* \mathbf{y}_j \right)$
5: **end function**

Chapter 7

Conclusion

Big data is generally difficult to work with, partly due to the inherent lack of scalability of the different algorithms involved as well as due to the fact that many state-of-the-art algorithms work in a "batch" mode i.e. they do not work well with data streams wherein we cannot possibly hope to include the entire stream as part of the training data. Processing data streams efficiently using standard approaches is also challenging in general, given streams require real-time processing and additionally cannot be stored permanently. Any form of analysis in data streams requires adequate summarization which can deal with the inherent constraints and that can approximate the characteristics of the stream well. One key insight of the work done in this thesis is that, we can in general choose to work with/build our model using only a tiny fraction of the data stream and still be able to learn *adequately* and *effectively* depending on the task at hand, while we choose to process/map the remaining samples of the data stream in a significantly cost-effective manner. To do this, identifying the "point of transition" is crucial. We demonstrate theoretically that such a "point of transition" exists for certain algorithms. Additionally, we provide error metrics to practically identify such *transition points*. This key intuition allowed us to formulate a generalized Out-of-Sample Extension framework for streaming NLSDR as part of the work in the thesis, which can be formulated with different manifold learning algorithms as necessary, and which has the ability to effectively deal with multiple manifolds in a variety of settings resulting from different challenges i.e. data streams which have underlying multi-modal dis-

tributions as well as those wherein non-uniform sampling of underlying data distribution of streams is encountered.

Including other standard state-of-the-art NLSDR methods as part of this generalized Out-of-Sample Extension framework as well as understanding relationships of our proposed approaches with other members of the NLDR family i.e. Variational Auto-encoders, Gaussian Process Latent Variable Models and Deep Gaussian models will be studied as part of future research for the current line of work.

Bibliography

[1] N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. *Journal of Computer sciences*, 58(1):137–147, 1999.

[2] S.-H. Bae, J. Y. Choi, J. Qiu, and G. C. Fox. Dimension reduction and visualization of large high-dimensional data via interpolation. In *Proceedings of the 19th ACM international symposium on high performance distributed computing*, pages 203–214. ACM, 2010.

[3] O. Barkan, J. Weill, and A. Averbuch. Gaussian process regression for out-of-sample extension. In *Machine Learning for Signal Processing (MLSP), 2016 IEEE 26th International Workshop on*, pages 1–6. IEEE, 2016.

[4] M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in neural information processing systems*, pages 585–591, 2002.

[5] Y. Bengio, Y. LeCun, et al. Scaling learning algorithms towards ai. *Large-scale kernel machines*, 34(5):1–41, 2007.

[6] Y. Bengio, J.-f. Paiement, P. Vincent, O. Delalleau, N. L. Roux, and M. Ouimet. Out-of-sample extensions for lle, isomap, mds, eigenmaps, and spectral clustering. In *Advances in neural information processing systems*, pages 177–184, 2004.

[7] M. Bernstein, V. De Silva, J. C. Langford, and J. B. Tenenbaum. Graph approximations to geodesics on embedded manifolds. Technical report, Technical report, Department of Psychology, Stanford University, 2000.

[8] R. Campana-Olivo and V. Manian. Parallel implementation of nonlinear dimensionality reduction methods applied in object segmentation using cuda in gpu. In *Proceedings of SPIE*, volume 8048, 2011.

[9] S. Chaudhuri, R. Motwani, and V. Narasayya. On random sampling over joins. In *ACM SIGMOD Record*, volume 28, pages 263–274. ACM, 1999.

[10] F. Chen, P. Deng, J. Wan, D. Zhang, A. V. Vasilakos, and X. Rong. Data mining for the internet of things: literature review and challenges. *International Journal of Distributed Sensor Networks*, 2015.

[11] L. Chen and A. Buja. Local multidimensional scaling for nonlinear dimension reduction, graph drawing, and proximity analysis. *Journal of the American Statistical Association*, 104(485):209–219, 2009.

[12] H. Choi and S. Choi. Kernel isomap. *Electronics letters*, 40(25):1612–1613, 2004.

