Incremental Dimensionality Reduction: Algorithms and Applications

Osama Abdel-Mannan

A Thesis

in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of Master of Applied Science (Electrical and Computer Engineering) at

Concordia University

Montréal, Québec, Canada

June 2007

Your file Votre référence
ISBN: 978-0-494-34426-2
Our file Notre référence
ISBN: 978-0-494-34426-2

Abstract

Incremental Dimensionality Reduction: Algorithms and Applications

Osama Abdel-Mannan

One key challenge in data mining and manifold learning applications is the ability to manipulate datasets and to extract useful information. This triggered the need for dimensionality reduction techniques to help visualize and analyze the data without excess computation time. Several methods have been proposed in recent years that are based on the batch mode, meaning that all the data points need to be present during the run of the algorithm. Thus, any newly arriving point has to be added to the dataset and the algorithm has to be rerun on the entire set to produce the output. This limitation gave rise to the incremental form of some existing algorithms enabling them to adapt to newly added points without loss of computational time.

Motivated by this incremental extension of the algorithms, this thesis addresses the limitations of some dimensionality reduction techniques and proposes incremental solutions. The core idea behind our proposed techniques is to enable these methods to adapt to the stream of data being added while preserving the significant characteristics of the low dimensional representation of the dataset. Our experimental results demonstrate an improved performance of the proposed approaches in comparison with existing algorithms.

Acknowledgements

First, praise be to God who has given me the strength and will to complete my work and aided me in all my endeavors.

I would like to express my deepest gratitude to my advisors, Dr. A. Ben Hamza and Dr. Amr Youssef, for their support and patience during my studies. Throughout my work, they provided encouragement and kind advice which lead to the completion of this thesis.

I am also grateful to my friends who gave me all the support I need and whose friendship is irreplaceable.

Lastly, and most importantly, to my family: my mother, my father and my sister, to whom I am forever indebted. They were always there for me and offered me everything one can ever wish for.

Thank you all.

Table of Contents

List of Figures

List of Abbreviations

PCA: Principal Component Analysis

KPCA: Kernel Principal Component Analysis

IKPCA: Incremental Kernel Principal Component Analysis

IPCA: Incremental Principal Component Analysis

Isomap: Isometric Mapping

MDS: Multidimensional Scaling

LLE: Local Linear Embedding

HLLE: Hessian Local Linear Embedding

LTSA: Local Tangent Space Alignment

NMF: Nonnegative Matrix Factorization

SVD: Singular Value Decomposition

CHAPTER 1

Introduction

The demand for data collection has been greatly increasing with the development of new technologies. Almost every new experiment or application requires the availability of considerable amount of data for the purpose of testing and analysis. With this acquisition of huge amounts of data, the need arises for controlling this data and extracting the necessary information from it. Depending on the type of data, various techniques of data analysis are used to extract useful information and facilitate conclusions. Usually, the data acquired from different real-world applications have high dimensionality. The study of high-dimensional manifolds [1], [2] is of great significance in many engineering applications, including data mining [3], data visualization, and pattern recognition [4]. Hence, the understanding of the behavior of these manifolds, and the algorithms proposed for manifold learning, is very important. Therefore, this requires selectivity when it comes to analyzing and using the data for a certain experiment. Being selective means to choose the significant characteristics of the dataset in order to analyze it in a simple way. One way is to reduce the dimension of

high-dimensional data in order to better visualize and analyze its behavior with little loss of information. Here comes the role of dimensionality reduction techniques. Dimensionality reduction is an important preprocessing step before classifying multidimensional data. Dimensionality reduction is primarily used to transform a high-dimensional dataset into a low-dimensional space while preserving the significant characteristics of the data structure. This helps in discovering the structure and behavior of the manifold in a lower and hence a simpler dimension for the purpose of learning, analyzing, and experimenting.

Over the years, many methods of dimensionality reduction have been proposed. These include Principal Component Analysis (PCA) [5], [6], Kernel PCA [7], [8], Isometric Mapping (Isomap) [9], Local Linear Embedding (LLE) [10], Hessian Local Linear Embedding (HLLE) [11], Local Tangent Space Alignment (LTSA) [13], and Laplacian Eigenmap [14]. However many of these algorithms operate in a batch mode, meaning that all data points need to be available during training. When data points arrive sequentially, these methods are computationally demanding and time consuming and need to repeatedly run algorithm whenever new data points are obtained.

In this chapter, we briefly review some dimensionality reduction algorithm, introduce their limitations, and propose incremental versions in order to adapt to newly added points. Motivated by the incremental version of the Local Linear Embedding (LLE) algorithm [15],

we propose an incremental version of Hessian LLE (HLLE) and Local Tangent Space Alignment (LTSA). The core idea behind the proposed technique is to enable these algorithms to adapt to newly added points as they arrive in order to better visualize the evolution of the dataset.

A major part of this thesis is devoted to implementing the incremental version of these algorithms and illustrate the results obtained. The proposed algorithms to some extent preserve the characteristics of the lower dimension representation after the addition of the incremented data points.

1.1 Background

This thesis addresses the application of dimensionality reduction techniques on high-dimensional datasets. The following background material is presented to provide context for this work.

1.1.1 Data Mining

Storing, modelling, and understanding data is a common concern for many research areas. There has been a considerable growth in acquiring data from different areas and for different objectives. With that, there has been an increased interest in utilizing data and extracting valuable information from them. The area related to such tasks is called data mining. The

main idea behind data mining is to extract analytical information from large databases. Data mining can be seen as an intersection of disciplines such as machine learning, statistics, pattern recognition, and other areas. It helps focus on the most important information. The tools used in data mining are useful in determining the trends and behaviors of datasets, allowing for future decisions and further analysis. Generally, data mining is used to analyze data from different perspectives and summarize it into useful information [3].

Datasets usually represent measurements taken from a certain process or environment. A major problem in mining scientific datasets is that the data is often high-dimensional, meaning there are a large number of features representing the object. As the number of dimensions increases the computational time for pattern recognition algorithms can become excessive. To address the problem of high dimensionality, a common approach is to identify the most important features associated with every object so that further processing can be simplified without compromising the quality of the final results.

1.1.2 Principal Component Analysis (PCA)

Principal component analysis is an explanatory technique that helps in learning about datasets. The objective of principal component analysis is to reduce the dimensionality of the dataset while retaining, as much as possible, the variation in the dataset [5], [6].

1.1 Background

Principal components are linear transformations of the original set of variables, and are uncorrelated and ordered so that the first few components carry most of the variations in the original dataset. The first principal component has the geometric interpretation that it is a new coordinate axis that maximizes the variation of the projections of the data points on the new coordinate axis. Figure 1.1 shows a scatter plot of data points having an elliptical shape. Note that, if the data points are projected onto the first principal component, most of the variation of the 2-D data points would be captured in one dimension. The first principal component e_1 is the direction of the greatest variation among the entire dataset.

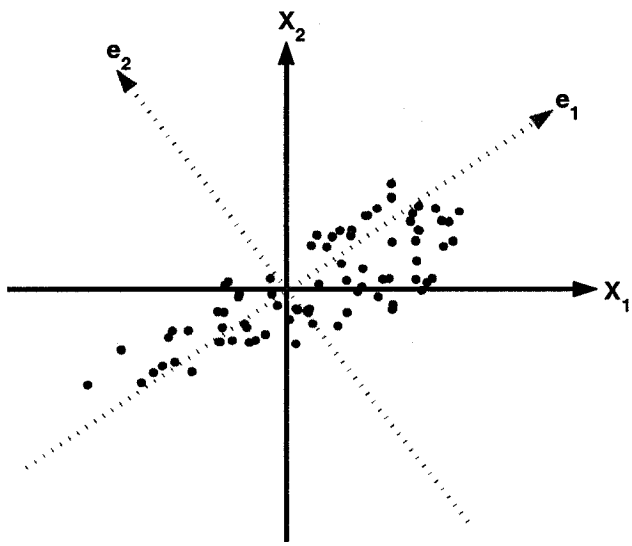


Figure 1.1: Illustration of PCA.

The goal of the dimension reduction can be explanation, visualization or further processing, like feature extraction. The principal component analysis finds basis vectors for a

subspace which maximize the variance retained in the projected data.

Given a dataset $D = \{X_i : i = 1, \ldots, N\}$ of p-dimensional vectors, the principal Component Analysis (PCA) is a linear projection technique

$$Y_i = Q(X_i - \overline{X}),$$

where Y_i is a q-dimensional projected data vector (usually $q << p$), X_i is the original date vector, $\overline{X}$ is its sample mean, and Q is a $N \times q$ matrix that contains the PCA projection vectors.

The q-projection vectors that maximize the variance of Y_i, i.e., the principal axes, are given by the eigenvectors of the sample covariance matrix. These eigenvectors correspond to the q largest nonzero eigenvalues.

The steps that constitute the PCA algorithm are summarized as follows:

- Compute sample mean $\overline{X} = (1/N) \sum_{i=1}^{N} X_i$
- Compute sample covariance $S = (1/N) \sum_{i=1}^{N} (X_i - \overline{X})(X_i - \overline{X})^T$
- Compute the eigenvalues λ_i and eigenvectors $\boldsymbol{e}_i$, that is, $S\boldsymbol{e}_i = \lambda_i \boldsymbol{e}_i$. The first eigenvector of a covariance matrix defines a variable with the most variation in the dataset. The eigenvectors are the principal components, and the eigenvalues measure the importance of the principal components. The eigenvalues are usually arranged in a nonincreasing order, i.e. $\lambda_1 \geq \ldots \geq \lambda_N$. The eigenvalues tell us about the amount of variation in data, and variance in a particular direction.
- Project onto q principal components: $Y_i = E_q^T (X_i - \overline{X})$, where $Q = E_q =$

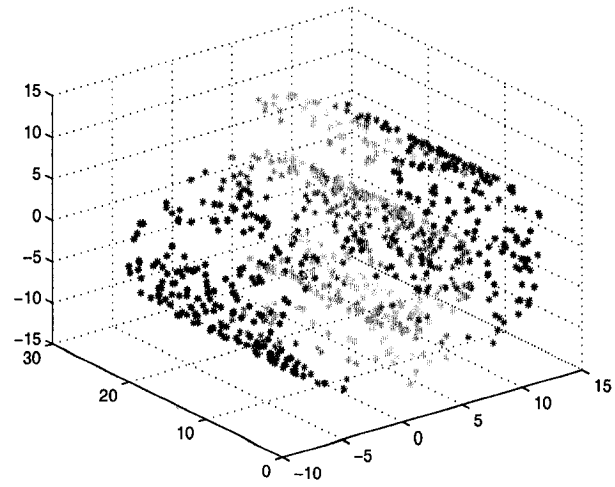
$(\boldsymbol{e}_1, \ldots, \boldsymbol{e}_q)$ is an $N \times q$ matrix of the retained q principal components that carry most of the variance. In other words, the first principal component accounts for as much of the variability in the data as possible, and each succeeding component accounts for as much of the remaining variability as possible. After calculating the eigenvectors, we sort them by their corresponding eigenvalues and then we pick the q vectors with the largest eigenvalues.

PCA also gives uncorrelated projected distributions. Besides retaining the maximum amount of variance in the projected data, PCA also has the following property: the projected data Y_i is decorrelated, i.e. the covariance matrix $E(Y_i Y_i^T)$ is a diagonal matrix. The variance in each principal direction can also be normalized, by using $Q = \Lambda^{-1/2} E_q^T$ instead of $Q = E_q^T$, where Λ is a diagonal matrix with the eigenvalues of sample covariance matrix. In this case, the covariance matrix of $\{Y_i : i = 1, \ldots, N\}$ is equal to the identity matrix.

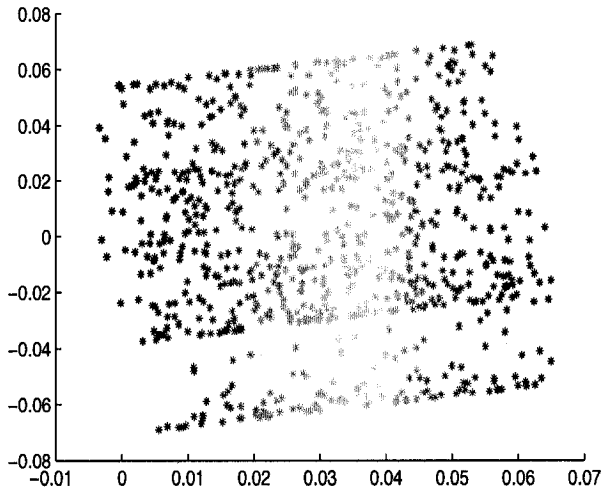
Moreover, PCA minimizes the least square reconstruction error. If the projected vectors Y_i are projected back into the original space using

$$\widetilde{X}_i = MQ(X_i - \overline{X}),$$

where $M = Q^T(QQ^T)^{-1}$, then the reconstruction error, $\|(X_i - \overline{X}) - \widetilde{X}_i\|$, is minimized.



(a)



(b)

Figure 1.2: PCA implemented on swissroll dataset. (a) 3D swissroll dataset, (b) The 2D representation result of PCA applied on the swissroll dataset.

Dimensionality reduction with PCA

The question comes when choosing a value for q. The proportion of variance retained by mapping down to q dimensions can be found as the normalized sum of the q largest eigenvalues

$$r = \frac{\sum_{i=1}^{q} \lambda_i}{\sum_{i=1}^{p} \lambda_i} \times 100\%.$$

In many applications, q is chosen such that $95\% \geq r \geq 90\%$ variance is retained. The remaining variance is assumed to be due to noise.

The number of principal components q, $q \ll p$, can be determined using the scree graph which is the plot of the k-th eigenvalue versus the component number k, or retain q components that account for (e.g. $65 - 90\%$) of variation. The lower dimensional vector $\hat{X}_i$ captures the most important features of the original data X_i. Figure 1.2 shows a swissroll dataset reduced from 3-dimension to 2-dimension using PCA.

When applied directly to image data, PCA projection vectors are global in the image domain. This fact has been exploited in many applications. For example, in face recognition, the first eigenvector ("eigenface") corresponds to the basic face shape all humans share, and further eigenvectors correspond to variations between individual faces (hair, glasses etc.)

1.1.3 Isometric Mapping (Isomap)

Isomap [9] is a dimensionality reduction technique that uses geodesic distances [20] instead of Euclidean distances. Geodesic distances represent the shortest paths along the curved

surface of the manifold, as in Figure 1.3. Isomap uses Multidimensional Scaling (MDS) techniques in its implementation [3], [16]. Unlike linear techniques, Isomap can discover the nonlinear degrees of freedom that underlie complex natural observations [17], [9].

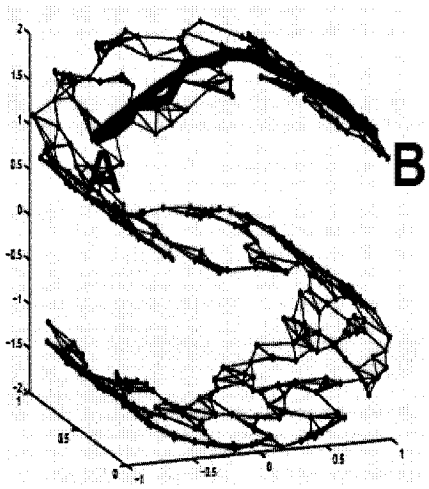
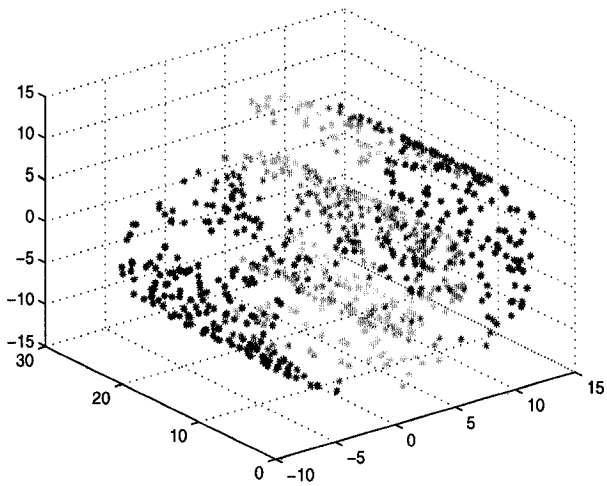


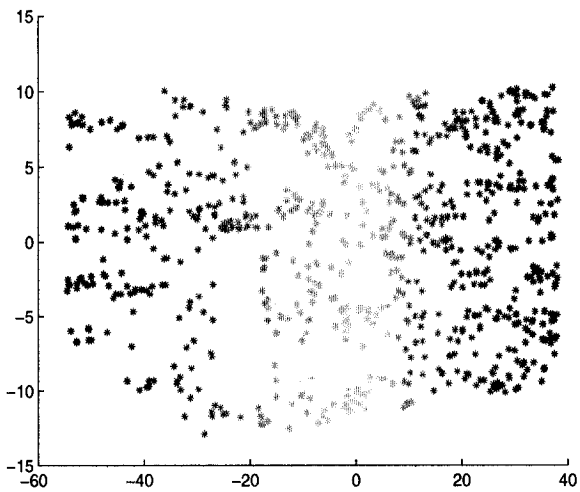
Figure 1.3: Finding Geodesic distances used by Isomap [20]

The Isomap algorithm can be summarized as follows:

- The points that are neighbors on the manifold are determined based on the distances between pairs of points in the input space.
- The geodesic distances between all pairs of points on the manifold are estimated by computing their shortest path distances [20].
- MDS is applied to matrix of graph distances [18] , constructing an embedding of



(a)



(b)

Figure 1.4: Isomap implemented on swissroll dataset. (a) 3D swissroll dataset, (b) The 2D representation result of Isomap applied on the swissroll dataset.

the data in a low-dimensional Euclidean space that best preserves the manifold's estimated intrinsic geometry. Figure 1.4 shows a swissroll dataset reduced from 3-dimension to 2-dimension using Isomap.

