# A Statistical Study On Locally Linear Embedding Modified With Different Distance Metrics

Dissertation

Submitted in partial fulfillment of the requirements for the

M. Tech (Computer Science)

degree of the Indian Statistical Institute

By

**Sourya Poddar**

Under the supervision of

**Dr. Swagatam Das**

Electronics and Communication Sciences Unit

Indian Statistical Institute, Kolkata

# Declaration

I, **Sourya Poddar** (**CS1324**), registered as a student of **M. Tech** program in **Computer Science**, **Indian Statistical Institute**, **Kolkata** do hereby submit my Dissertation Report entitled "**A Statistical Study On Locally Linear Embedding Modified With Different Distance Metrics**". I certify

1. The work contained in this Dissertation Report is original and has been done by me under the guidance of my supervisor.

2. The material contained in this Dissertation Report has not been submitted to any University or Institute for the award of any degree.

3. I followed by guidelines provided by the Institute in preparing the report.

4. Whenever I have used materials (data, theoretical analysis, figures, and text) from other sources, I have given due credit to them by citing them in the text of report and giving their details in the bibliography.

.....

**Place : ISI, Kolkata** **Sourya Poddar**

**Date : July, 2015** **(CS1324)**

# Certificate

This is to certify that the thesis titled "**A Statistical Study On Locally Linear Embedding Modified With Different Distance Metrics**" submitted by **Sourya Poddar** in partial fulfillment for the award of the degree of **M. Tech** in **Computer Science** is a bonafide record of work carried out by him under our supervision. The thesis has fulfilled all the requirements as per the regulations of this Institute and, in my opinion, has reached the standard needed for submission. The results embodied in this thesis have not been submitted to any other university for the award of any degree or diploma.

.....

**Dr. Swagatam Das**
**Electronics and Communication Sciences Unit**
**Indian Statistical Institute, Kolkata**

# Acknowledgement

I take this opportunity to express my deepest gratitude and appreciation to all those who have helped me directly or indirectly towards the successful completion of this thesis. First and foremost, I am grateful to my supervisor Dr. Swagatam Das for his immense help, co-operation, motivation, and guidance to carry out this work. His continued support in the form of discussion, constructive criticism, and clear understanding has helped me to proceed in the right direction.

I would also like to thank all professors, lab administrators and friends for their continuous and unconditional support. I am also grateful to Mr. Shounak Datta for his immense support and help during the duration of this project. The smooth completion of this project would not have been possible without his encouragement.

I want to especially thank my Mom, Dad, and Brother, without whom my life would not be possible. The smooth completion of this project has only been possible because of everyone's support and good will.

Sourya Poddar

# Abstract

High-dimensional data means data with large number of features and samples. Sometimes, the number of features may even be larger than the number of samples. So, it gets very difficult and computationally complex to handle such huge amount of data. High-dimensional data are obtained from different domains like engineering, informatics, biometrics, neuroimaging, etc. These data needs to be processed and classified or clustered while using in pattern recognition, image processing, information retrieval, computer vision, etc. Classifying these data is a difficult problem because of the enormous size of the data. Conventional classification methods can also handle such huge data but takes a lot of computational time which makes many classical techniques impractical. A trivial approach of solving this problem is to apply any dimensionality reduction technique followed by a classification method. There are many linear and non-linear dimensionality reduction techniques. Some of the popular methods are Principal Component Analysis, Linear Discriminant Analysis, Factor Analysis (linear), Sammon's Mapping, Locally Linear Embedding, Isomap, Local Tangent Space Alignment, Self-Organising Map, Laplacian Eigenmap (non-linear), etc. These dimensionality reduction techniques varies in terms of basic assumptions about data, principles, computational complexity, objectives (preserving local or global properties), etc.