[13] J. Y. Choi, S.-H. Bae, X. Qiu, and G. Fox. High performance dimension reduction and visualization for large high-dimensional data analysis. In *Cluster, Cloud and Grid Computing (CCGrid), 2010 10th IEEE/ACM International Conference on*, pages 331–340. IEEE, 2010.

[14] M. Datar, A. Gionis, P. Indyk, and R. Motwani. Maintaining stream statistics over sliding windows. *SIAM journal on computing*, 31(6):1794–1813, 2002.

[15] P. Domingos and G. Hulten. Mining high-speed data streams. In *Proceedings of the sixth ACM SIGKDD international conference on KDD*, pages 71–80. ACM, 2000.

[16] I. L. Dryden and K. V. Mardia. *Statistical shape analysis*, volume 4. Wiley, 1998.

[17] I. L. Dryden and K. V. Mardia. *Statistical Shape Analysis: With Applications in R*. John Wiley & Sons, 2016.

[18] M. Duan, J. Fan, M. Li, L. Han, and S. Huo. Evaluation of dimensionality-reduction methods from peptide folding–unfolding simulations. *Journal of chemical theory and computation*, 9(5):2490–2497, 2013.

[19] E. Elhamifar and R. Vidal. Sparse manifold clustering and embedding. In *Advances in NIPS*, pages 55–63, 2011.

[20] M. Fan, H. Qiao, B. Zhang, and X. Zhang. Isometric multi-manifold learning for feature extraction. In *IEEE 12th ICDM, 2012*, pages 241–250. IEEE, 2012.

[21] M. Fan, X. Zhang, H. Qiao, and B. Zhang. Efficient isometric multi-manifold learning based on the self-organizing method. *Inf. Sci.*, 345(C):325–339, 2016.

[22] C. Fowlkes, S. Belongie, F. Chung, and J. Malik. Spectral grouping using the nystrom method. *IEEE transactions on pattern analysis and machine intelligence*, 26(2):214–225, 2004.

[23] Y. Goldberg and Y. Ritov. Local procrustes for manifold embedding: a measure of embedding quality and embedding algorithms. *Machine learning*, 77(1):1–25, 2009.

[24] H. Gunawan, O. Neswan, and W. Setya-Budhi. A formula for angles between subspaces of inner product spaces. *Contributions to Algebra and Geometry*, 46(2):311–320, 2005.

[25] J. Ham, D. D. Lee, S. Mika, and B. Schölkopf. A kernel view of the dimensionality reduction of manifolds. In *Proceedings of the twenty-first ICML*. ACM, 2004.

[26] J. Ham, D. D. Lee, and L. K. Saul. Semisupervised alignment of manifolds. In *AISTATS*, pages 120–127, 2005.

[27] R. Hettiarachchi and J. F. Peters. Multi-manifold lle learning in pattern recognition. *Pattern Recognition*, 48(9):2947–2960, 2015.

[28] G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[29] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[30] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American statistical association*, 58(301):13–30, 1963.

[31] H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6):417, 1933.

[32] H. Huang, L. ten Bosch, B. Cranen, and L. Boves. Phone classification via manifold learning based dimensionality reduction algorithms. *Speech Communication*, 76:28–41, 2016.

[33] A. Hyvärinen, J. Karhunen, and E. Oja. *Independent component analysis*, volume 46. John Wiley & Sons, 2004.

[34] H. V. Jagadish, N. Koudas, S. Muthukrishnan, V. Poosala, K. C. Sevcik, and T. Suel. Optimal histograms with quality guarantees. In *VLDB*, volume 98, 1998.

[35] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323, 1999.

[36] O. C. Jenkins and M. J. Matarić. A spatio-temporal extension to isomap nonlinear dimension reduction. In *Proceedings of the twenty-first international conference on Machine learning*, page 56. ACM, 2004.

[37] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[38] R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artificial intelligence*, 97(1-2):273–324, 1997.

[39] D. Kong, C. H. Ding, H. Huang, and F. Nie. An iterative locally linear embedding algorithm. *arXiv preprint arXiv:1206.6463*, 2012.

[40] O. Kouropteva, O. Okun, and M. Pietikäinen. Incremental locally linear embedding algorithm. *Image Analysis*, pages 145–159, 2005.

[41] J. B. Kruskal and M. Wish. *Multidimensional scaling*, volume 11. Sage, 1978.