1.1.4 Local Linear Embedding (LLE)

LLE [10] is a dimensionality reduction technique that computes low-dimensional, neighborhood preserving embedding of high-dimensional datasets. LLE succeeds in identifying the underlying structure of the manifold. Unlike Isomap, LLE eliminates the need to estimate pairwise distances between widely separated data points.

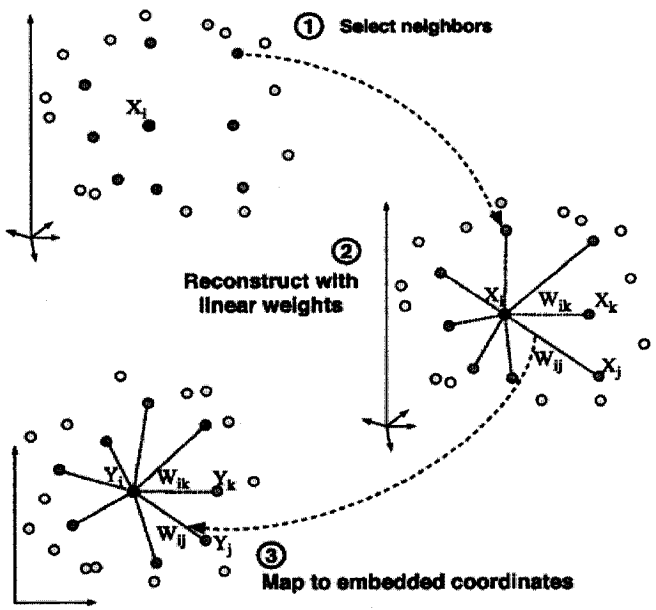
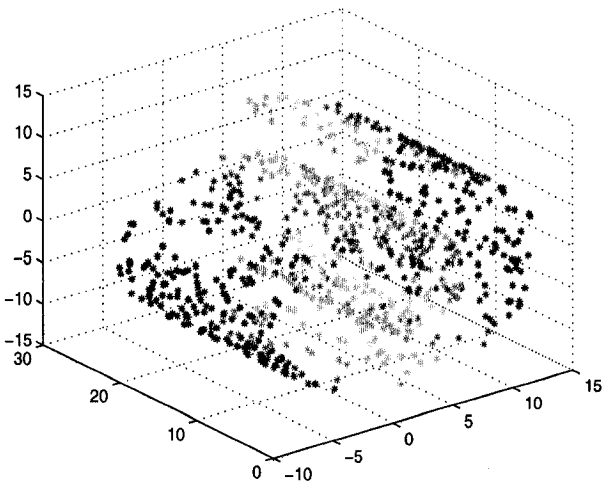
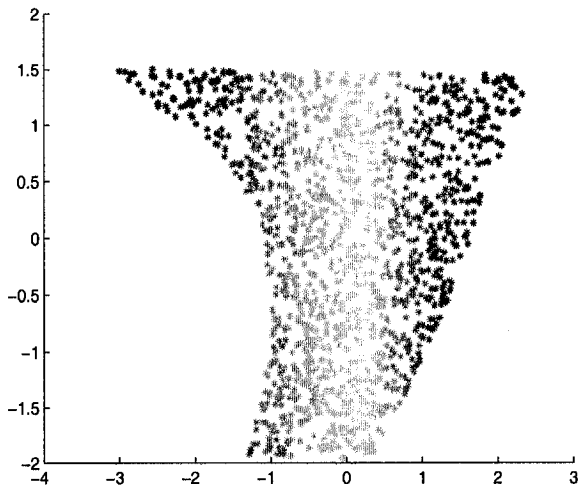


Figure 1.5: LLE steps [10]



(a)



(b)

Figure 1.6: LLE implemented on swissroll dataset. (a) 3D swissroll dataset, (b) The 2D representation result of LLE applied on the swissroll dataset.

The local geometry of the data points, $X = \{x_1, x_2, \ldots, x_i\}$, can be characterized by linear coefficients that reconstruct each data point from its neighbors. The k-nearest neighbors per data point are identified, as measured by Euclidean distance. From these distances between points, a weights matrix ,$W = \{W_{ij}\}$, is formed that best reconstructs each data point x_i from its neighbors. The reconstruction weights, W_{ij}, reflect intrinsic geometric properties of the data that are not affected by transformations such as translation, rotation, and scaling. Thus, the same weights that reconstruct a data point in the higher dimension should also reconstruct its embedded coordinates in the lower dimension. Finally, from these weights, the high-dimensional points are mapped into low-dimensional vectors. Each of the vectors $Y = \{y_1, y_2, \ldots, y_i\}$ that are best reconstructed by the weights W_{ij} are computed. Figure 1.5 shows the steps involed in the LLE algorithm, and Figure 1.6 shows LLE applied on the swissroll dataset.

1.1.5 Hessian Local Linear Embedding (HLLE)

HLLE [11] derives from a context of local isometry in which the manifold (viewed as a manifold in Euclidean space $\mathbb{R}^n$) is locally isometric to an open, connected subset Θ of Euclidean space $\mathbb{R}^d$, where $d < n$. The theoretical framework revolves around a quadratic form $\mathcal{H}(f) = \int_M \|H_f(m)\|_F^2 dm$ defined on function $f : M \longmapsto \mathbb{R}$. Here H_f denotes the Hessian of f, and $\mathcal{H}(f)$ averages the Frobenius norm [12] of the Hessian over M. To define the Hessian, orthogonal coordinates on the tangent planes of M are used.

1.1 Background

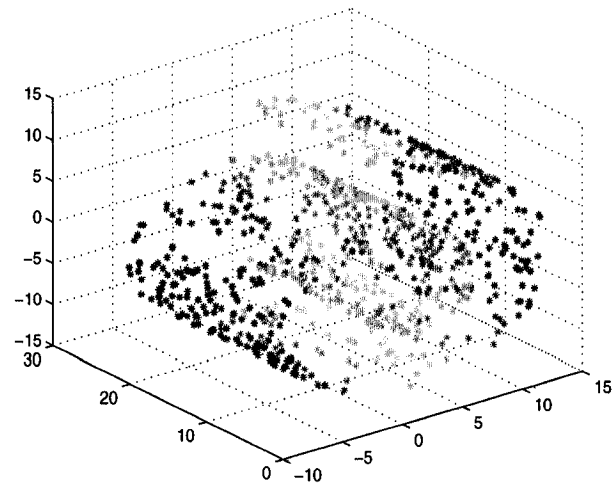
If M is locally isometric to an open, connected subset of $\mathbb{R}^d$, then $\mathcal{H}(f)$ has a $(d+1)$-dimensional null space consisting of the constant functions and a d-dimensional space of functions spanned by the original isometric coordinates. Hence, the isometric coordinates can be recovered up to a linear isometry. HLLE can be viewed as a modification of Locally Linear Embedding (LLE).

For a data input m_i of N points in $\mathbb{R}^n$ producing output data points w_i of N points in $\mathbb{R}^d$, the HLLE algorithm is summarized as follows:

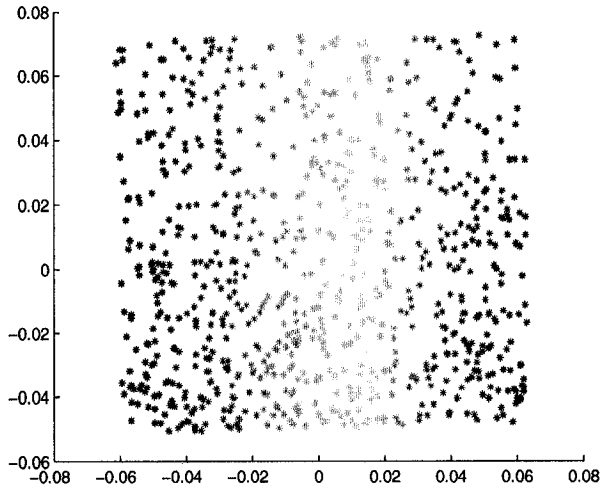
- The k-nearest neighbors of the data points are identified
- Singular decomposition of the manifold is performed to obtain the tangent coordinates, producing matrices U, D, and V.
- The Hessian estimator H^i matrix is developed.
- A quadratic form is developed by building a symmetric matrix

$$\mathcal{H}_{ij} = \sum_l \sum_r (H^l_{r,i} H^l_{r,j})$$

  where H^l is a $\frac{d(d+1)}{2} \times k$ matrix associated with estimating the Hessian over neighborhood N_l. The entries in the Hessian matrix correspond to rows r, and the specific points in the neighborhood correspond to columns i.
- An approximate null space is found by performing eigenanalysis of $\mathcal{H}$ and identifying the $(d+1)$-dimensional subspace corresponding to the $(d+1)$-smallest eigenvalues.
- The basis for the null space is found. The given basis has basis vectors $w^1, \ldots, w^d$,



(a)



(b)

Figure 1.7: HLLE implemented on swissroll dataset. (a) 3D swissroll dataset, (b) The 2D representation result of HLLE applied on the swissroll dataset.

which are the embedding coordinates.

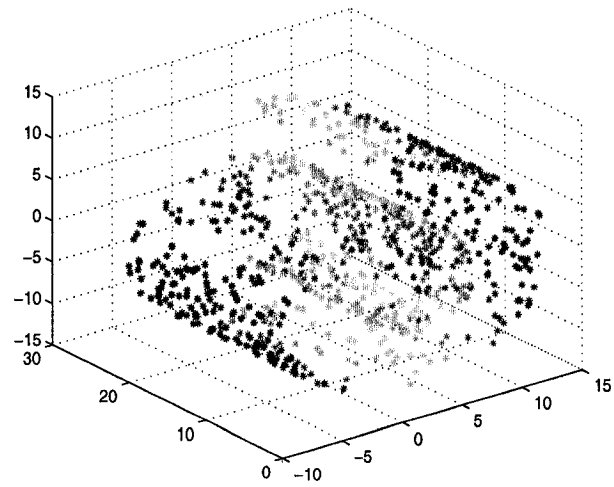
Figure 1.7 shows HLLE applied on the swissrole dataset.

1.1.6 Local Tangent Space Alignment (LTSA)

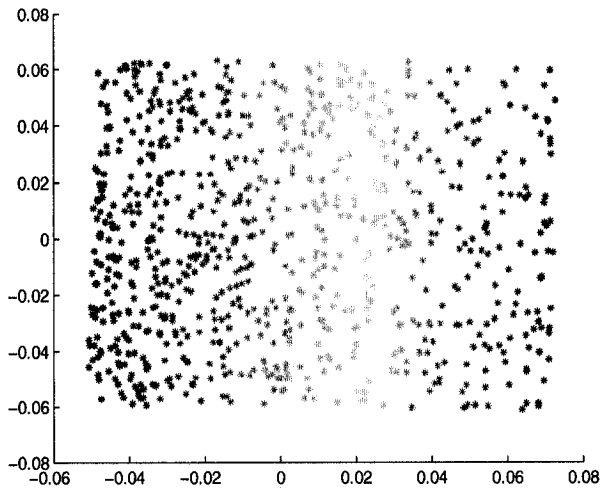
LTSA [13] constructs a principal manifold [13] that goes through the middle of the data points and finds the global coordinate system that characterizes the set of data points in a low-dimensional space. A tangent line space is used to construct this global coordinate system for the manifold. Figure 1.8 shows LTSA applied on the swissrole dataset.

The algorithm is presented as follows:

- The k-nearest neighbors of x_i are calculated, where x_i, $i = 1, \ldots, N$, is a set of N points of m-dimension.
- The d largest unit eigenvectors $(g_1, \ldots, g_d)$ are computed and then set $G_i = [e/\sqrt{k}, g_1, \ldots, g_d]$, where e is a k-dimensional column of all ones.
- The alignment matrix B is constructed by locally summing $B(I_i, I_i) \leftarrow B(I_i, I_i) + I - G_i G_i^T$
- The global coordinates are aligned by computing the $d+1$ smallest eigenvectors of B and the eigenvector matrix $[u_2, \ldots, u_{d+1}]$ is picked up corresponding to the 2^{nd} to $(d+1)^{\text{st}}$ smallest eigenvalues. Set $\Gamma = [u_2, \ldots, u_{d+1}]^T$, where Γ is the output containing the embedded lower dimensional data points.



(a)



(b)

Figure 1.8: LTSA implemented on swissroll dataset. (a) 3D swissroll dataset, (b) The 2D representation result of LTSA applied on the swissroll dataset.

1.2 Contributions

The contributions of this thesis are as follows:

- **Incremental Hessian Locally Linear Embedding:** We introduced an incremental form of HLLE which enables the algorithm to adjust to new arriving points while preserving the significant characteristics of the dataset. The proposed method follows a similar pattern of the incremental LLE and its generalizations [23].

- **Incremental Local Tangent Space Alignment:** An incremental form of LTSA is proposed which allows for newly added points to be adapted while keeping the characteristics of the lower dimensional representation intact. The proposed method follows a similar pattern of the incremental LLE and the above proposed incremental HLLE and their generalizations [24].

- **Incremental Kernel PCA Application:** We applied the IKPCA algorithm on the spectra dataset and studied its behavior. It showed that the algorithm succeeds in trying to keep the accumulation ration within the set threshold and maintain that value.

1.3 Thesis overview

The rest of the thesis is organized as follows:

- In Chapter 2, we describe the basic concepts of the LLE and the HLLE algorithms. We also introduce our proposed method of incremental HLLE and explain the procedures applied. We then demonstrate, through experimental results, the improved performance of our proposed technique in comparison with the incremental LLE, and we provide a concise interpretation of our results with conclusions [23].

- In Chapter 3, we present the LTSA algorithm and introduce our proposed method of incremental LTSA. Illustrating experimental results show the effectiveness of the proposed compared to the previous methods [24].

- In Chapter 4, Kernel PCA algorithm is described along with the Incremental KPCA algorithm. Then we explain the significance of the Spectra dataset and demonstrate, through experimental results, the performance of the IKPCA algorithm when applied to the spectra dataset. We also provide a concise interpretation of the results.

- In the Conclusions Chapter, we summarize the contributions of this thesis, and we propose several future research directions that are directly or indirectly related to the work performed in this thesis.

1.4 Publications

- O. Abdel-Mannan, A. Ben Hamza, and A. Youssef, "Incremental Dimensionality-Reduction Algorithm using Heissien Locally Linear Embedding," *Proc. IEEE International Symposium on Signal Processing and its Applications*, Sharjah, UAE, 2007.

- O. Abdel-Mannan, A. Ben Hamza, and A. Youssef, "Incremental Local Tangent Space Alignment Algorithm," *Proc. IEEE Canadian Conference on Electrical and Computer Engineering*, Vancouver, Canada, 2007.

CHAPTER 2

Incremental Hessian Locally Linear Embedding

In this chapter, we present an incremental version of Hessian Locally Linear Embedding (HLLE) on the basis of incremental Locally Linear Embedding (LLE) generalizations [15], [21], [22]. The main idea behind our algorithm is to produce a lower dimension representation of a high-dimensional manifold such that the significant characteristics of the dataset are preserved while adapting to newly added points arriving to the dataset. The experimental results verify how the new projection of points, along with the additional points, produce a good fit to the original manifold [23].

2.1 Introduction

The need to analyze large amounts of multidimensional data is simplified through the concept of dimensionality reduction. Discovering the compact representations of high-dimensional data is vital in areas related to data analysis and visualization. Therefore, several methods were devised to reduce the dimension of multidimensional data and thus make it simpler to analyze and study. LLE [10], [26] and HLLE [11] are examples of such methods that compute low-dimensional, neighborhood-preserving embeddings of high-dimensional inputs. They are two nearest-neighbor techniques that optimally reduce the high dimensionality by reconstructing data points from a weighted combination of neighbors. This is achieved by finding the lowest eigenvalues and their corresponding eigenvectors during the computation of the lower dimension projections.

However, these algorithms are constructed for datasets in batch mode and do not accommodate newly added points. In other words, if a new data point arrives, the algorithm has to be run entirely again to include the newly added point in the computations, which is not very practical for datasets which are very large in size. To circumvent this problem, an incremental LLE algorithm was recently proposed in [15].

Motivated by the incremental form of LLE, in which a new generalization was adapted and compared to two previously proposed generalizations [21], [22], we propose an incremental form of HLLE similar to the incremental LLE implementation. The core idea behind our proposed algorithm is it to allow for incremental update of the algorithm to adapt to newly

introduced data points.

The rest of this chapter is organized as follows. In the next section, we recall the basic concepts of the LLE algorithm. In Section 2.3, we describe the HLLE algorithm as well as introduce our proposed method of incremental HLLE and explain the procedures applied. In Section 2.4, we demonstrate, through experimental results, the much improved performance of our proposed technique in comparison with the incremental LLE, and we provide a concise interpretation of our results. Finally, some conclusions are included in Section 2.5.

2.2 Problem Formulation

2.2.1 LLE Algorithm

A dimensionality reduction algorithm takes a D-dimensional input dataset $X = x_1, x_2, \ldots, x_N$, of N points $x_i \in \mathbb{R}^D$. The algorithm then produces N corresponding d-dimensional vectors y_i constructing matrix $Y = y_1, y_2, \ldots, y_N$, $y_i \in \mathbb{R}^d$. The LLE algorithm runs to reconstruct each data point as a linear combination of its neighbors using two parameters: k, which is the number of nearest neighbors, and d, which is the number of dimensions into which the data is embedded.