# Table of Contents

# List of Tables

# List of Figures

# Abbreviations

***PCA*** ..... Principal Component Analysis

***MDS*** ..... Multi-Dimensional Scaling

***SOM*** ..... Self-Organising Map

***LTSA*** ..... Local Tangent Space Alignment

***LLE*** ..... Locally Linear Embedding

***ICA*** ..... Independent Component Analysis

***LDA*** ..... Linear Discriminant Analysis

***SVD*** ..... Singular Value Decomposition

***FA*** ..... Factor Analysis

***ANN*** ..... Artificial Neural Network

***MVU*** ..... Maximum Variance Unfolding

***SDP*** ..... Semidefinite Programming Problem

***KPCA*** ..... Kernel Principal Component Analysis

***LLC*** ..... Locally Linear Coordination

***EM*** ..... Expectation Maximization

***RBM*** ..... Restricted Boltzmann Machine

***UCI*** ..... University of California, Irvine

***MNIST*** ..... Mixed National Institute of Standards and Technology

***k-NN*** ..... k-Nearest Neighbour

***EC*** ..... Euclidean Distance Method

***DF*** ..... Diffusion Distance Method

***CT*** ..... Commute-time Distance Method

***BI*** ..... Biharmonic Distance Method

***MMR*** ..... Minimum Misclassification Ratio

***ANOVA*** ..... Analysis of Variance

# Chapter 1

# Introduction

## 1.1 Motivation

Dimensionality reduction techniques play a major role in handling huge data and extracting relevant information from them. So, improving this techniques will help immensely in working with real-datasets. One of the popular dimensionality reduction technique is Locally Linear Embedding. The main motivation of our work is to enhance the performance of Locally Linear Embedding technique. The performance of this technique is evaluated based on the misclassification results obtained while classifying the low-dimensional embeddings of the high-dimensional real datasets.

An important stage of Locally Linear Embedding technique is to compute the $K$ nearest neighbouring points of all the datapoints based on the pairwise distances between them. In the classical Locally Linear Embedding technique, Euclidean distance is used as the distance measure. While Euclidean distance computes the shortest distance between two datapoints, it may not give the desired result. This is because the high-dimensional real-data may not lie in a Euclidean space, so it

may not exhibit Euclidean geometry properties globally. Our aim is to overcome this bottleneck of Locally Linear Embedding by modifying this existing distance measure.

So, instead of using the Euclidean distance measure, we tried some different distance measures like diffusion distance, commute-time distance and biharmonic distance. The working principle behind these distance measures are graph connectivity, random walk and timestep. Based on these distance measures, the high-dimensional real datasets are embedded into low-dimensional space using Locally Linear Embedding.

The low-dimensional data are classified using the classical $k$-Nearest Neighbour classification technique. The misclassification errors as obtained from the classification method are regarded as the parameter for evaluating the performances of these distance-measure methods.

The performances of these distance-measure methods are analyzed based on some non-parametric statistical tests like Friedman Test, Multiple Sign Test, Median based Contrast Estimation Test, Friedman Aligned Rank Test and Quade Test.

Apart from the Locally Linear Embedding, there are some other non-linear dimensionality reduction techniques like Laplacian Eigenmap, Isomap, Local Tangent Space Alignment and Maximum Variance Unfolding that also involves computing the nearest neighbours based on Euclidean distance. These techniques can also be modified with these three different distance measures i.e. diffusion distance, commute-time distance and biharmonic distance.

## 1.2 Thesis Outline

The remaining chapters are organized as follows.

In Chapter 2, we have discussed in details the significance of dimensionality reduction and its uses in manifold learning. We have also discussed in details some of the traditional and widely used dimensionality reduction techniques. They can be both linear and non-linear.

In Chapter 3, we have mentioned in details the working principle behind Locally Linear Embedding and its algorithm, complexity along with the advantages and disadvantages of using Locally Linear Embedding.

In Chapter 4, we have presented our proposed modifications on Locally Linear Embedding. The different distance measures used in our proposed modification are described in details.

In Chapter 5, we have performed all the nonparametric statistical testing. These testing are done on the results obtained after classifying the low dimensional embedding. These embedding's are done by using Locally Linear Embedding while the classification is done by applying $k$-Nearest Neighbour classification technique.

In Chapter 6, we have discussed the result of the experiments and gave a brief analysis of the outcome of our work. This section contains the conclusion of our study. Apart from this, we have given some lines of expanding our work for further research and development.

# Chapter 2

# Dimensionality Reduction

## 2.1 General Idea

High-dimensional datasets provide both challenges and opportunities. It also helps to develop new theories. A major aspect of dealing with high-dimensional data is that all the features of the datasets are not relevant. Some of the features of the dataset are redundant and do not contribute in understanding the structure of the data. There are some methods that can handle high-dimensional data efficiently. They can construct predictive models with high accuracy. But the problem with these methods is that they have high computational complexity. So, dimensionality reduction plays a major role in handling high dimensional data. Before constructing the model, the dimension of the high-dimensional datasets may need to be reduced.