[42] S. Kumar, M. Mohri, and A. Talwalkar. Sampling methods for the nyström method. *Journal of Machine Learning Research*, 13(Apr):981–1006, 2012.

[43] M. H. Law and A. K. Jain. Incremental nonlinear dimensionality reduction by manifold learning. *IEEE transactions on pattern analysis and machine intelligence*, 28(3):377–391, 2006.

[44] G. Lee, C. Rodriguez, and A. Madabhushi. Investigating the efficacy of nonlinear dimensionality reduction schemes in classifying gene and protein expression studies. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 5(3):368–384, 2008.

[45] J. A. Lee and M. Verleysen. *Nonlinear dimensionality reduction*. Springer Science & Business Media, 2007.

[46] Y.-K. Lei, Z.-H. You, Z. Ji, L. Zhu, and D.-S. Huang. Assessing and predicting protein interactions by combining manifold embedding with multiple information integration. *BMC bioinformatics*, 13(7):S3, 2012.

[47] W. Li, S. Prasad, J. E. Fowler, and L. M. Bruce. Locality-preserving dimensionality reduction and classification for hyperspectral image analysis. *IEEE Transactions on Geoscience and Remote Sensing*, 50(4):1185–1198, 2012.

[48] A. V. Little, J. Lee, Y.-M. Jung, and M. Maggioni. Estimation of intrinsic dimensionality of samples from noisy low-dimensional manifolds in high dimensions with multiscale svd. In *IEEE/SP 15th Workshop on SSP'09*, pages 85–88. IEEE, 2009.

[49] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[50] S. Mahapatra and V. Chandola. S-isomap++: Multi manifold learning from streaming data. 2017.

[51] S. Mahapatra and V. Chandola. Learning manifolds from non-stationary streaming data. *arXiv preprint arXiv:1804.08833*, 2018.

[52] K. V. Mardia, J. T. Kent, and J. M. Bibby. Multivariate analysis (probability and mathematical statistics), 1980.

[53] C. Orsenigo. An improved set covering problem for isomap supervised landmark selection. *Pattern Recognition Letters*, 49:131–137, 2014.

[54] C. Orsenigo and C. Vercellis. A comparative study of nonlinear manifold learning methods for cancer microarray data classification. *Expert systems with Applications*, 40(6):2189–2197, 2013.

[55] J. C. Platt. Fast embedding of sparse similarity graphs. In *Advances in neural information processing systems*, pages 571–578, 2004.

[56] M. Polito and P. Perona. Grouping and dimensionality reduction by locally linear embedding. In *Advances in NIPS*, pages 1255–1262, 2002.

[57] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical recipes in C*, volume 2. Cambridge university press Cambridge, 1996.

[58] N. Qi, Z. Zhang, Y. Xiang, and P. d. B. Harrington. Locally linear embedding method for dimensionality reduction of tissue sections of endometrial carcinoma by near infrared spectroscopy. *Analytica chimica acta*, 724:12–19, 2012.

[59] D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

[60] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500):2323–2326, 2000.

[61] Y. Ruan, G. L. House, S. Ekanayake, U. Schutte, J. D. Bever, H. Tang, and G. Fox. Integration of clustering and multidimensional scaling to determine phylogenetic trees as spherical phylograms visualized in 3 dimensions. In *Cluster, Cloud and Grid Computing (CCGrid), 2014 14th IEEE/ACM International Symposium on*, pages 720–729. IEEE, 2014.

[62] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[63] O. Samko, A. D. Marshall, and P. L. Rosin. Selection of the optimal parameter value for the isomap algorithm. *Pattern Recognition Letters*, 27(9):968–979, 2006.

[64] L. K. Saul and S. T. Roweis. Think globally, fit locally: unsupervised learning of low dimensional manifolds. *Journal of machine learning research*, 4(Jun):119–155, 2003.

[65] F. Schoeneman, S. Mahapatra, V. Chandola, N. Napp, and J. Zola. Error metrics for learning reliable manifolds from streaming data. In *Proceedings of 2017 SDM*, pages 750–758. SIAM, 2017.

[66] J. Shawe-Taylor and C. Williams. The stability of kernel principal components analysis and its relation to the process eigenspectrum. *Advances in NIPS*, pages 383–390, 2003.