The LLE algorithm consists of three main steps [10]:

- **Step 1:** The neighborhood of X is found by finding the k-nearest neighbors of

each point x_i producing $x_i^1, x_i^2, \ldots, x_i^k$ using Euclidean distances.

- **Step 2:** The weight matrix $W = w_{ij}$, which reconstructs each point x_i from its k-nearest neighbors is computed by minimizing the following function:

$$\varepsilon(W) = \sum_{i=1}^{N} \|x_i - \sum_{j=1}^{k} w_{ij} x_j\|^2 \tag{1}$$

subject to two constraints: $w_{ij} = 0$, if $x_{ij} \notin \{x_i^1, x_i^2, \ldots, x_i^k\}$, and $\sum_{j=1}^{k} w_{ij} = 1$.

- **Step 3:** The embedded vector Y is reconstructed from X by minimizing:

$$\delta(Y) = \sum_{i=1}^{N} \|y_i - \sum_{j=1}^{k} w_{ij} y_j\|^2 \tag{2}$$

under the constraints: $\frac{1}{N} \sum_{i=1}^{N} y_i y_j^T = I$ (normalized unit covariance), and $\sum_{i=1}^{N} y_i = 0$ (translation-invariant embedding). Obtaining a low-dimensional embedding with these constraints corresponds to computing the d+1 eigenvectors associated with the d+1 smallest eigenvalues of a sparse, symmetric cost matrix M:

$$M = (I - W)^T (I - W) \tag{3}$$

2.3 Proposed Method

A variant method to the above explained LLE algorithm is the HLLE algorithm [11]. HLLE needs to solve N separate eigenproblems of $k \times k$ size, as well as a single $N \times N$ sparse

eigenproblem. In fact, the methods used in HLLE bear considerable similarity to the previously proposed LLE. Both algorithms are able to handle large N problems, since smaller neighborhoods are being considered in the primary computations. The process in HLLE is however somewhat different in requiring estimates of second derivatives.

The HLLE algorithm consists of the following steps.

- **Step 1:** The k-nearest neighbors of each data point are found using Euclidean distances. A matrix M^i is formed for each neighborhood N^i.
- **Step 2:** Tangent coordinates are obtained by performing a singular value decomposition of the matrix M^i containing the distances between the neighboring points.
- **Step 3:** A Hessian estimator is developed in quadratic form built by:

$$\mathcal{H}_{ij} = \sum_l \sum_r (H^l_{r,i} H^l_{r,j}) \tag{4}$$

  where H^l is a $\frac{d(d+1)}{2} \times k$ matrix associated with estimating the Hessian over neighborhood N_i. The entries in the Hessian matrix correspond to rows r, and the specific points in the neighborhood correspond to columns i.
- **Step 4:** Eigenanalysis of the Hessian matrix is performed, and the eigenvectors corresponding to the smallest eigenvalues (discarding the vector whose values are zero) are computed producing matrix V. Matrix R is defined as $R = VV^T$. Thus, embedding coordinates are obtained from the matrix $W = V^T R^{-1/2}$.

2.3 Proposed Method

Similar to the Incremental LLE described in [15], the method employed to implement incremental HLLE makes use of two possible linear generalizations [21], [22], as well as the generalization proposed by the incremental LLE method [15]. These are applied to form a relation between the high and low-dimensional points that belong to a manifold neighborhood. Let X and Y be the matrices containing the high-dimensional input and the low-dimensional output, respectively, w_{ij} be the linear weights that reconstruct each point x_i, where $xi \in X$, $i = 1, 2, \ldots, N$, from its k-nearest neighbors and M be the cost matrix produced using eigenavalue computations, the two generalizations and the incremental form of HLLE are described as follows.

Linear generalization *I*: The first generalization exploits a transformation matrix derived from the neighborhood of the data points and then applied to find the new projection. Let X^{N+1} and Y^{N+1} be the matrices containing the k nearest neighbors of x_{N+1} and y_{N+1} respectively. Thus, the equality assumption $Y^{N+1} = ZX^{N+1}$ is approximate, where Z is an unknown transformation matrix of size $d \times D$. Consequently, the projection of the newly added point can be found by multiplying the obtained transformation:

$$y_{N+1} = Zx_{N+1} \tag{5}$$

Linear generalization *II*: In the second generalization, the newly projected point can be found by first finding the k nearest neighbors of x_{N+1} and the linear weights w_{N+1} that reconstruct x_{N+1} from its neighbors using Equation 1 with the constraints mentioned. Then

the new y_{N+1} is found by:

$$y_{N+1} = \sum_{j=1}^{N} w_{N+1} y_j \tag{6}$$

Incremental HLLE: When a new data point is added to the manifold, the aim is to accommodate this new point and accordingly produce the linear weights matrix and compute the cost matrix M_{new} in the same manner as if it would be computed if HLLE was run on the whole dataset, including the new point. This can be achieved by the following steps.

- **Step 1.** The new neighborhood of the new dataset is found, i.e. the k nearest neighbors to the newly added point, as well as the neighborhood of other data points which have, at this time, developed a new neighborhood as a result of the point addition.
- **Step 2.** With the new neighborhood obtained, the modified neighborhoods result in new distances between points. A new weight matrix is computed only for those points whose distances have been changed, by solving Equation (1). The weights in this new weight matrix are used to update the old weight matrix for those changed points.
- **Step 3.** The updated weight matrix is then used to calculate the new cost matrix M_{new}, which will have a size of $(N+1) \times (N+1)$.

2.4 Experimental Results

This section presents simulation results where the incremental LLE, generalization *I*, generalization *II*, and the proposed incremental HLLE algorithms are applied to three types of datasets: the swissroll dataset, the wine dataset [27], and some 3D models. The procedures applied in incremental LLE were implemented to form a basis of reference and comparison to our proposed incremental HLLE algorithm.

2.4.1 Swissroll dataset

The swissroll dataset consists of 1,400 points and is 3-dimensional. The dataset is depicted in Figure 2.1.

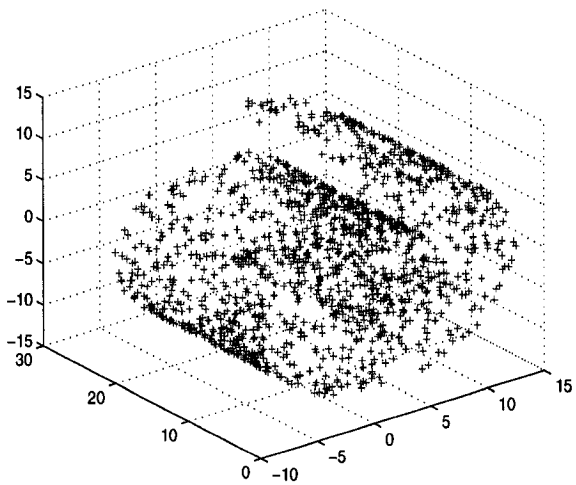


Figure 2.1: Swissroll 3D dataset

2.4 Experimental Results

First, regular LLE and HLLE, with k=15, are applied to the manifold projecting the dataset into a two-dimensional space. Five additional points are introduced to the dataset and the linear generalizations along with incremental LLE and incremental HLLE are then applied to the manifold.

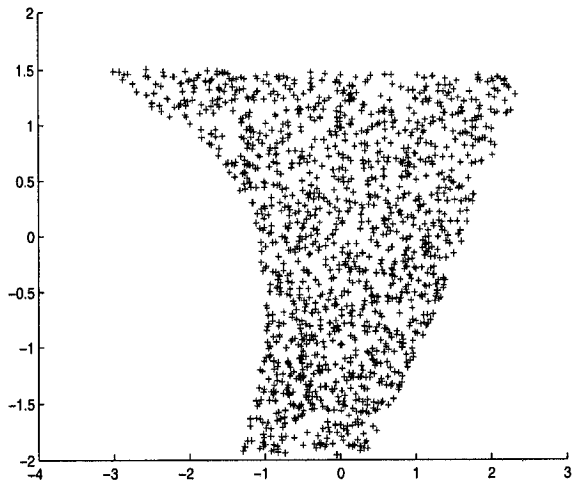


Figure 2.2: 2D representation of swissroll dataset using regular LLE

The results shown in Figure 2.3 through Figure 2.5 illustrate the projection of the 3-D manifold into 2-D using generalization *I*, generalization *II*, and incremental LLE. The circles in the figures represent the additional points that were incremented using the algorithm. These results show a slightly different outcome from what was produced in the original incremental LLE implementation in [15]. This is due to the eigenproblem that should be solved by minimizing the following function to obtain the new coordinates of the projected points:

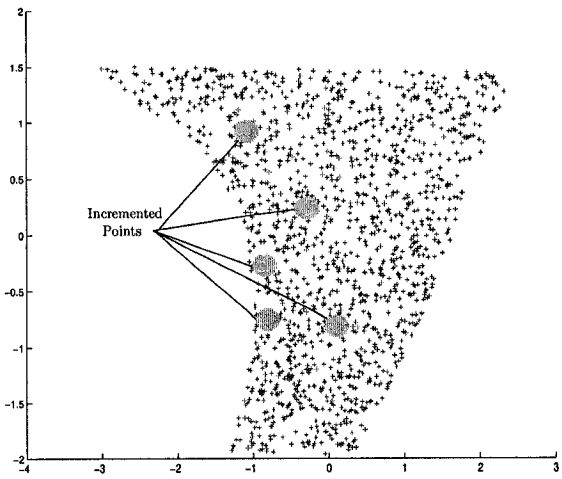


Figure 2.3: LLE linear generalization *I*

$$\min_{Y_{new}}(Y_{new}M_{new}Y_{new}^T - diag\{\lambda_1, \lambda_2, \ldots, \lambda_d\}) \tag{7}$$

However, despite the differences, the outcome still serves the purpose of projecting the high-dimensional dataset into a lower dimension while adapting to the newly added points.

Following the same concepts, the proposed algorithm is performed on the swissroll dataset. Five points are also added to the 1,400 original points and generalization *I*, generalization *II*, and incremental HLLE are applied to the dataset. The results in Figure 2.7 through Figure 2.9 show the projection using the linear generalizations and incremental HLLE with the five points added to the 3-D manifold producing a 2-D output.

Figure 2.7 through Figure 2.9 show that incremental HLLE preserves the projection dispersion of the lower dimension representation of the manifold, keeping the distribution of

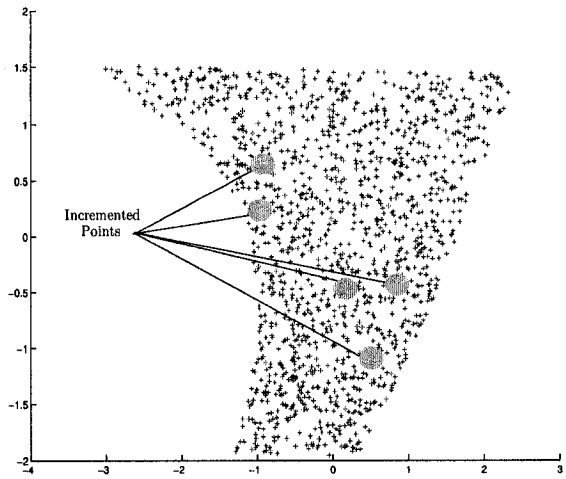


Figure 2.4: LLE linear generalization *II*

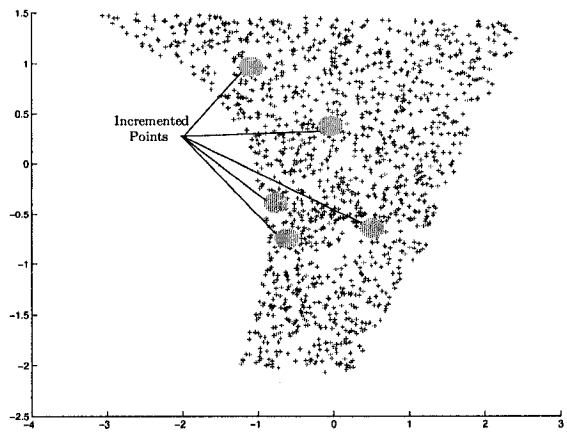


Figure 2.5: Incremental LLE

points intact, while accommodating the newly added points in it. The circles in the figures represent the additional points that were incremented using the algorithm.

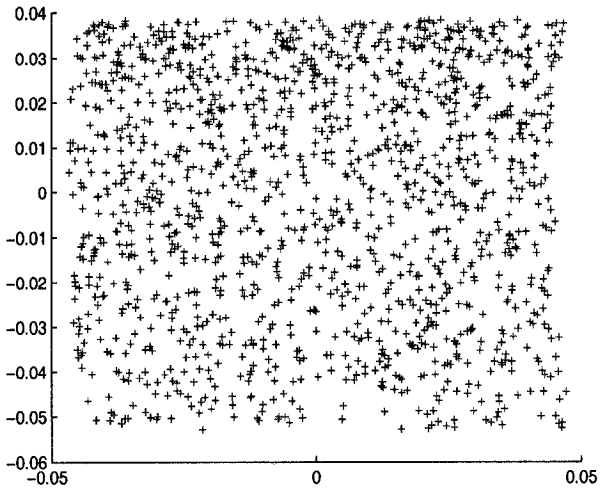


Figure 2.6: 2D representation of swissroll dataset using regular HLLE

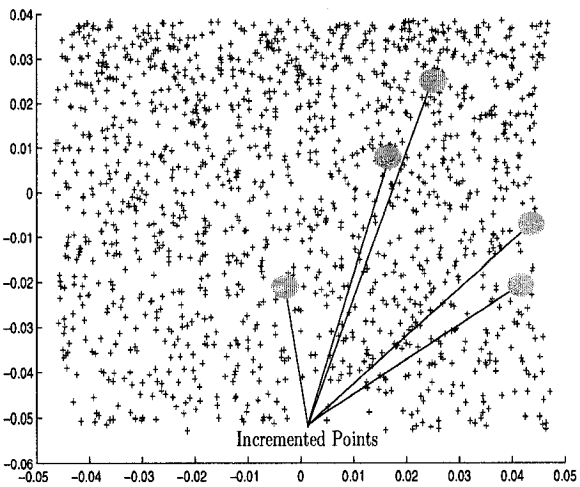


Figure 2.7: HLLE linear generalization I

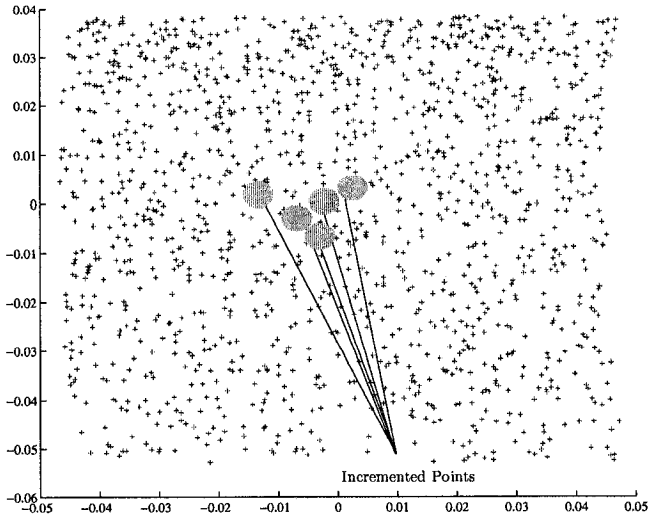


Figure 2.8: HLLE linear generalization *II*

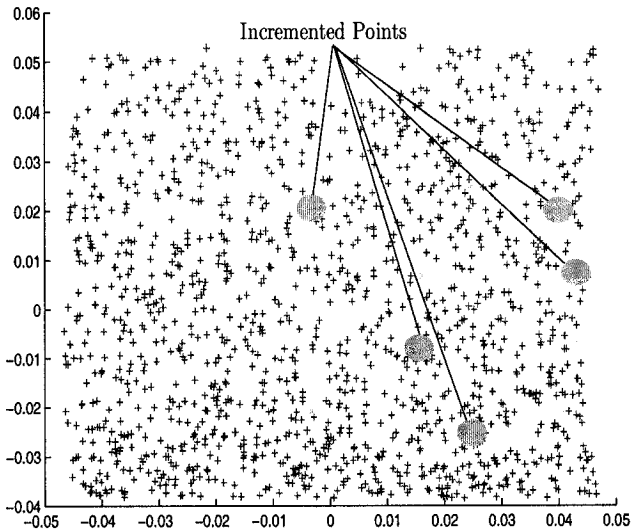


Figure 2.9: Incremental HLLE

2.4.2 Wine dataset

In another experiment, we tested our proposed algorithm on the wine dataset [27]. The wine dataset contains the results of a chemical analysis of three types of wine grown in a specific area. The data is represented in the 178 samples, with the results of 13 chemical analyses recorded for each sample, i.e. having a dimensionality of 13.

In this experiment, regular HLLE (k=15) is applied on the first 118 points. The remaining 60 points are incremented in 15 iterations, four points at a time.