Many machine learning algorithms tend to have high-dimensional data. Handling high dimensional data is an important aspect of statistical pattern recognition. High dimensional data requiring more than three dimensions can be difficult to represent, visualize and interpret. By reducing the dimensions of the data, the analysis algorithms can perform more efficiently by giving more accurate results

Many real-life high-dimensional datasets can be represented as datapoints lying close to a low-dimensional nonlinear manifold. The geometrical structure of the manifolds can be obtained from a set of sampled datapoints. In general, dimensionality reduction is an unsupervised learning problem. It can be difficult to solve this problem as the sampled datapoints may contain noise. The low-dimensional structures obtained after dimensionality reduction can be used for performing classification, clustering, outlier detection and data visualization.

There are many traditional dimensionality reduction techniques like Principal Component Analysis (PCA) or Multi-Dimensional Scaling (MDS). Both are eigenvector methods which captures linear variabilities in high dimensional data. Traditional dimension reduction techniques generally give better results when datapoints are lying close to a linear subspace in the input space. It gets difficult for them to discover nonlinear structures which remains embedded in the datapoints.

PCA is a linear dimensionality reduction algorithm which computes the linear projections of greatest variance from the top eigenvectors of the data covariance matrix. It is a statistical procedure using an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components. When PCA reduces a high dimensional dataset into two dimensions, the output of the reduced dimension values may not be well organized and cause overlaps and thereby losing the entire property of embedding in lower dimensional data. This shows that the high-dimensional vectors sampling the manifold varies non-linearly.

On the other hand, MDS tries to preserve the inter-point distances as observed in higher-dimensional space in the lower-dimensional projection as well. This is a structure preserving mapping. When distance metric is Euclidean, PCA and MDS gives same result.

Apart from the traditional linear dimensionality reduction techniques like PCA

and MDS, there are also some non-linear dimensionality reduction methods like Self-Organizing Map (SOM), Sammon's mapping, Isomap, Laplacian Eigenmap, Local Tangent Space Alignment (LTSA), Locally Linear Embedding (LLE), etc.

The non-linear dimensionality reduction methods generally exhibit two kind of properties, so they are classified into two categories. One type deals with providing a mapping either from the high-dimensional space to the low-dimensional embedding or from the low-dimensional space to the high-dimensional embedding. Beside, they also provides the preliminary feature extraction step. Based on the results obtained from this step, the pattern recognition algorithms are applied. On the other hand, the second kind of non-linear dimensionality reduction techniques is more concerned with only giving a visualization of the data. They are based on the proximity of the data i.e. distance measurements.

## 2.2 Curse of Dimensionality

Algorithms designed with intuitions based on two or three-dimensional spaces may face unexpected problems when dealing with high-dimensional data. The phenomenon which maybe most relevant for dimension reduction is the concentration of distances. When number of dimension components grows, contribution of any one axis becomes more and more meaningless. It becomes difficult to tell two points apart unless they differ in most dimensions. Most points are roughly equally far from each other, and the distribution of pairwise distances becomes peaked.

An important practical consequence of this is that the concept of nearest neighbour is weakened. When the difference in distances to the nearest and the furthest neighbours becomes less than the measurement noise, algorithms relying on neighbourhoods break down. This and other peculiarities of high-dimensional spaces are referred to as the curse of dimensionality.

Curse of dimensionality can severely affect distance-based algorithms, and become noticeable already in 5 to 10 dimensions, which is much less than the number of attributes in many modern datasets. These distance-based algorithms are used based on the basic assumption that the intrinsic dimensionality of the data is fairly low. Non-linear dimensionality reduction, sometimes thought as a remedy for the curse of dimensionality. It implicitly relies on the assumption of a low intrinsic dimensionality. Most dimensionality reduction methods use nearest neighbours or comparisons of pairwise distances, and are thus are equally sensitive to the curse of dimensionality as other machine learning methods.