[67] R. Sibson. Studies in the robustness of multidimensional scaling: Perturbational analysis of classical scaling. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 217–229, 1979.

[68] V. D. Silva and J. B. Tenenbaum. Global versus local methods in nonlinear dimensionality reduction. In *Advances in neural information processing systems*, pages 721–728, 2003.

[69] R. Souvenir and R. Pless. Manifold clustering. In *Tenth IEEE ICCV, 2005*, volume 1, pages 648–653. IEEE, 2005.

[70] M. Spivak. A comprehensive introduction to differential geometry, vol. 1, 3rd edition, publish or perish, berkeley, 1979. *Google Scholar*.

[71] H. Strange and R. Zwiggelaar. A generalised solution to the out-of-sample extension problem in manifold learning. In *AAAI*, pages 293–296, 2011.

[72] H. Strange and R. Zwiggelaar. *Open Problems in Spectral Dimensionality Reduction*. Springer, 2014.

[73] A. Talwalkar, S. Kumar, and H. Rowley. Large-scale manifold learning. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[74] J. B. Tenenbaum, V. De Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500):2319–2323, 2000.

[75] B. Thirion and O. Faugeras. Nonlinear dimension reduction of fmri data: the laplacian embedding approach. In *Biomedical Imaging: Nano to Macro, 2004. IEEE International Symposium on*, pages 372–375. IEEE, 2004.

[76] M. Torki, A. Elgammal, and C. S. Lee. Learning a joint manifold representation from multiple data sets. In *20th ICPR, 2010*, pages 1068–1071. IEEE, 2010.

[77] L. Van Der Maaten, E. Postma, and J. Van den Herik. Dimensionality reduction: a comparative review. *Journal of Machine Learning Research*, 10:66–71, 2009.

[78] M. S. Venkatarajan and W. Braun. New quantitative descriptors of amino acids based on multidimensional scaling of a large number of physical–chemical properties. *Journal of Molecular Modeling*, 7(12):445–453, 2001.

[79] J. Venna and S. Kaski. Local multidimensional scaling. *Neural Networks*, 19(6):889–899, 2006.

[80] A. Vergara, S. Vembu, T. Ayhan, M. A. Ryan, M. L. Homer, and R. Huerta. Chemical gas sensor drift compensation using classifier ensembles. *Sensors and Actuators B: Chemical*, 166:320–329, 2012.

[81] J. S. Vitter. Random sampling with a reservoir. *ACM Transactions on Mathematical Software (TOMS)*, 11(1):37–57, 1985.

[82] C. Wang and S. Mahadevan. Manifold alignment using procrustes analysis. In *Proceedings of the 25th ICML*, pages 1120–1127. ACM, 2008.

[83] K. Q. Weinberger, B. Packer, and L. K. Saul. Nonlinear dimensionality reduction by semidefinite programming and kernel matrix factorization. In *AISTATS*, 2005.

[84] C. K. Williams. On a connection between kernel pca and metric multidimensional scaling. *Machine Learning*, 46(1-3):11–19, 2002.

[85] C. K. Williams and M. Seeger. Using the nyström method to speed up kernel machines. In *Advances in NIPS*, pages 682–688, 2001.

[86] Y. Wu and K. L. Chan. An extended isomap algorithm for learning multiclass manifold. In *Machine Learning and Cybernetics, 2004. Proceedings of 2004 International Conference on*, volume 6, pages 3429–3433. IEEE, 2004.

[87] W. Xing, A. A. Shah, and P. B. Nair. Reduced dimensional gaussian process emulators of parametrized partial differential equations based on isomap. In *Proceedings of the Royal Society of London A: Sciences*, volume 471. The Royal Society, 2015.

[88] L. Yang. Building k edge-disjoint spanning trees of minimum total length for isometric data embedding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1680–1683, 2005.

[89] W. Yang, C. Sun, and L. Zhang. A multi-manifold discriminant analysis method for image feature extraction. *Pattern Recognition*, 44(8):1649–1657, 2011.

[90] Z. Zhang and H. Zha. Principal manifolds and nonlinear dimensionality reduction via tangent space alignment. *SIAM journal on scientific computing*, 26(1):313–338, 2004.