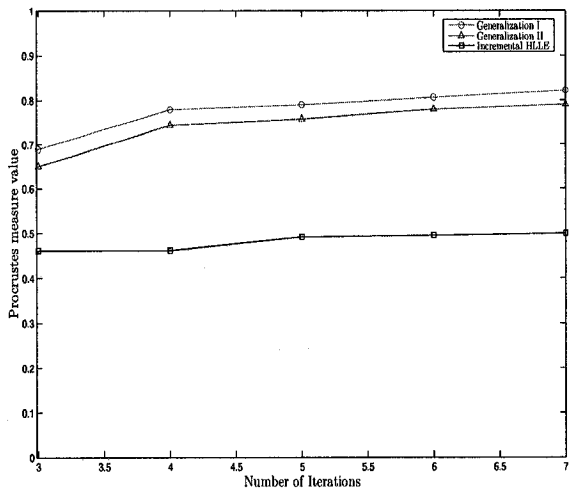


Figure 2.10: HLLE procrustes measure for wine dataset

Procrustes measure [25] is used to study the behavior of the algorithms as the number of data points increases. The Procrustes measure determines how a linear transformation of the points in the lower dimension projection conforms to the high-dimensional data. The lower the value the better these two representations conform. Figure 2.10 shows the result

of the iterations performed with incremental HLLE. From the plot in Figure 2.10, it can be seen that incremental HLLE maintains a lower value of the Procrustes measure compared to the other generalizations, which demonstrates the conformity of the low-dimensional data representation to the high-dimensional one.

2.4.3 3D datasets

Finally, the proposed algorithm was applied on some 3D models with different number of points. In all the following experiments, k is set to 8, and the dimension was reduced from 3-dimension to 2-dimension. Five points were incremented to the dataset and are represented by the circles in the figures.