## 2.3 Intrinsic Dimensionality

The number of attributes is usually different from dimensionality. In problems with less data points than dimensions, the number of data points dictates the maximum dimensionality of the subspace that the data vectors can span. Sometimes data can contain directions that are mostly noise, or it can have attributes that are mostly irrelevant for the task in hand and can be left out by variable selection. And, even more importantly, data may contain dependencies which effectively lower its dimensionality. The smaller dimensionality caused by dependencies is referred to as the intrinsic dimensionality of data, or the number of degrees of freedom that is sufficient for describing the data.

One of the most important parameter of dimensionality reduction is this intrinsic dimensionality. For data visualization, we need to view data in two-dimension or three-dimension. So in those cases, we are forced to set intrinsic dimension to two or three. There is a chance of considerable information loss in such cases.

## 2.4 Manifold Learning

A manifold can be defined as a topological space which resembles Euclidean space near each point. Each point of an $n$-dimensional manifold has a neighbourhood that is homeomorphic to the Euclidean space of dimension $n$. Some example of one-dimensional manifolds are lines and circles. The two-dimensional manifolds are known as surfaces. Some example of two-dimensional manifolds are planes, torus, sphere, etc. All two-dimensional manifolds may not be possible to be embedded in three dimensional space.

Manifold exhibit the properties of Euclidean space locally. The space near each point may be considered to be Euclidean space. But globally, it generally do not behave like an Euclidean space.

Manifold is an important idea which is adopted in several scientific fields like geometry, modern physics, mathematics, etc. Highly complicated structures of datasets can described with the Euclidean space properties. Manifolds are generally obtained as graph of function or while solving systems of equations. Manifolds may have additional features like they can be differentiable (allow applying calculus on manifolds), Riemannian (allow measuring distances and angles), Symplectic (handle phase spaces in the Hamiltonian formalism of classical mechanics) or Lorentzian (deals with general relativity in space-time model).

Non-linear dimensionality reduction can be done by the help of manifold learning. The basic principle behind manifold learning is that the some dimensions of the high-dimensional datasets are redundant. Manifold Learning generalizes linear frameworks to perform efficiently on non-linear datasets as well. Manifold learning is generally an unsupervised learning although supervised learning methods are also present. In unsupervised learning, the geometry of the high-dimensional dataset is learnt from the data itself, there is no need of using predetermined classifications.

The three basic steps of most manifold learning algorithms are

1. Construction of the nearest neighbour graph based on the pairwise distances between the sample datapoints.

2. Linear approximation of the local manifold geometry inside the neighbourhood of each sample datapoint.

3. Minimization of the global error function which provides the global embedding. This is done by solving an eigenvalue problem.

An important step for non-linear dimensionality reduction method is to effectively select the neighbourhood size to construct the neighbourhood graph. The aim of manifold learning is to uncover the intrinsic coordinate system of the manifold from which the data is sampled. Manifolding learning enables 'unfolding' of the manifold for better visualization. Dimensionality reduction is done using these coordinates instead of the Euclidean coordinates of the embedding space. The manifold coordinates can be approximated by studying the data in a local scale, and then combining local, linear views into a global nonlinear shape. Local views are obtained by restricting the analysis to the nearest neighbours of a data point.

Manifolds learning is not only concerned with the ways of looking at the structure of data. Some other aspects of manifold learning are clustering of the data, generative topographic mapping, classification of labeled data, emphasizing its natural density structure or simple unfolding of manifolds. Manifold learning rely heavily on neighbourhoods, but neighbourhoods can be exploited without bringing in strong manifold assumptions.

## 2.5 Dimensionality Reduction Methods

Dimensionality reduction is the method of transforming high-dimensional data into a low-dimensional data. The representation in the low dimension is meaningful. In ideal situations, the dimensionality of the low-dimensional representation of a dataset is equal to the intrinsic dimensionality of that data. The minimum number of parameters that meaningfully represents the observed properties of the data is called its intrinsic dimensionality.

Let us now formally define the dimensionality reduction problem. Assume we have dataset represented in a $n \times D$ matrix $X$ consisting of $n$ data vectors $x_i$ (i ε1,2,...,n) with dimensionality $D$. Let the intrinsic dimensionality of this dataset be $d$ where $d < D$ and often $d << D$. The high-dimensional dataset $X$ is transformed into low dimensional dataset $Y$ by the dimensionality reduction techniques. We get a mapping from $D$ dimension to $d$ dimension. The geometry of the dataset is preserved as much as possible. In practice, we do not know the intrinsic dimensionality or geometry of the data manifold. So, the dimensionality reduction problems can only be solved based on some assumptions like assuming the intrinsic dimension of the dataset.