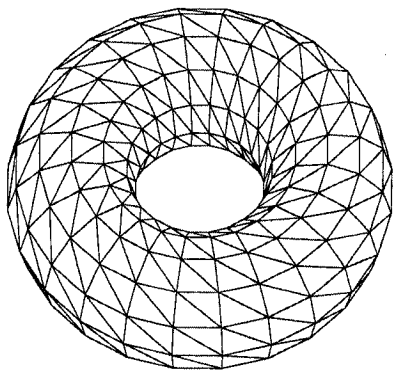


Figure 2.11: 3D model of a Torus

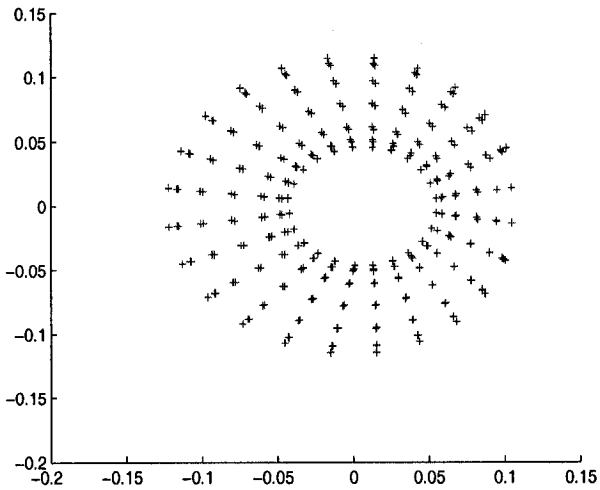


Figure 2.12: Regular HLLE applied on the torus 3D model

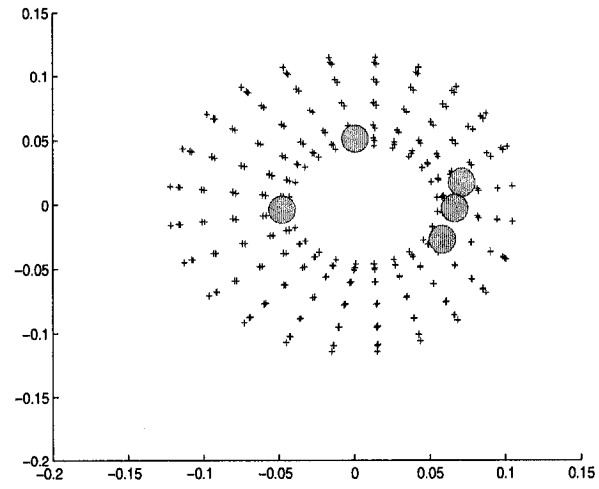


Figure 2.13: HLLE generalization I applied on the torus 3D model

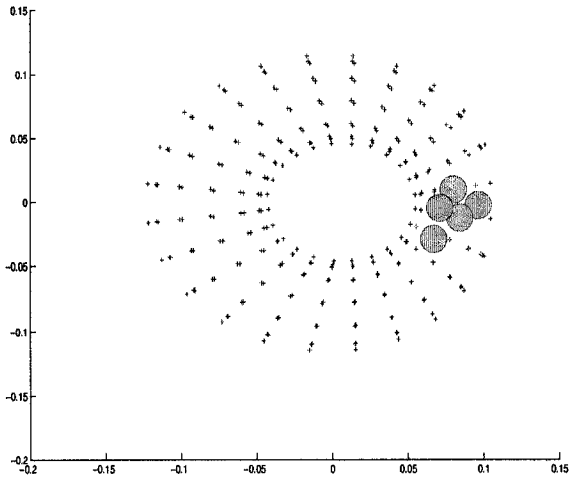


Figure 2.14: HLLE generalization *II* applied on the torus 3D model

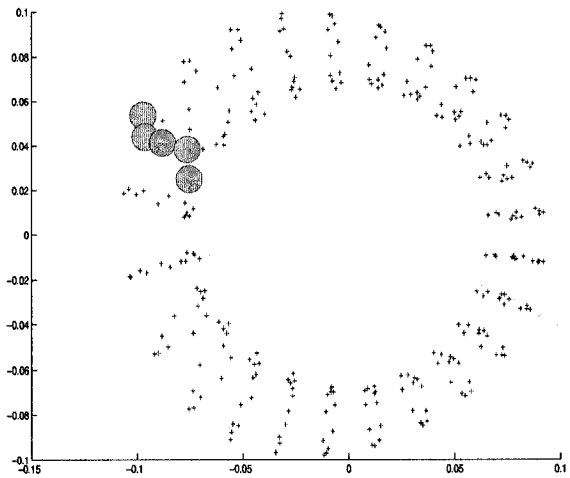


Figure 2.15: Incremental HLLE applied on the torus 3D model

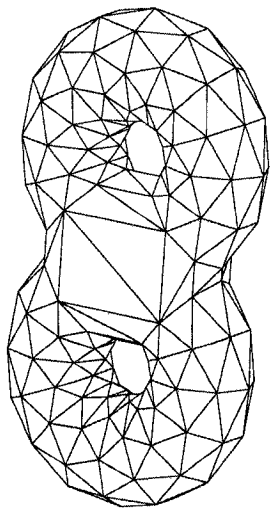


Figure 2.16: 3D model of a Double Torus

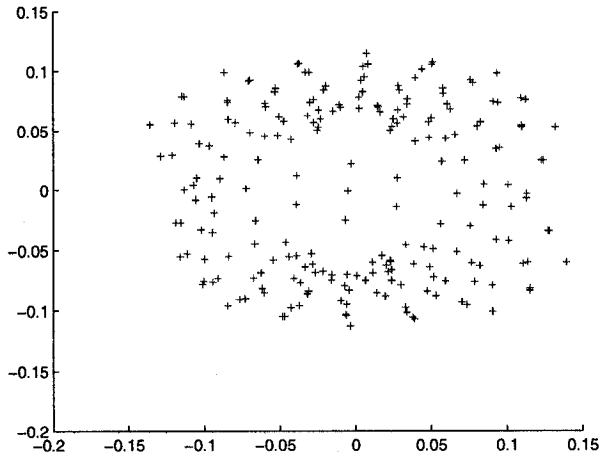


Figure 2.17: Regular HLLE applied on the double torus 3D model

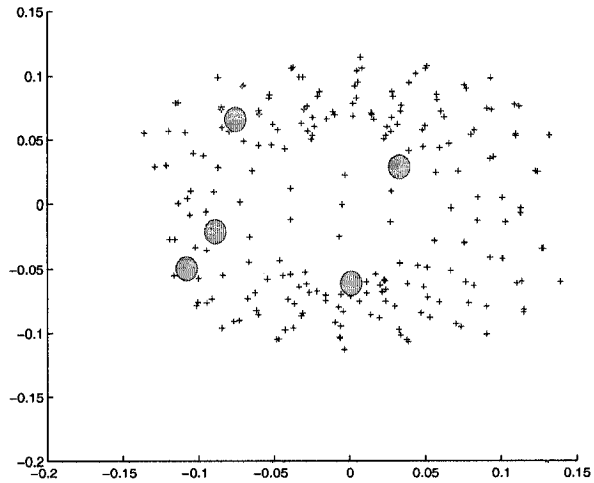


Figure 2.18: HLLE generalization *I* applied on the double torus 3D model

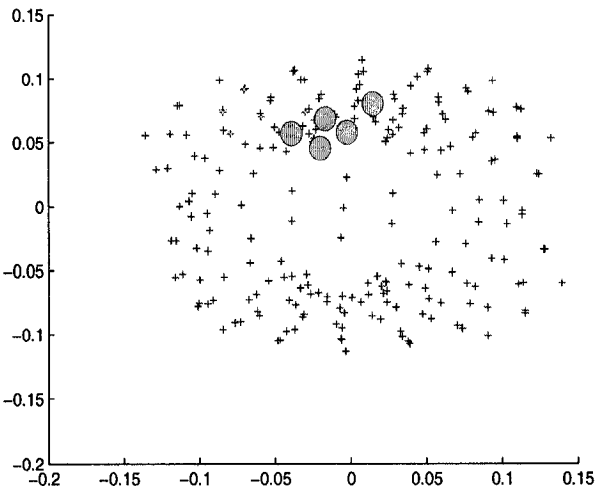


Figure 2.19: HLLE generalization *II* applied on the double torus 3D model

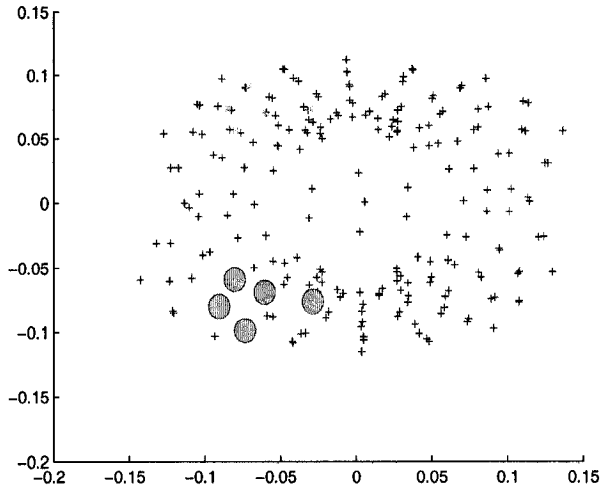


Figure 2.20: Incremental HLLE applied on the double torus 3D model

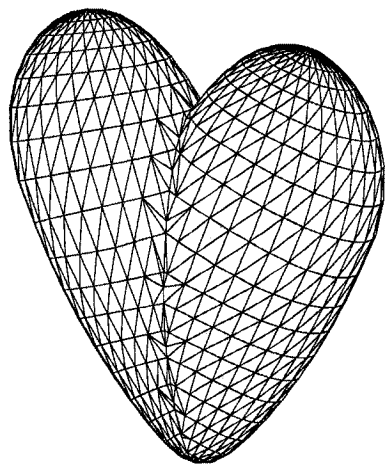


Figure 2.21: 3D model of a Heart

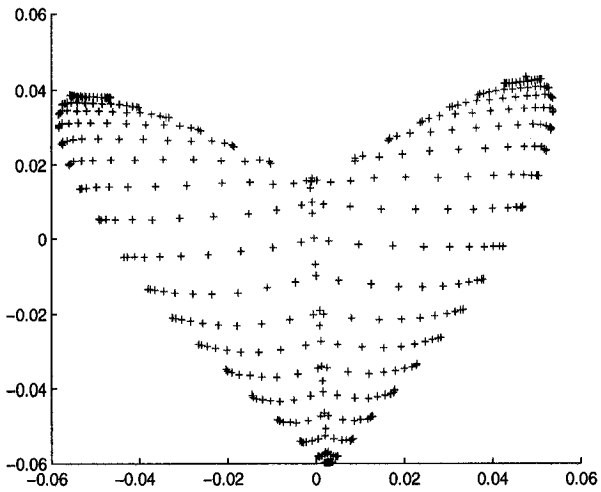


Figure 2.22: Regular HLLE applied on the heart 3D model

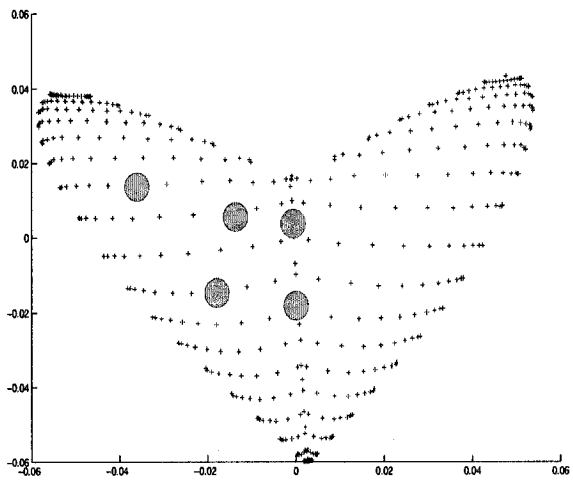


Figure 2.23: HLLE generalization I applied on the heart 3D model

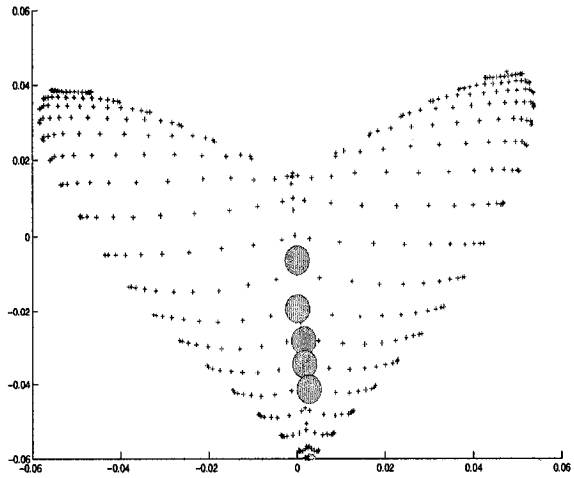


Figure 2.24: HLLE generalization *II* applied on the heart 3D model

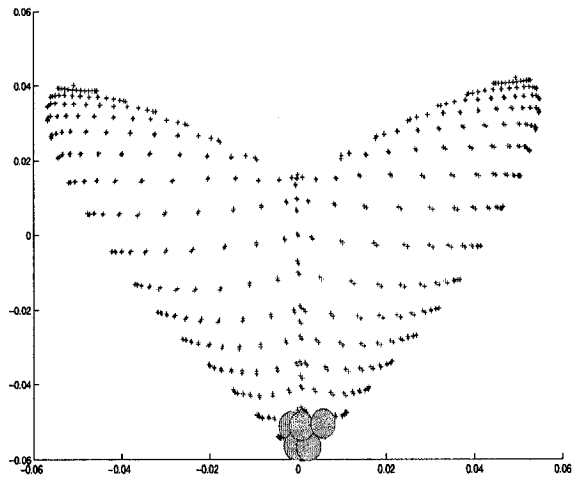


Figure 2.25: Incremental HLLE applied on the heart 3D model

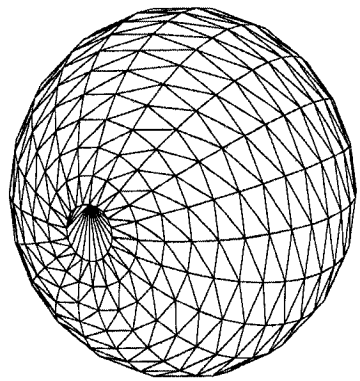


Figure 2.26: 3D model of a Sphere

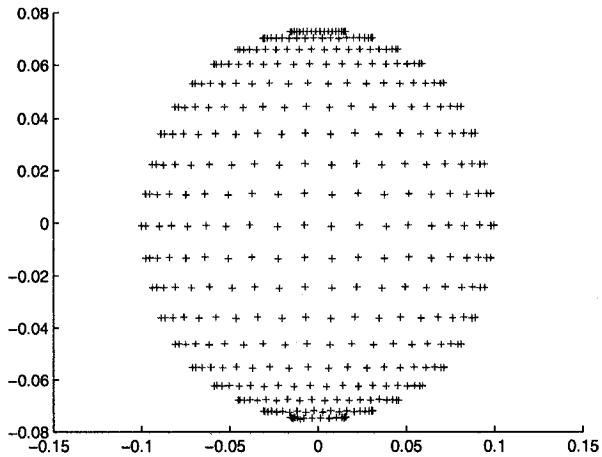


Figure 2.27: Regular HLLE applied on the sphere 3D model

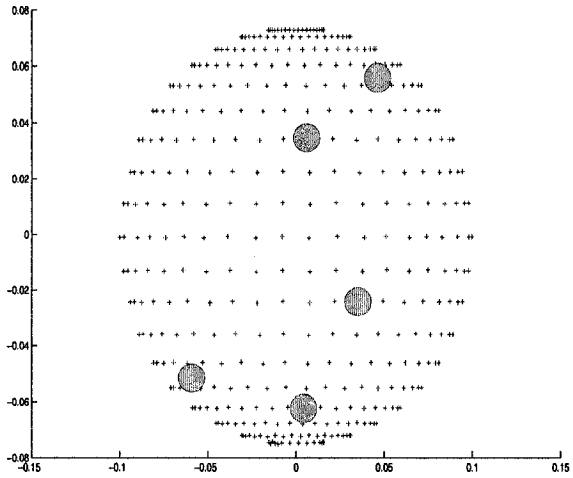


Figure 2.28: HLLE generalization *I* applied on the sphere 3D model

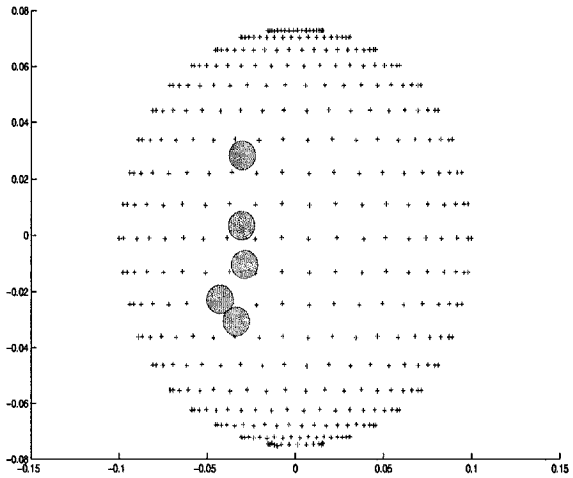


Figure 2.29: HLLE generalization *II* applied on the sphere 3D model

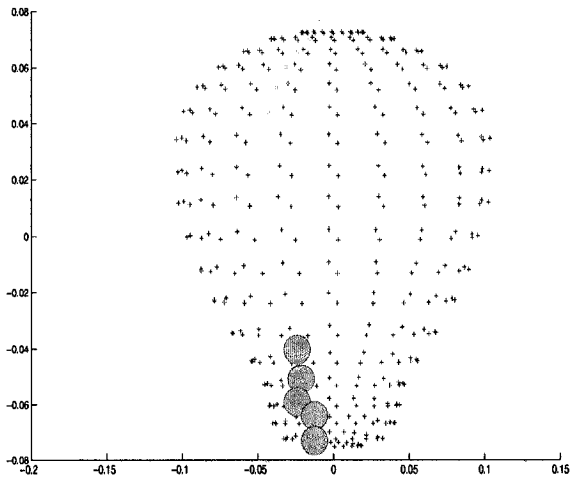


Figure 2.30: Incremental HLLE applied on the sphere 3D model

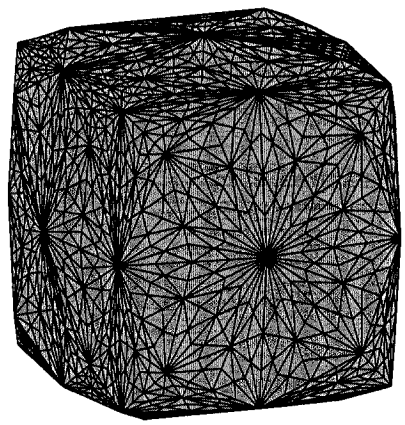


Figure 2.31: 3D model of a Cube

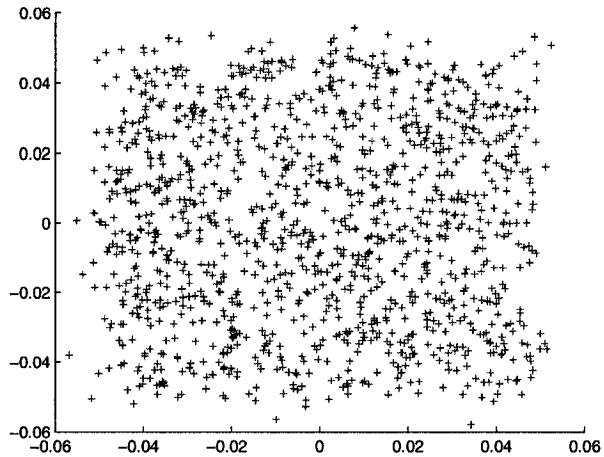


Figure 2.32: Regular HLLE applied on the cube 3D model

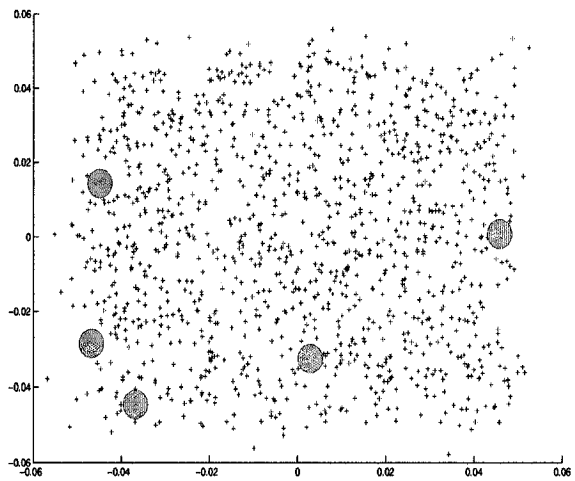


Figure 2.33: HLLE generalization I applied on the cube 3D model

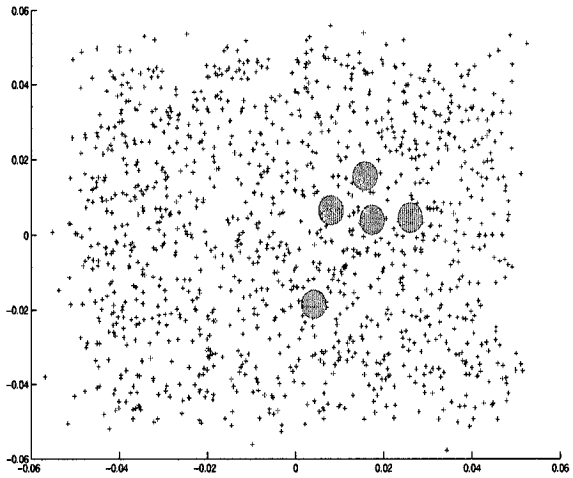


Figure 2.34: HLLE generalization *II* applied on the cube 3D model

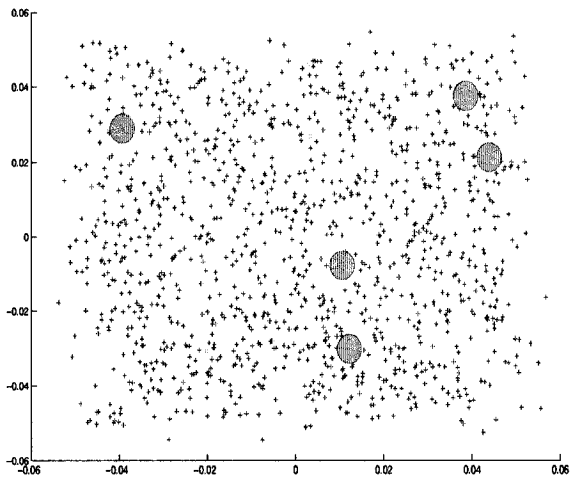


Figure 2.35: Incremental HLLE applied on the cube 3D model

2.5 Conclusion

In this chapter, we proposed an incremental form of HLLE which enables the algorithm to adjust to new arriving points while preserving the significant characteristics of the dataset. The proposed method follows a similar pattern of the incremental LLE and its generalizations.

CHAPTER 3

Incremental Local Tangent Space Alignment

In this chapter we introduce another incremental dimensionality reduction implementation applied on the Local Tangent Space Alignment Algorithm (LTSA) [13] following the concepts implemented in Chapter 2. This incremental algorithm constructs low-dimensional representation from sample data points in high-dimensional spaces, while preserving the important features of the dataset and also adapt to newly added points [24].

3.1 Introduction

As mentioned in the previous chapter, LLE and HLLE produce a low-dimensional representation from a high-dimensional dataset. They, however, tend to fail to detect nonlinear structures in the data points. LTSA succeeds in constructing a nonlinear low-dimensional

representation from high-dimensional data points. LTSA is an unsupervised method for nonlinear dimension reduction which represents the local geometry of the manifold using tangent spaces aligned to give the internal global coordinates of the data points with respect to the underlying manifold by means of a partial eigen-decomposition of the neighborhood connection matrix.

However, similar to the previous algorithms LLE and HLLE, the LTSA algorithm is unsuitable for sequentially added data. In other words, when a new point is added, the algorithm needs to be rerun entirely with the original data augmented by the new samples. Therefore, we implemented an incremental form of LTSA to avoid such a limitation. This incremental algorithm enables the newly added points to be added as they arrive and produce the lower dimensional representation as if the algorithm is run on the whole dataset.

This chapter is organized as follows. In Section 3.2 we introduce the basic concept of the LTSA algorithm. Then in Section 3.3, we introduce our proposed method of incremental LTSA and explain the procedures applied. In Section 3.4, we provide some experimental results and illustrate the performance of our proposed technique in comparison with the incremental LLE and incremental HLLE. Finally, we conclude in Section 3.5.

3.2 Problem Formulation

3.2.1 Local Tangent Space Alignment Algorithm (LTSA)

The LTSA algorithm takes an m-dimensional input dataset $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$, of N points $\boldsymbol{x}_i \in \mathbb{R}^m$. The algorithm then produces N corresponding d-dimensional vectors, $\boldsymbol{\tau}_i$, where $d < m$, constructing matrix $\Gamma = \{\boldsymbol{\tau}_1, \boldsymbol{\tau}_2, \ldots, \boldsymbol{\tau}_N\}$, $\boldsymbol{\tau}_i \in \mathbb{R}^d$, for the manifold constructed from k local nearest neighbors.

The LTSA algorithm consists of four main steps [13]:

- **Step 1:** Find the k nearest neighbors $\boldsymbol{x}_{ij}$ of $\boldsymbol{x}_i$ for $j = 1, \ldots, k$ and set

$$X_i = \{\boldsymbol{x}_{ij}, \ldots, \boldsymbol{x}_{ik}\}. \tag{1}$$

- **Step 2:** Extract local information by computing the d largest unit eigenvectors $g_1, \ldots, g_d$ of the correlation matrix $(X_i - \bar{\boldsymbol{x}}_i e^T)^T (X_i - \bar{\boldsymbol{x}}_i e^T)$ and set

$$G_i = \{e/\sqrt{k}, g_1, \ldots, g_d\}, \tag{2}$$

where $\bar{\boldsymbol{x}}_i = \frac{1}{k} \sum_j \boldsymbol{x}_{ij}$, e is a k-dimensional column of all ones.

- **Step 3:** Construct the alignment matrix B, by locally summing the following with initial $B = 0$:

$$B(I_i, I_i) \leftarrow B(I_i, I_i) + I - G_i G_i^T, (i = 1, \ldots, N), \tag{3}$$

where I_i represents the set of indices for the k nearest neighbors of $\boldsymbol{x}_i$. I is an $N \times N$ identity matrix.

- **Step 4:** Align global coordinates by computing the $d+1$ smallest eigenvectors of B, pick the eigenvector matrix $[u_2, \ldots, u_{d+1}]$ and set the global coordinates $\Gamma = [u_2, \ldots, u_{d+1}]^T$ corresponding to the 2^{nd} to $(d+1)^{\text{st}}$ smallest eigenvalues.

3.3 Proposed Method

Similar to the incremental LLE [15] and incremental HLLE [23], the method employed to implement incremental LTSA makes use of two possible linear generalizations [22], [27], as well as the generalization proposed in incremental LLE and HLLE methods. These were applied to form a relation between the high and low-dimensional points that belong to a manifold neighborhood.

Let X and Γ be the matrices containing the high-dimensional input and the low-dimensional output, respectively, and let $B = \{B_{ij}\}$ be the alignment matrix elements reconstructed for each point neighborhood of k-nearest neighbors. The two generalizations and the incremental form of LTSA are described as follows.

Linear Generalization *I*: The first generalization exploits a transformation matrix derived from the neighborhood of the data points and then applied to find the new projection.

3.3 Proposed Method

Let X^{N+1} and Γ^{N+1} be the matrices containing the k nearest neighbors of $\boldsymbol{x}_{N+1}$ and $\boldsymbol{\tau}_{N+1}$ respectively. Thus, the equality assumption $\Gamma^{N+1} = ZX^{N+1}$ is approximate, where Z is an unknown transformation matrix of size $d \times m$. Consequently, the projection of the newly added point can be found by multiplying the obtained transformation:

$$\boldsymbol{\tau}_{N+1} = Z\boldsymbol{x}_{N+1} \tag{4}$$

Linear Generalization *II*: In the second generalization, the newly projected point can be found by first finding the k nearest neighbors of $\boldsymbol{x}_{N+1}$ and computing the alignment matrix B using Equation 3. Then the new $\boldsymbol{\tau}_{N+1}$ is found by:

$$\boldsymbol{\tau}_{N+1} = \sum_{j=1}^{N} B_{N+1j}\boldsymbol{\tau}_j \tag{5}$$

Incremental LTSA: When a new data point is added to the manifold, the aim is to accommodate this new point and accordingly produce the alignment matrix and align the global coordinates in the same manner as if it would be computed if LTSA is run on the whole dataset, including the new point. This can be achieved by the following steps:

Step 1. The new neighborhood of the new dataset is found. The other data points of the dataset have, at this time, developed a new neighborhood as a result of the point addition. So, the k nearest neighbors to those points as well as to the newly added point is found. .

Step 2. With the new neighborhood obtained, the modified neighborhoods result in new distances between points. A new alignment matrix is computed only for those points whose distances have been changed, by solving Equation 3. The results from this new alignment

matrix are used to update the old alignment matrix for those changed points.

Step 3. The updated alignment matrix is then used to align the global coordinates and produce the new low-dimensional points.

3.4 Experimental Results

This section presents experimental results in which the proposed incremental LTSA algorithm, along with generalization *I* and generalization *II*, are applied on different datasets. The datasets used in this experiment are the same that were used for the previously proposed algorithm, which are the swissroll, the wine dataset [27], and some 3D models.

3.4.1 Swissroll dataset

The same swissroll dataset used in Chapter 2 experiment is used here. It consists of 1,400 points and has a dimensionality of three. Regular LTSA, with $k = 15$, is first applied to the manifold projecting the dataset into a two-dimensional space, then five additional points are added to the dataset to apply the linear generalizations along with incremental LTSA.

The results in Figure 3.1 through Figure 3.3 show the projection using the linear generalizations and incremental LTSA with the five points added to the 3-D manifold producing a 2-D output. The figures show that incremental LTSA preserves the projection dispersion of the lower dimension representation of the manifold while accommodating the newly added

points in it. The circles in the figures represent the additional points that were incremented using the algorithm.

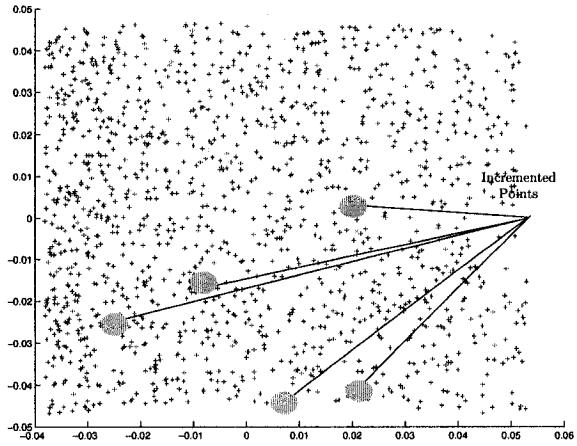


Figure 3.1: LTSA Generalization *I* applied on swissroll dataset

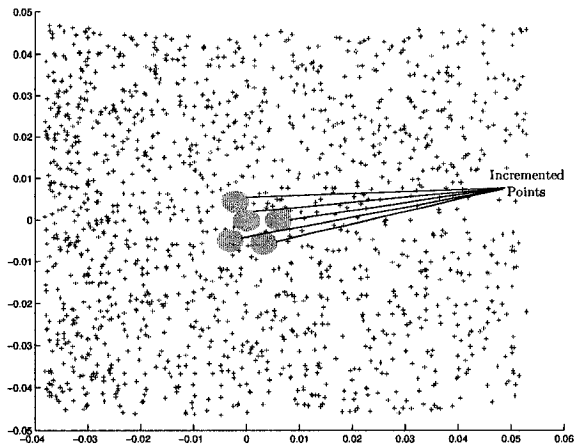


Figure 3.2: LTSA Generalization *II* applied on swissroll dataset

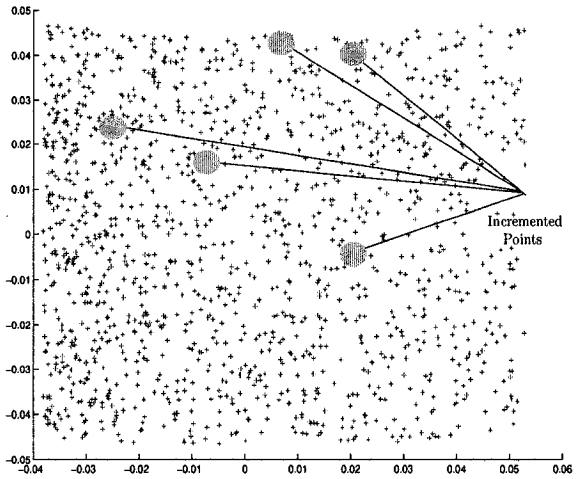


Figure 3.3: Incremental LTSA applied on swissroll dataset

It is worth noting that the time required to run the incremental LTSA is shorter than the time required to run incremental LLE and incremental HLLE. Also, the distribution of points in the low-dimensional space is affected by the number of k-nearest neighbors selected.