Dimensionality reduction methods are of two types. They are

1. Linear Dimensionality Reduction
2. Non-Linear Dimensionality Reduction

There are several dimensionality reduction techniques, both linear and non-linear. They are discussed in the following sections.

## 2.6 Linear Methods

The technique of embedding high-dimensional data into a low-dimensional subspace is called dimensionality reduction. When the data lie on or near a linear subspace of the high-dimensional space, the technique applied is linear in nature. Some of the popular linear dimensionality reduction techniques existing are Principal Component Analysis (PCA), Factor Analysis (FA), Independent Component Analysis (ICA), Linear Discriminant Analysis (LDA) and Singular Value Decomposition (SVD). But these linear dimensionality reduction techniques cannot adequately handle complex non-linear data. Among these techniques, PCA gives the best performance. So it is the most popular (unsupervised) linear dimensionality reduction technique.

A brief discussion on some of the linear dimensionality reduction techniques are given below.

### 2.6.1 Principal Component Analysis

Principal Component Analysis (PCA) is the most popular and best linear dimensionality reduction technique that is based on minimizing the least square error. PCA gives a low-dimensional representation of the data which gives us the information about the variance in the data. For the dataset, PCA finds a linear basis of reduced dimensionality which have the maximal amount of variance in the data. The dimension of the data is reduced by finding a few orthogonal linear combinations of the original variables with the largest variance. This orthogonal combinations are known as Principal Components. The first principal component gives us the linear combination with maximum variance, second principal component gives the second maximum variance and so on.

Mathematically, PCA tries to find a linear mapping $M$ such that it maximizing

$M^T cov(X)M$. Here, $cov(X)$ is the covariance matrix of the data $X$. The mean of each variable is subtracted from the variable to make the data zero-mean. This linear mapping is formed by the $d$ principal eigenvectors of the covariance matrix of this zero-mean data. PCA solves the eigenproblem $cov(X)M = \lambda M$. This Eigen problem is solved to obtain the $d$ principal eigenvalues $\lambda_i$ and its corresponding eigenvectors $v_i$. The low-dimensional data representations $y_i$ of the datapoints $x_i$ are computed by mapping them onto the linear basis $M$, i.e., $Y = (X - X_m)M$

The main drawback of PCA is that the size of the covariance matrix is proportional to the dimensionality of the datapoints. When the dimension of the dataset is very large, it may be infeasible to compute the eigenvectors of the covariance matrix. In datasets where $n < D$, this drawback can be handled by computing the eigenvectors of the matrix $(X - X_m)(X - X_m)^T$

### 2.6.2 Independent Component Analysis

Independent Component Analysis (ICA) [Common(1994), Hyvarinen(1999), Hyvarinen(2000), Hyvarinen(2001)] is an example of information-theory based algorithm. It is a generative model. While PCA looks for uncorrelated factors i.e, a constraint on the second-order statistics, ICA looks for independent factors i.e, a constraint on all their moments. This is an advantage if the factors are truly independent. The equation for the generative model is $X = W_m x_m$ where $x_m$ is the source, $W_m$ is the mixing matrix and $X$ is the observation. The target is to compute the unmixing matrix $W_u$ such that the components of $x_u = W_u^T X$ are as independent as possible. The sources are recovered in different scale and order. A limitation of this technique is that usually sources are supposed to be non-Gaussian since the linear combination of two Gaussian variables is also Gaussian making the separation an ill-posed problem. Different ICA methods differ in the way they measure the independence of the estimates of the source vari-

ables, resulting in different estimates of the $W_u$ and $x_u$. The different methods are Non-Gaussianity, Maximum Likelihood and Non-linear decorrelation.