3.4.2 Wine dataset

Similarly, we tested our proposed algorithm on the wine dataset [27]. This dataset contains 178 data points and has a dimensionality of 13. In this experiment, regular LTSA ($k = 15$) is applied on the first 118 points. The remaining 60 points are incremented in 15 iterations, four points at a time. Procrustes measure is also used to study the behavior of the algorithms as the number of data points increases.

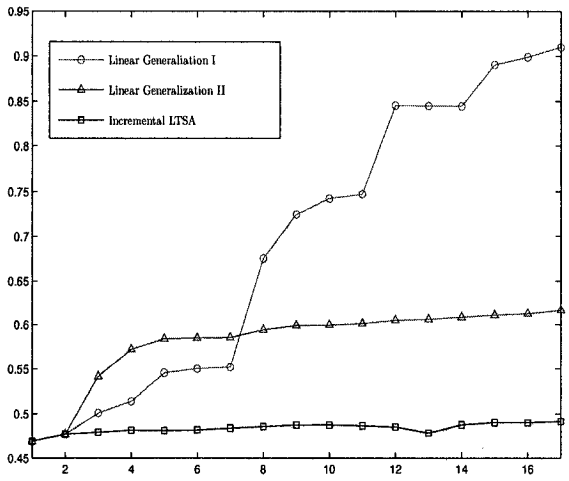


Figure 3.4: LTSA procrustes measure for wine dataset

The procrustes measure determines how a linear transformation of the points in the lower dimension projection conforms to the high-dimensional data. The lower the value the better these two representations conform. Figure 3.4 shows the result of the iterations performed with incremental LTSA. It is evident from the plot in Figure 3.4 that incremental LTSA maintains a lower value of the procrustes measure compared to the other generalizations, which demonstrates the conformity of the low-dimensional data representation to the high-dimensional one.

3.4.3 3D datasets

Similarly, we applied the algorithm on the same 3D models used in Chapter 2, with k set to 8 and the dimension was reduced from 3-dimension to 2-dimension.