### 2.6.3 Linear Discriminant Analysis

Linear Discriminant Analysis (LDA) is a supervised dimensionality reduction technique. The feature selection in traditional LDA is obtained by maximizing the difference between classes and minimizing the distance within classes. For better separation, the high dimensional space is reduced into low dimensional subspace. The optimization function is a function of $v$ where $v$ is the projection vector separating the two classes. The optimization function is

$$J(v) = \frac{(M_1 - M_2)^2}{s_1^2 + s_2^2} \tag{2.1}$$

where $M_1$, $M_2$ are the means of the projected classes while $s_1$, $s_2$ are the scatter of the projected classes. The aim is to maximize the objective function i.e. maximize the distance between class-means and minimize the intra-class scatter. Now, after some linear algebra and calculus, the optimization function boils down to

$$J(v) = \frac{v^T S_B v}{v^T S_W v} \tag{2.2}$$

Here, $S_B$ is the between-class scatter matrix and $S_W$ is the within-class scatter matrix. $S_B$ is represented as $S_B = (m_1 - m_2)(m_1 - m_2)^T$ where $m_1$, $m_2$ are the mean of the two classes. $S_W$ is defined as $S_W = S_1 + S_2$ where $S_1$ and $S_2$ are the class scatter matrix represented as $S_1 = \sum_{x_i \epsilon class1}(x_i - m_1)(x_i - m_1)^T$ and $S_2 = \sum_{x_i \epsilon class2}(x_i - m_2)(x_i - m_2)^T$

The linear discriminant vectors are determined by the LDA algorithm based on the global structure information of the dataset. These vectors are all global in nature. But in the case of a test datapoint, the local linear discriminant vectors are used instead of global ones while extracting features from the datapoints. This

is because in case of global discriminant vectors, classification may be incorrect. The major drawbacks of LDA are Small Sample Size problem and Common Mean problem. When the dimensionality of the dataset is greater than the number of samples, it results to the small sample size problem. Common mean problem occurs when the classes have same mean i.e $J(v) = 0$. Beside, LDA is sensitive to outliers.

### 2.6.4 Singular Value Decomposition

Let $M$ be a $m \times n$ matrix, and let the rank of $M$ be $r$. Based on the Singular Value Decomposition (SVD), we will get a decomposition of the matrix $M$ into matrices $U$, $\Sigma$ and $V$ such that

1. Matrix $U$ is a $m \times r$ column-orthonormal matrix. Each of the columns of $U$ is a unit vector. The dot product of every two columns is 0.

2. Matrix $V$ is a $n \times r$ column-orthonormal matrix where the rows of $V^T$ are orthonormal in nature.

3. Matrix $\Sigma$ is a diagonal matrix having all the non-zero elements in the principal diagonal.

Thus, $M$ can be written as

$$M = U\Sigma V^T \tag{2.3}$$

We do not get a unique decomposition when the rank of $M$ is greater than the number of columns in the matrices $U$, $\Sigma$ and $V$. In order to get the best solution, the columns of $U$ and $V$ corresponding to the smallest singular values in the exact decomposition needs to be eliminated. The elements of $S$ are called the singular values of $M$.

One of the major application of SVD is in dimensionality reduction. A high dimensional dataset may be too large to be handled. So, it may be decomposed into three matrix by its SVD components $U$, $\Sigma$ and $V$. Even these matrices may be too large to tackle. Then dimensionality of these three matrices are reduced by taking only the non-negative singular values of the matrix $\Sigma$. The corresponding rows of $U$ matrix and $V$ matrix can be eliminated. Similarly, if we want to reduce the dimension by $x$, we can simply eliminate the rows of $U$ matrix and $V$ matrix, corresponding to the smallest $x$ singular values. Thus, by using this decomposition method, the dimension of the matrix $M$ is reduced.

### 2.6.5 Factor Analysis

Factor analysis (FA) [Spearman(1904), Thurstone(1947), Kaiser(1960), Lawley(1971), Mulaik(1971), Harman(1976)] is another statistical technique intimately related to PCA and dimensionality reduction. It is a generative model that assumes that the observed data has been produced from a set of latent, unobserved variables (called factors) through the equation $X = Wx + n$. In this technique all the variances of the factors are absorbed into $W$ such that the covariance of $X$ is an identity matrix. Factors are assumed to follow a multivariate normal distribution, and to be uncorrelated to noise. Under these conditions, the covariance of the observed variables is $C_x = WW^T + C_n$ where $C_n$ is the covariance matrix of the noise and has to be estimated from the data. Matrix $W$ is solved by factorization of the matrix $WW^T = C_x - C_n$. This factorization is not unique since any orthogonal rotation of $W$ results in the same decomposition of $C_x - C_n$. This fact is exploited to produce simpler factors in the same way as the PCA is rotated. The rotation is done in the same method as that in PCA. If there is no noise, FA is same as PCA but PCA is not a generative model.