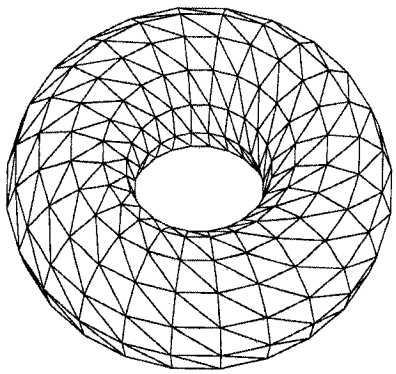


Figure 3.5: 3D model of a Torus

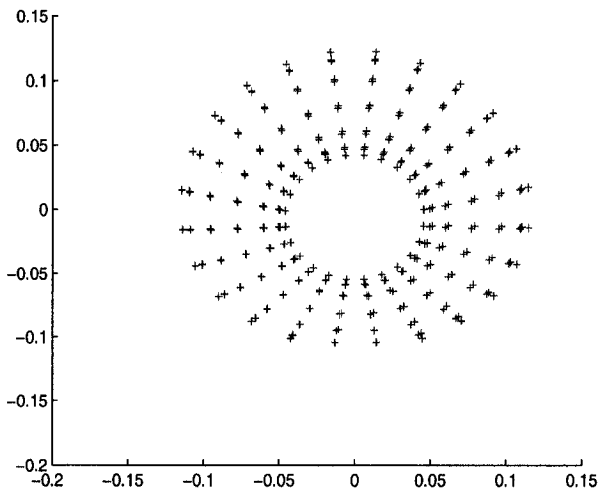


Figure 3.6: Regular LTSA applied on the torus 3D model

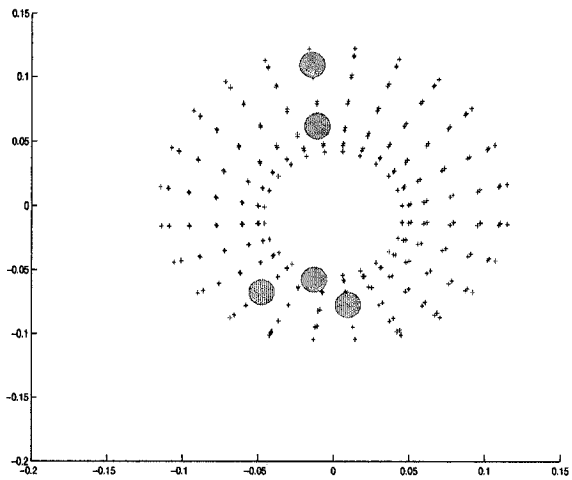


Figure 3.7: LTSA generalization *I* applied on the torus 3D model

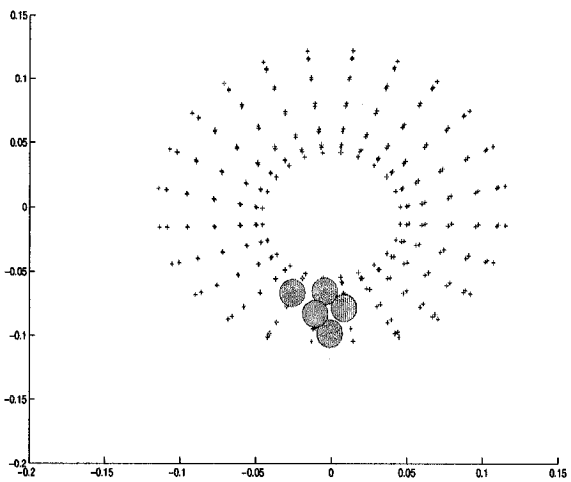


Figure 3.8: LTSA generalization *II* applied on the torus 3D model

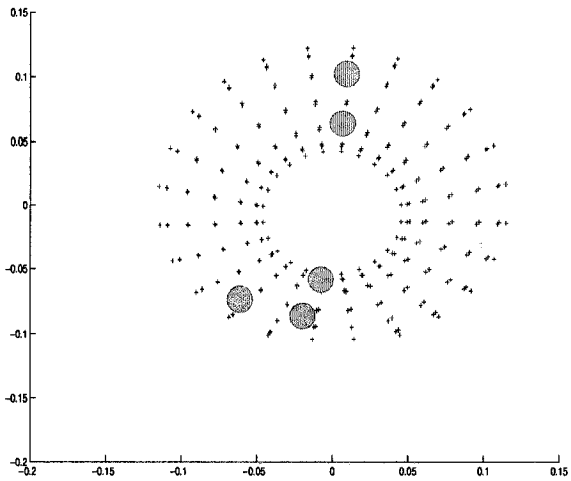


Figure 3.9: Incremental LTSA applied on the torus 3D model

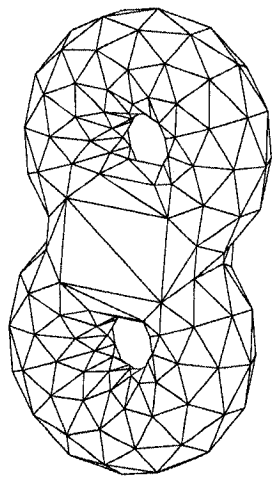


Figure 3.10: 3D model of a Double Torus

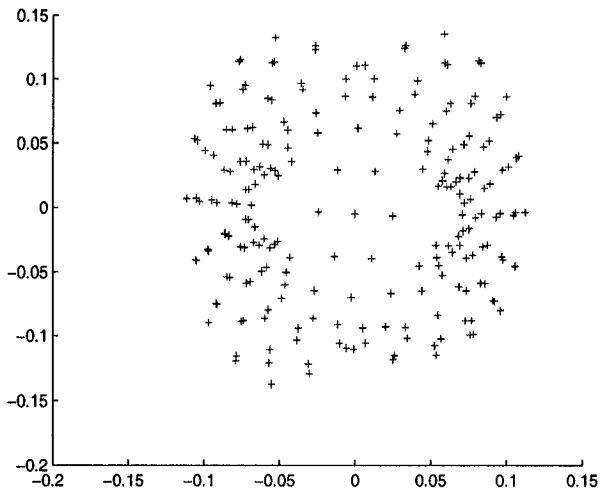


Figure 3.11: Regular LTSA applied on the double torus 3D model

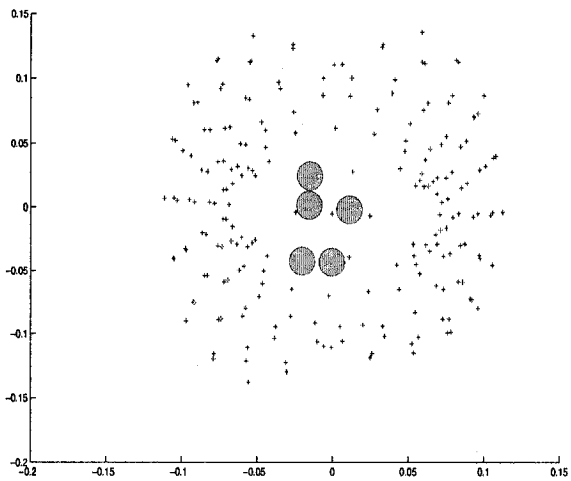


Figure 3.12: LTSA generalization I applied on the double torus 3D model

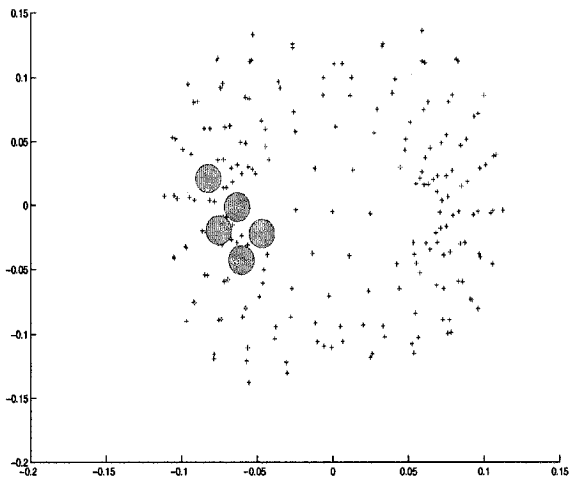


Figure 3.13: LTSA generalization *II* applied on the double torus 3D model

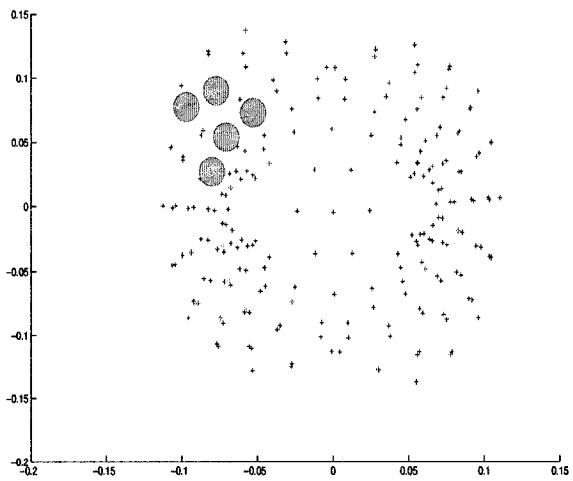


Figure 3.14: Incremental LTSA applied on the double torus 3D model

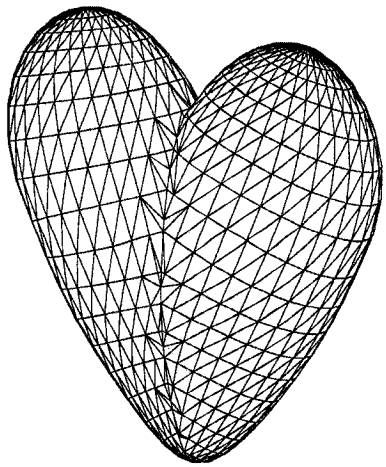


Figure 3.15: 3D model of a Heart

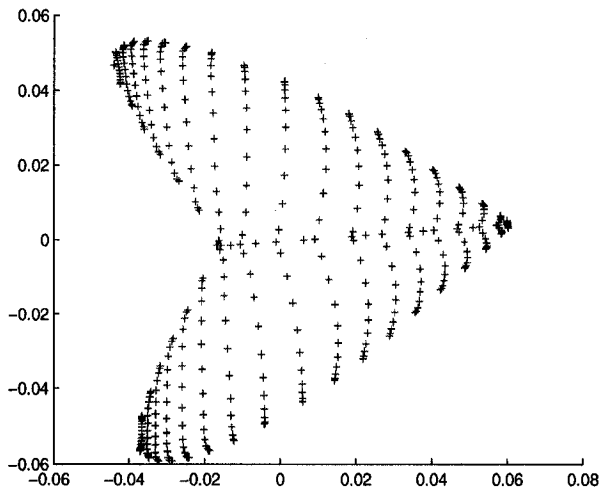


Figure 3.16: Regular LTSA applied on the heart 3D model

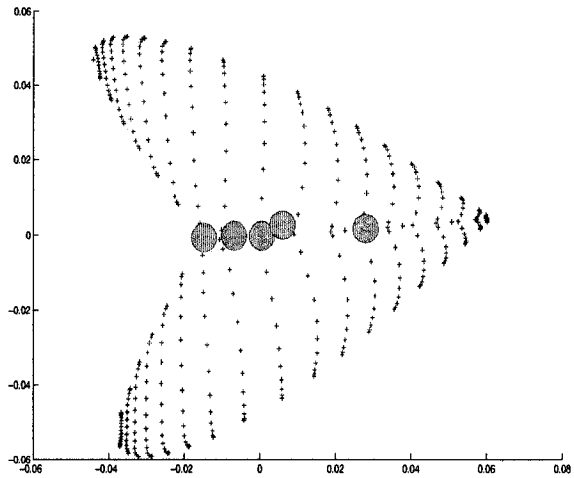


Figure 3.17: LTSA generalization *I* applied on the heart 3D model

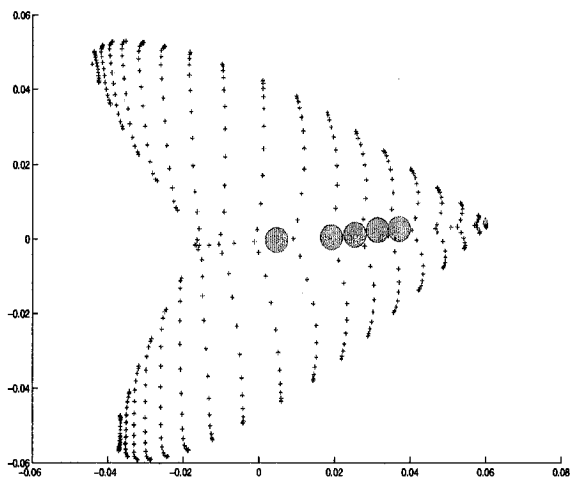


Figure 3.18: LTSA generalization *II* applied on the heart 3D model

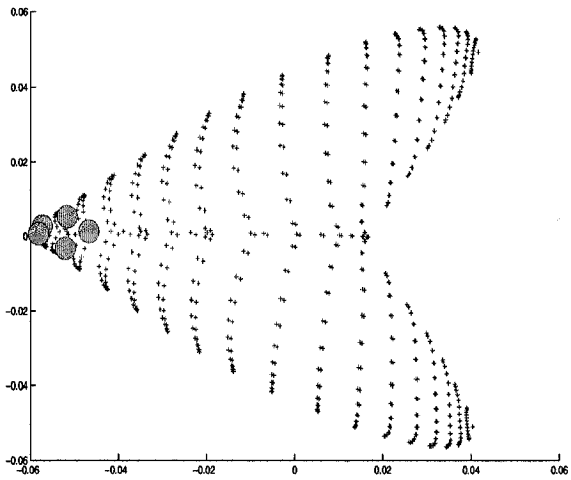


Figure 3.19: Incremental LTSA applied on the heart 3D model

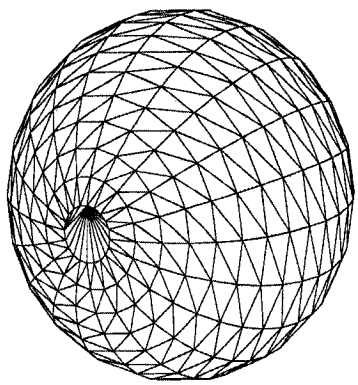


Figure 3.20: 3D model of a Sphere

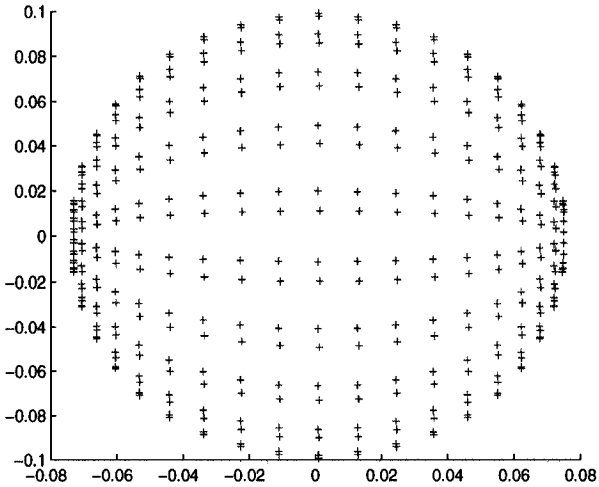


Figure 3.21: Regular LTSA applied on the sphere 3D model

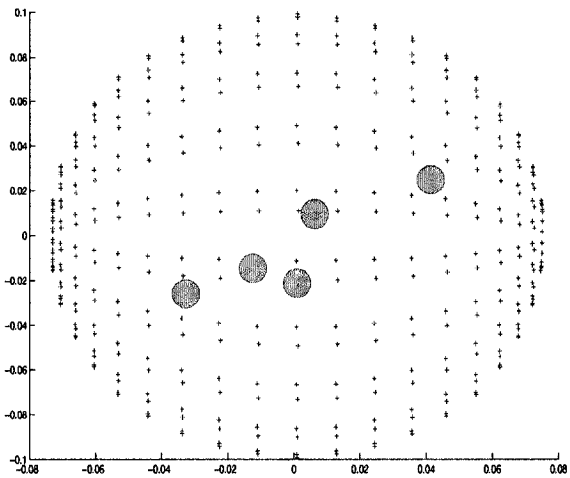


Figure 3.22: LTSA generalization I applied on the sphere 3D model

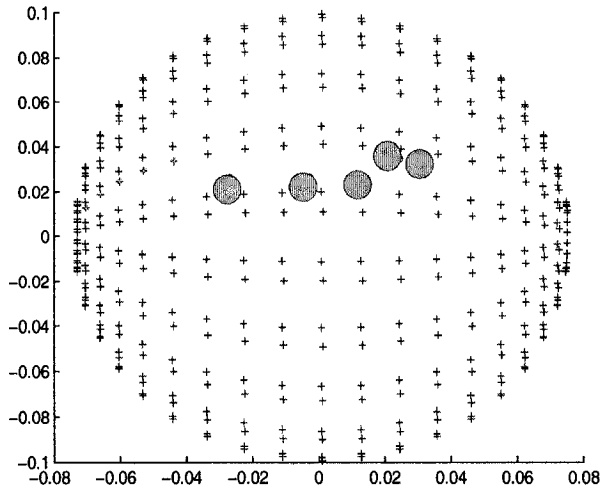


Figure 3.23: LTSA generalization *II* applied on the sphere 3D model

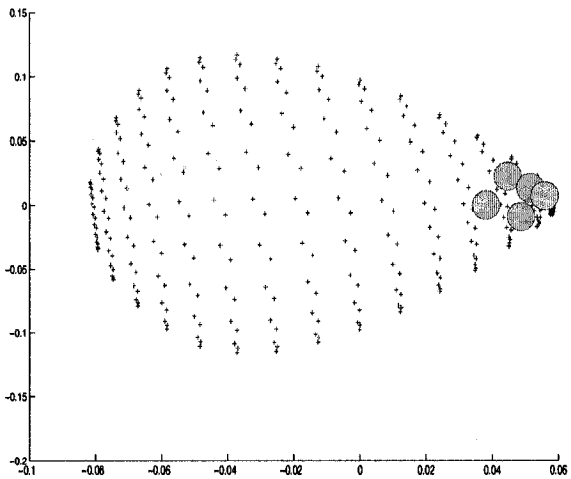


Figure 3.24: Incremental LTSA applied on the sphere 3D model

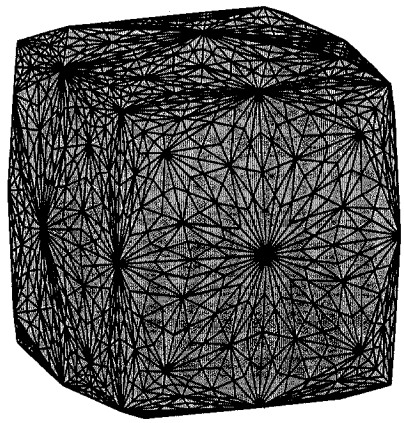


Figure 3.25: 3D model of a Cube

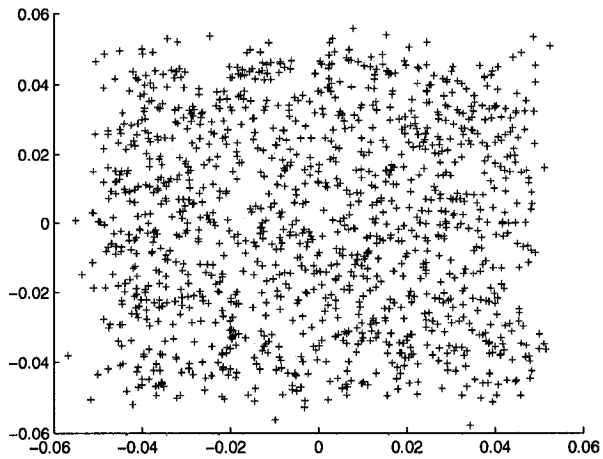


Figure 3.26: Regular LTSA applied on the cube 3D model

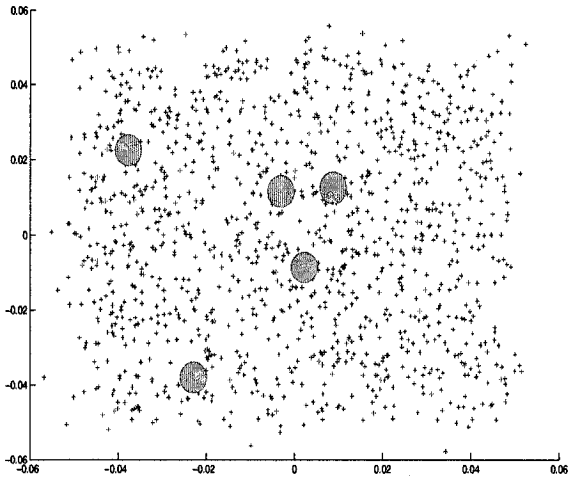


Figure 3.27: LTSA generalization *I* applied on the cube 3D model

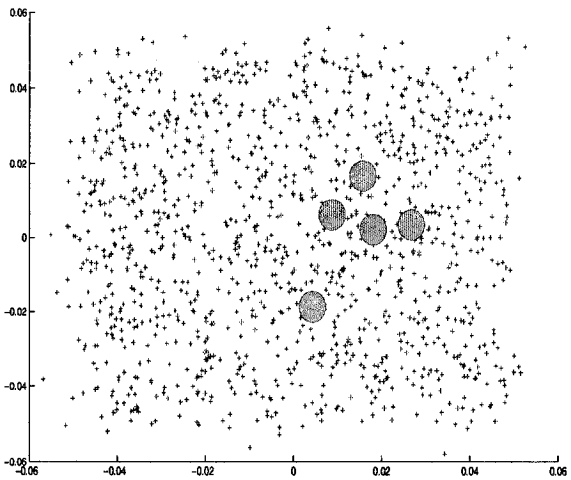


Figure 3.28: LTSA generalization *II* applied on the cube 3D model

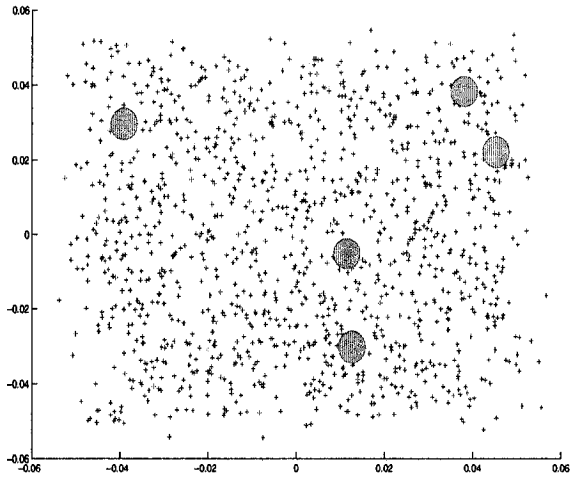


Figure 3.29: Incremental LTSA applied on the cube 3D model

3.5 Conclusion

LTSA belongs to a class of dimension reduction techniques. However, the inability of LTSA algorithm to adapt to newly added points has lead to our proposed incremental algorithm. This proposed method enables the algorithm to adjust to new arriving points while preserving the significant characteristics of the dataset. The proposed method follows a similar pattern of the incremental LLE and incremental HLLE and their generalizations.

CHAPTER 4

Application of Incremental Kernel PCA on Spectroscopy datasets

We present an experimental application using the Incremental Kernel Principal Component Analysis (IKPCA) algorithm [7], [8]. The IKPCA is applied on the spectra dataset. Spectra dataset is acquired from reflectance spectroscopy in which light is studied as a function of wavelength, providing useful information of materials and their elements. The IKPCA algorithm constructs a nonlinear high-dimensional feature space incrementally. The aim of this experimentation is to show how the IKPCA algorithm performs when applied on the spectra dataset.

4.1 Introduction

The study of datasets obtained from real-world applications is becoming of great interest in recent years, and with the increasing amount of information being acquired the size of these datasets is becoming considerably huge. Therefore, discovering the structure of these datasets and being able to analyze and use them in experimentation is of great significance in many engineering applications, including data visualization and pattern recognition. Having such datasets with large number of points, brings up the need for techniques that are able to handle these datasets in an effective and less time consuming manner.

Recently, Kernel Principal Component Analysis (KPCA) [7], [8] has been studied following the PCA technique. KPCA gives a set of nonlinear axes in the input space, thus allowing it to represent complex data distributions with a small number of axes. One drawback, however, is that KPCA can be applied in batch mode, i.e. it is not suitable for streaming data added sequentially, as is the case in real situations where input data are dynamically changed. Therefore, an incremental form of the algorithm was proposed to overcome this problem. This incremental Kernel PCA (IKPCA) algorithm extends from the Incremental PCA algorithm (IPCA) [28]. In the IKPCA algorithm, a set of linearly independent data is kept in memory in order to update the feature space. Since the number of independent data is determined for given training samples and eigen-axes are represented by a linear combination of these independent data, the algorithm is able to maintain a significant feature space efficiently. With this, an incremental form for the eigen-axes can be carried out and the new

eigen-axis is augmented according to an accumulation ratio that has to be within a certain threshold. Motivated by the incremental form of KPCA, we have conducted an experiment in which the IKPCA algorithm was applied to the spectra dataset.

This chapter is organized as follows. In the next section, we explain the Kernel PCA algorithm and the Incremental KPCA algorithm and the steps involved in the algorithm. In Section 4.3, we explain the significance of the Spectra dataset. In Section 4.4, we demonstrate, through experimental results, the performance of the IKPCA algorithm when applied to the spectra dataset, and provide a concise interpretation of the results. Finally, we conclude in Section 4.5.

4.2 Problem Formulation

4.2.1 Kernel PCA

The Kernel PCA algorithm produces a set of eigen-axes in a high-dimensional feature space. In this feature space, an n-dimensional input x is mapped onto an l-dimensional vector $\phi(x)$ where $\phi(\cdot)$ is the function that maps an input space into the l-dimensional feature space.

The algorithm involves the following steps [28]:

- **Step 1:** To obtain the eigenvectors in the feature space, the following covariance

needs to be defined:

$$Q = \frac{1}{N}\sum_{i=1}^{N}(\phi(x^{(i)}) - c)(\phi(x^{(i)}) - c)^T \tag{1}$$

where N is the number of input samples and $c = \frac{1}{N}\sum_{i=1}^{N}\phi(x^{(i)})$.

This calculation is practically difficult to carry out because the dimension of a feature space is generally very high. Hence, it is avoided by using the so-called kernel trick [29].

Assume that a set of m linearly independent samples $\{\phi(x_1), \ldots, \phi(x_m)\}$ span the space where N training samples $\{\phi(x^{(1)}), \ldots, \phi(x^{(N)})\}$ are distributed, where $m \leq N$.

- **Step 2:** After that, the kernel matrix H_{ij} is computed:

$$H_{ij} = \begin{bmatrix} k(x_1, x_1) & \ldots & k(x_1, x_j) \\ \vdots & \ddots & \vdots \\ k(x_i, x_1) & \ldots & k(x_i, x_j) \end{bmatrix} \tag{2}$$

where $k(\cdot)$ is a kernel function. Using Equations 1 and 2, the following kernel eigenvalue problem can be derived:

$$\frac{1}{N}L^{-1}H_{Nm}^T(I_N - 1_N)H_{Nm}(L^{-1})^T(L^T\alpha_i) = \lambda_i(L^T\alpha_i) \tag{3}$$

where $L^T\alpha_i$ is the ith eigenvector spanning the feature space and λ_i is the corresponding eigenvalue. Here L is obtained by the Cholesky factorization for H_{mm}(i.e. $H_{mm} = LL^T$).

- **Step 3:** The number d of eigenspace components is determined based on the following accumulation ration:

$$A_c(d) = \frac{\sum_{i=1}^{d} \lambda_i}{\sum_{i=1}^{m} \lambda_i} \tag{4}$$

for which d is chosen such that $A_c(d)$ is larger than a threshold value θ.

4.2.2 Incremental Kernel PCA (IKPCA)

In conventional Incremental PCA (IPCA) there are two operations involved: the augmentation of an eigen-axis and the rotation of the eigen-axis. On the other hand, in KPCA, the feature space degenerates into a lower dimensional space whose dimensions correspond to the number of linearly independent mapped input samples. Therefore the IKPCA algorithm must include three operations: a check on linear independence of the new sample to the samples given, an augmentation of eigen-axis, and a rotation of the eigen-axis.

The steps are as follows [28]:

- **Step 1:** A set of linearly independent samples $\{\phi(x_1), \ldots, \phi(x_m)\}$ are obtained

from N initial training samples. Then the kernel matrix in Equation 2 is calculated and the kernel eigenvalue problem in Equation 3 is solved to get α_i and which are used to calculate the accumulation ratio $A_c(d)$.

- **Step 2:** A new sample is set to x, and $N \leftarrow N + 1$.

- **Step 3:** The kernel matrix H' is calculated as follows:

$$H' = \begin{bmatrix} & & & k(x_1, x) \\ & H_{mm} & & \vdots \\ & & & k(x_m, x) \\ k(x, x_1) & \dots & k(x, x_m) & k(x, x) \end{bmatrix} \tag{5}$$

which includes H_{mm} as well as the kernel functions of x.

- **Step 4:** The accumulation ratio $A'_c(d)$ is calculated based on Equation 4 and the following updates:

$$\sum_{i=1}^{d} \lambda'_i = \frac{N}{N+1} \sum_{i=1}^{d} \left(\lambda_i + \frac{1}{N+1} \{\alpha_i^T (K_m(x) - \tilde{c})\}^2 \right) \tag{6}$$

$$\sum_{i=1}^{m} \lambda'_i = \frac{N}{N+1} \left(\sum_{i=1}^{m} \lambda_i + \frac{1}{N+1} \{k(x,x) - 2\beta^T \tilde{c} + \tilde{c}^T \tilde{c}\} \right) \tag{7}$$

where $\beta = (H_{mm})^{-1}K_m(x)$ and $K_m(x) = [k(x_1, x), \ldots, k(x_m, x)]^T$, and the mean vector $\tilde{c}$ can be calculated by:

$$\tilde{c'} = \frac{1}{N+1}(N\tilde{c} + K_m(x)) \tag{8}$$

- **Step 5:** If $A'_c(d) < \theta$, the following eigenvalue problem is solved with $d \leftarrow d+1$:

$$\frac{N}{(N+1)^2}\left((N+1)\begin{bmatrix}\Lambda & \mathbf{0}\\ \mathbf{0} & 0\end{bmatrix} + \begin{bmatrix}\boldsymbol{gg^T} & \boldsymbol{fg^T}\\ \boldsymbol{fg^T} & f^2\end{bmatrix}\right)R = R\Lambda' \tag{9}$$

where R is a rotation matrix, Λ' is the diagonal matrix of l eigenvalues, $g = [\alpha_1, \ldots, \alpha_d](K_m(x) - \tilde{c})$, and $f = \alpha^T(K_m(x) - \tilde{c})$.

On the other hand, if $A'_c(d) > \theta$, solve Equation 9 with $f = 0$. In both cases, the rotation matrix R and Λ' are obtained.

4.3 Application to Spectroscopy

4.3.1 Spectra Dataset

Spectra are information collected as a result of Reflectance spectroscopy, which is the study of light as a function of wavelength emitted, reflected, or scattered from different materials,

such as solid, liquid, or gas. The functions, i.e. spectra, provide useful information about the composition of a material, and are a powerful tool for identifying different elements in the material by way of examining the spectrum of the radiation of the material [30].

The process for acquiring and analyzing the spectra involves a source of light, an element to separate light into its component wavelengths, and a detector to sense the presence of light after wavelength separation. This method is helpful in applications concerning mineralogy, especially when other methods are more time consuming.

Figure 4.1, shows a spectra image (which will be used later in the experimentation of the IKPCA algorithm). These spectra were collected from the U.S. Geological Survey (USGS) Digital Spectral Library [31].

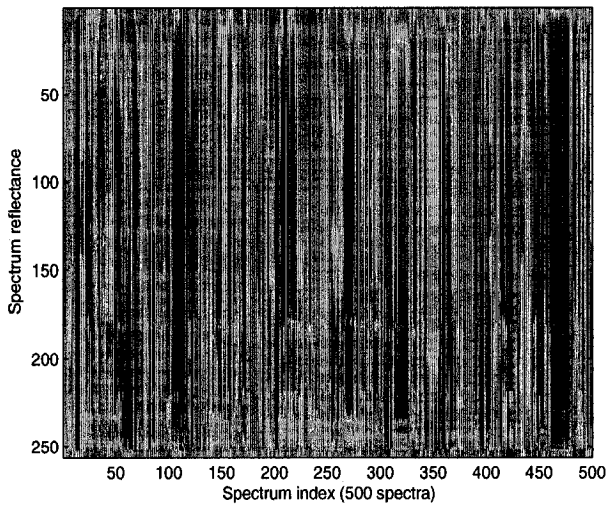


Figure 4.1: Spectra image

The image in Figure 4.1 show an image of spectrum data. In Figures 4.2 to 4.5, samples

from this image spectra are shown for different spectrum values. Each spectrum is a plot of reflectance versus wavelength.

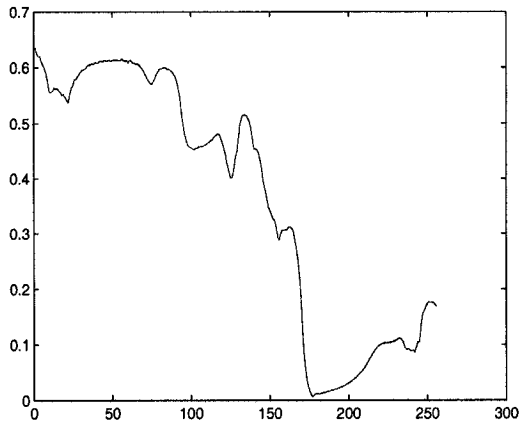


Figure 4.2: Wavelength sample from Spectra image (spectrum 50)

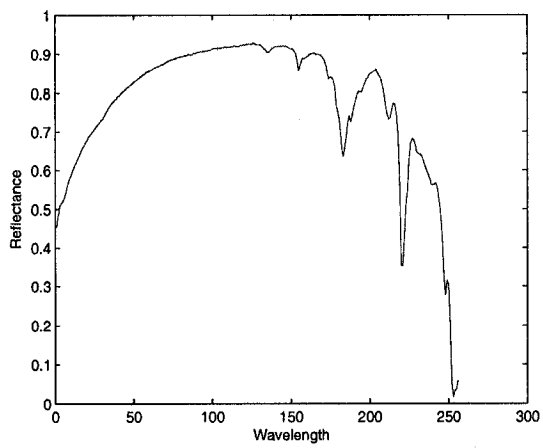


Figure 4.3: Wavelength sample from Spectra image (spectrum 85)

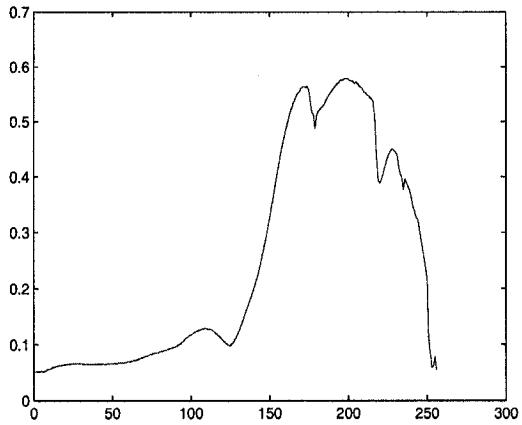


Figure 4.4: Wavelength sample from Spectra image (spectrum 225)

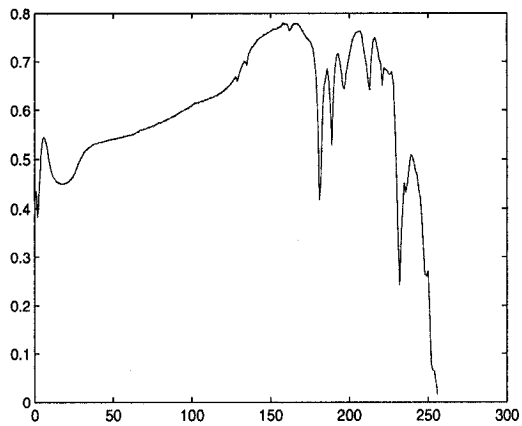


Figure 4.5: Wavelength sample from Spectra image (spectrum 500)

4.4 Experimental Results

This section presents experimental results in which the IKPCA algorithm is applied to the spectra dataset. The spectra dataset contains 500 points of 256 dimensions. The algorithm was run on the first 350 points and the remaining points were added incrementally. The threshold value θ was set to 0.7 for which the accumulation ratio $A'_c(d)$ must be higher. The results of the accumulation ratio was as follows:

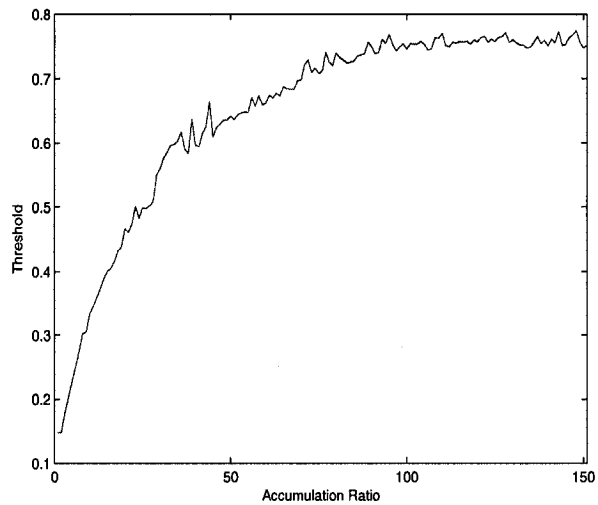


Figure 4.6: IKPCA application on Spectra dataset

The result presented in Figure 4.6 shows that the accumulation ratio starts with a low value at the beginning of the algorithm run. However, as the algorithm progresses, and more points are added to the dataset, the accumulation ratio is increasing gradually untill it reaches the threshold value set. At that point, the accumulation ratio is seen to be steady and the values are around that of the set threshold. This result shows that the accumulation

ratio runs in accordance with the algorithm to maintain the threshold value desired.

4.5 Conclusion

Discovering the structural behavior of a high-dimensional dataset is very helpful in dataset learning and analysis. An example of such datasets is the spectra dataset which is acquired by the study of light reflection and its wavelength components from different materials. This spectra dataset is useful in numerous applications including mineralogy. Hence, techniques to study such datasets is vital in engineering applications. KPCA is of those techniques that deal with large datasets for the purpose of analysis. However, KPCA has the limitations of not adapting to newly added points to the dataset and hence there is a need to rerun the algorithm for the entire set. Therefore, the IKPCA algorithm was proposed to overcome that drawback. In this chapter, we applied the IKPCA algorithm on the spectra dataset and studied its behavior. It has been shown that the algorithm succeeds in trying to keep the accumulation ration within the set threshold and maintain that value.

CHAPTER 5

Conclusions and Future Work

In this thesis, we presented algorithms for dimensionality reduction of multidimensional datasets. We showed the use of these algorithms through experimentation with 3D datasets and datasets of much higher dimension. We have demonstrated the effectiveness of the proposed methods in comparison with other methods and verified their efficiency through experimental application.

In the next Section, the contributions made in each of the previous chapters and the concluding results drawn from the associated research work are presented. Suggestions for future research directions related to this thesis are provided in Section 5.2.

5.1 Contributions of the thesis

5.1.1 Incremental Hessian Locally Linear Embedding

We introduced an incremental form of HLLE which enables the algorithm to adjust to new arriving points while preserving the significant characteristics of the dataset. The proposed method follows a similar pattern of the incremental LLE and its generalizations.

5.1.2 Incremental Local Tangent Space Alignment

We proposed an incremental form of LTSA which enables the algorithm to adjust to new arriving points while preserving the significant characteristics of the dataset. The proposed method follows a similar pattern of the incremental LLE and incremental HLLE and their generalizations.

5.1.3 Incremental Kernel PCA Application

We applied the IKPCA algorithm on the spectra dataset and studied its behavior. We showed that the algorithm succeeds in keeping the accumulation ration within the set threshold and maintain that value.

For future work, we plan to experiment with a variety of datasets and different techniques and extensions of the above three algorithms.

5.2 Future research directions

Several interesting research directions motivated by this thesis are discussed next. In addition to extending the functionality of certain dimensionality reduction techniques, we intend to research on other methods as well as attempt the following projects in the near future:

5.2.1 Incremental Kernel LLE

Following the concepts implemented in Incremental Kernel PCA, we have been working on implementing similar update methods to apply on the Kernel LLE algorithm.

Kernel LLE looks for an embedding that preserves the local geometry in the neighborhood of each data point. In the Kernel LLE algorithm, similar to regular LLE, the weight matrix $W = \{W_{i,j}\}$ is calculated which reconstructs each point from its nearest neighbors. This weight matrix will be used to find the kernel matrix, which will then be used to perform the incremental KPCA steps mentioned above.

The steps of calculating the kernel matrix using LLE is as follows:

- The sparse matrix of local predictive weights $W_{i,j}$ is constructed, such that $\sum_j W_{i,j} = 1$ and $W_{i,j} = 0$ if x_j is not the k-nearest neighbor of x_i and $\sum_j (W_{i,j}x_j - x_i)^2$ is minimized.
- The LLE matrix $M = (I - W)^T(I - W)$ is constructed.
- Then the kernel matrix $K = \lambda_{max}I - M$ is constructed, where λ_{max} is the maximum eigenvalue of M.

- Finally, $\tilde{K} = HKH$ which is the kernel matrix of the centered mapped data is constructed, where, $H = I - J/n$ in which I is an identity matrix and J is a matrix of all ones.

This K is the equivalent of H_{mm} in the KPCA implementation. This matrix will be used to perform the steps mentioned in Incremental KPCA in order to obtain the accumulation ratio, the rotation matrix R and Λ'.

5.2.2 Incremental Nonnegative Matrix Factorization (NMF)

NMF can be applied to the statistical analysis of multidimensional data [33]. Given a set of n-dimensional data vectors, the vectors are placed in the columns of matrix V of size $n \times m$, where m is the number of samples in the dataset. This matrix is then approximately factorized into matrices W and H of size $n \times r$ and $r \times m$ respectively. Usually r is chosen to be smaller than n or m, so that W and H are smaller than the original matrix V. This results in a compressed version of the original data matrix giving:

$$V \approx WH$$

Similarly, Singular Value Decomposition (SVD) takes a matrix of data M of size $n \times m$ and produces

$$M = USV^T$$

where the columns of U (size $n \times r$) are the left singular vectors, V^T (size $r \times m$) has rows

that are the right singular vectors, and S (size $r \times 1$) has a diagonal of singular values.

An incremental form of SVD has been proposed by Brand in [32]. We have been working on finding a relation between NMF and SVD such that the incremental SVD can be subsequently applied to the NFM algorithm to enable it to adapt to newly added points.

List of References

[1] R. Bishop, R. Crittenden, *Geometry of Manifolds*, American Mathematical Society, Providence, 2001.

[2] M. H. Law, N. Zhang, and A. K. Jain, "Nonlinear Manifold Learning for Data Stream," *Proceedings of SIAM Data Mining*, pp. 33-44, Orlando, Florida, 2004.

[3] D. Hand, H. Mannila, and P. Smyth, *Principles of Data Mining*, MIT Press, Cambridge, Massachusetts, 2001.

[4] C. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2007.

[5] I. Jolliffe, *Principal Component Analysis*, New York, Springer, 1986.

[6] L. Smith, "A Tutorial on Principal Component Analysis," February 2002

[7] B. Schölkopf, A. Somola, and K. Müller, "Nonlinear Component Analysis as a Kernel Eigenvalue Problem," *Neural Computation*, vol. 10, pp. 1299-1319, 1998.

References

[8] S. Abe, *Support Vector Machines for Pattern Classification*, Springer, London, 2005.

[9] M. Law, A. Jain, "Incremental Nonlinear Dimensionality Reduction by Manifold Learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Vol 28, no. 3, March 2006.

[10] S. Roweis, L. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323-2326, December 2000.

[11] D. Donoho, C. Grimes, "Hessian Eigenmaps: new locally linear embedding techniques for high-dimensional data," *Proceedings of National Academy of Science of the United States of America*, pp. 5591-5596, 2003.

[12] G. Golub, C. Van Loan, *Matrix Computations*, 3rd ed. Baltimore, MD: Johns Hopkins, 1996.

[13] Z. Zhang, H. Zha, "Principal manifolds and nonlinear dimensionality reduction via tangent space alignment," *SIAM Journal on Scientific Computing*, vol. 26, no. 1, pp. 313-338, 2004.

[14] M. Belkin, P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Technical Report TR-2002-01*, University of Chicago, Department of Computer Science, 2002.

[15] O. Kouropteva, O. Okun, and M. Pietikäinen, "Incremental Locally Linear Embedding Algorithm," *Lecture Notes in Computer Science 3540*, pp. 521-530, 2005.

[16] M. Davison, "Multidimensional Scaling," Florida: Krieger Publishing Company, 1992.

[17] J. Tenenbaum, V. de Silva, and J. Langford, "A Global Geometric Framework for Nonlinear Dimensionality Reduction,", *Science*, vol. 290, pp. 2319-2323, December 2000.

[18] M. Bernstein, V. de Silva, J. Langford, and J. Tenenbaum, "Graph approximations to geodesics on embedded manifolds,", *Technical report*, Department of Psychology, Stanford University, 2000.

[19] T. Hastie, W. Stuetzle "Principal curves," *Journal of the American Statistical Association*, vol. 84, no. 406, pp. 502-516, 1989.

[20] J. Costa, A. Hero, "Geodesic Entropic Graphs for Dimension and Entropy Estimation in Manifold Learning," *IEEE Trans. on Signal Processing,* vol. 52, no. 8, pp. 2210-2221, August, 2004.

[21] O. Kouropteva, O. Okun, and M. Pietikäinen, "Beyond locally linear embedding algorithm," *Technical Report MVG-01-2002*, University of Oulu, 2002.

[22] L. Saul, S. Roweis, "Think globally, fit locally: unsupervised learning of nonlinear manifolds," *Journal of Machine Learning Research*, vol. 4, pp. 119-155, 2003.

[23] O. Abdel-Mannan, A. Ben Hamza, and A. Youssef, "Incremental Dimensionality-Reduction Algorithm using Heissien Locally Linear Embedding," *Proc. IEEE International Symposium on Signal Processing and its Applications*, Sharjah, UAE, 2007.

[24] O. Abdel-Mannan, A. Ben Hamza, and A. Youssef, "Incremental Local Tangent Space Alignment Algorithm," *Proc. IEEE Canadian Conference on Electrical and Computer Engineering*, Vancouver, Canada, 2007.

[25] M. Koschat, D. Swayne, "A Weighted Procrustes Criterion," *Psychometrika*, vol. 56: pp. 229-239, Springer, New York, 1991.

[26] de Ritter D, Kouropteva O, Okun O, Pietikäinen M, and Duin RPW, "Supervised locally linear embedding," *Artificial Neural Networks and Neural Information Processing, ICANN/ICONIP 2003 Proceedings,* Lecture Notes in Computer Science 2714, Springer, 333-341.

[27] http://www.ics.uci.edu/ mlearn/databases/wine/

[28] S. Kimura, S. Ozawa, and S. Abe, "Incremental Kernel PCA for Online Learning of Feature Space," *CIMCA-IAWTIC'06*, vol. 1, pp. 595-600, 2005.

[29] M. Aizerman, E. Braverman, and L. Rozonoer (1964). "Theoretical foundations of the potential function method in pattern recognition learning," *Automation and Remote Control*, vol. 25, pp. 821-837, 1964.

[30] J. Chamers, P. Griffiths, *Handbook of Vibrational Spectroscopy*, New York: Wiley, 2002.

[31] http://speclab.cr.usgs.gov/spectral-lib.html

[32] M. Brand, "Incremental Singular Value Decomposition of Uncertain Data with Missing Values," *European Conference on Computer Vision (ECCV),* Vol 2350, pps 707-720, May 2002.

[33] D. Lee, H. Seung, "Algorithms for non-negative matrix factorization," *Advances in Neural Information Processing Systems 13,* 2001.

[34] J. Tenebaum, V. De Silva, and J. Langford, "A Global Geometric Framework for Nonlinear Dimensionality Reduction," *Science*, vol. 290, pp. 2319-2323, 2007.

[35] H. Law, N. Zhang, and A. Jain, "Nonlinear Manifold Learning for Data Stream," *Proceedings of SIAM Data Mining*, pp. 33-44, Orlando, Florida, 2004.

[36] S. Berrani, L. Amsaleg, and P. Gros, "Approximate k-nearest-neighbor searches: a new algorithm with probabilistic control of the precision," *Technical Report, RR-4675*, INRIA, 2007.

[37] P. Hall, D. Marshall, and R. Martin, "Incremental Eigenanalysis for Classification," *Department of Computer Science, University of Wales Cardiff, UK, Research Report Serious*, no. 98001, May 1998.

[38] Y. Bengio, J.-F. Paiement, and P. Vincent, "Out-of-sample extensions for LLE, Isomap, MDS, eigenmaps, and spectral clustering," *Advances in Neural Information Processing Systems 16*, MIT Press, 2004.

[39] J. Costa, A. Hero, "Manifold learning using Euclidean K-nearest neighbor graphs," *Proceedings of IEEE International Conference on Acoustic Speech and Signal Processing*, vol. 4, pp. 988-991, Montreal, May, 2004.

[40] A. Ben Hamza, D. Brady , "Reconstruction of reflectance spectra using robust non-negative matrix factorization," *IEEE Transactions on Signal Processing*, vol. 54, no. 9, pp. 3637-3642, September 2006.

[41] S. De Backer, A. Naud, and P. Scheunders, "Non-linear Dimensionality Reduction Techniques for Unsupervised Feature Extraction," *Pattern Recognition Letter*, vol. 19, no. 8, pp. 711-720, June 1998.

[42] M. Benito, D. Pena, "A fast approach for dimensionality reduction with image data," *PR*, vol. 38, no. 12, pp. 2400-2408, December 2005.