## 2.7 Non-linear Methods

In this section, we discuss in details some of the popular nonlinear dimensionality reduction techniques. The main difference between linear and non-linear dimensionality reduction methods is that the non-linear dimensionality reduction techniques do not assume that the datapoints lie near a linear subspace in high-dimension. So, non-linear techniques can handle more complex high-dimensional datasets with better efficiency and embed them more accurately in the low-dimensional space.

The nonlinear techniques can handle complex nonlinear data. In practice, non-linear dimensionality reduction techniques give better performance than linear methods while handling real-world datasets as they are more non-linear in nature. Based on some previous studies, it is observed that linear techniques perform poorly in comparison to non-linear techniques. On the other hand, non-linear dimensionality reduction techniques tend to perform better on natural datasets. So, it can be said that the dimensionality reduction techniques performs differently on different datasets.

Non-linear dimensionality reduction techniques are of three main types. The first type of techniques are concerned with the global properties of the original data whereas the second type of techniques are more concerned about the local properties of the original data. These two type of techniques try to preserve the respective global and local properties of the original high-dimensional data in the low-dimensional representation as well. On the other hand, the third type of techniques deal with both the local and global features of the original data and try to retain all those properties in low-dimensional space.

Some of the non-linear dimensionality reduction techniques are discussed is the following sections.

### 2.7.1 Multi-Dimensional Scaling

There are several non-linear techniques that map the high-dimensional data into a low-dimensional space while nearly preserving the pairwise distances between the datapoints. These techniques are collectively known as Multi-Dimensional Scaling (MDS). This mapping is done based on optimizing a stress function. This function is the error which is measured as the difference between the pairwise distances in the low-dimensional and high-dimensional representation of the data.

The error is calculated by the raw stress function given below:

$$\phi(Y) = \sum_{ij} (\|x_i - x_j\| - \|y_i - y_j\|)^2 \tag{2.4}$$

where $x_i$, $x_j$ are the two high-dimensional datapoints and $\|x_i - x_j\|$ is the Euclidean distance between them. Similarly, $y_i$, $y_j$ are the two low-dimensional datapoints and $\|y_i - y_j\|$ is the Euclidean distance between them.

Some methods based on MDS are Sammon's Mapping and Isomap. This two methods are discussed in the following sections.

### 2.7.2 Sammon's Mapping

Sammon mapping algorithm is a member of the MDS family which maps high-dimensional data to a low-dimensional space while preserving the inter-point distances in low-dimensional projection as it was in the high-dimensional space. It is a type of MDS with a different stress function. John W. Sammon, Jr. (1969) proposed this method. In this method, the mapping cannot be represented as a linear combination of the original variables. Thus, it is not a linear approach. So, sometimes it becomes difficult to use this technique for classification applications.

The cost function for Sammon's mapping is given below.

$$\phi(Y) = \frac{1}{\sum_{ij} \|x_i - x_j\|} \sum_{i \neq j} \frac{(\|x_i - x_j\| - \|y_i - y_j\|)^2}{\|x_i - x_j\|} \tag{2.5}$$

The difference between raw stress function and Sammon cost function is that the Sammon cost function emphasizes more on retaining distances that were originally small. It acts as a weightage to the significance of a datapoint based on its distance.

There are many methods for minimizing the cost function. They are the Eigen decomposition of a pairwise dissimilarity matrix, gradient descent or some iterative methods. The number of iterations needed are determined experimentally. The convergence of the solutions are not always guaranteed.

The Sammon mapping is one of the most successful non-linear metric MDS method. There are some studies on improving on the algorithm while the stress function is somewhat overlooked. Bregman divergence has been used to improve the performance of the Sammon mapping.

### 2.7.3 Isomap

Isometric feature mapping, or Isomap is a combination of the Floyd–Warshall algorithm with classical MDS. This method was proposed by Joshua B. Tenenbaum, Vin de Silva, John C. Langford in the year 2000. MDS computes the pair-wise distances between all datapoints and based on these distances, it computes the position for each point. But MDS only considers the Euclidean distance between the datapoints. As a result, the global geometry of the datapoints may not be captured properly. In case of high-dimensional data, two datapoints may lie at a larger distance along the manifolds but they may be close to one another if the typical interpoint distance is considered. Isomap handles this problem by retaining the pairwise geodesic distances between those datapoints. Distance between two points along the manifold is called the geodesic distance. Isomap takes help of the Floyd–Warshall algorithm to compute the pairwise distances between every datapoints based on the assumption that only the pairwise distances between

neighbouring points are known. As a result, the geodesic distances between all possible pair of datapoints are obtained. Based on these distance informations, Isomap implement classical MDS to perform the dimensionality reduction on all the datapoints. The details of the Floyd–Warshall algorithm is discussed in the Appendix.

The Isomap algorithm has three major steps which are discussed below.

**Algorithm**

1. **Neighbourhood Graph Construction**

   This step determines which points are neighbours on the manifold $M$. The distances $d_{ij}^x$ between pairs of points $i$ and $j$ in the high-dimensional space are computed. This two points are connected if the distance $d_{ij}^x$ is less than some value, say $\epsilon$ or the points are among $k$ nearest neighbours of each other. Based on these neighbourhood relations, the neighbourhood graph is constructed where the nodes represents datapoints whereas the edges represents the weight $d_{ij}^x$ between neighbouring points $i$ and $j$.

2. **Shortest Path Computation**

   In this step, Isomap computes the shortest path distances $d_{ij}^g$ between every pair of points in the graph $G$ by using the Floyd-Warshall algorithm. Based on these $d_{ij}^g$ values, the geodesic distances $d_{ij}^m$ between all pairs of points on the manifold $M$ are estimated.

3. **Low-dimensional Embedding**

   This is the final step of the Isomap algorithm. In this step, Isomap applies classical MDS to the graph distances matrix $D_G$, whose element at $(i, j)$-th position is $d_{ij}^g$, to construct the low-dimensional embedding $Y$ of the data

in a $d$-dimensional Euclidean space. $Y$ preserves the manifold's estimated intrinsic geometry.

Let $y_i$ be the coordinate vector of point $i$ in the embedding $Y$. They are chosen to minimize the cost function

$$\phi(Y) = \|\tau(D_G) - \tau(D_Y)\|_{L^2} \tag{2.6}$$

where $D_Y$ denotes the Euclidean distance matrix whose elements are represented as $d_{ij}^y = \|y_i - y_j\|$ i.e. the Euclidean distance between points $y_i$ and $y_j$. The $\tau$ operator convert distances to inner products, which uniquely characterize the geometry of the data, thus supporting efficient optimization. The global minimum of the cost function is achieved by setting the coordinates $y_i$ to be the top $d$ eigenvectors of the matrix $\tau(D_G)$.

There are four major weaknesses of Isomap technique. Firstly, it is topologically unstable i.e. it may do errors while constructing the connections in the neighbourhood graph $G$. This errors can severely affect the performance of Isomap. Secondly, it may suffer from 'holes' in the manifold. Thirdly, it may suffer from short-circuiting which causes problem in case of manifold unfolding. Lastly, in case of convex manifold, it may fail.

### 2.7.4 Locally Linear Embedding

Locally Linear Embedding (LLE) is a local technique for dimensionality reduction that attempts to preserve local properties of the original data in the low dimensional representation. LLE is less sensitive to short-circuiting than Isomap, because only a small number of properties are affected if short-circuiting occurs. LLE preserves the local properties of the manifolds. LLE computes the $K$-nearest neighbours of every datapoint. Based on these neighbouring points, LLE represents

a datapoint as an approximate linear combination of its k-nearest neighbours. This linear combination gives a weighted combination of the neighbouring points based on the relative distance of the datapoint and its neighbours. The low-dimensional embedding of the datapoints is obtained by nearly preserving this reconstruction weights in the linear combinations.

This Locally Linear Embedding (LLE) method is discussed in detail in the later sections.

### 2.7.5 Local Tangent Space Alignment

Local Tangent Space Alignment (LTSA) is a non-linear dimensionality reduction technique. It follows the principle of representing each datapoints using the local tangent space based on their local properties. The basic assumption is that when the manifold is accurately unfolded, it will automatically align the tangent hyperplanes of the manifold.

There are two linear mappings, one from a high-dimensional datapoint to its local tangent space and another one from the corresponding low-dimensional datapoint to the same local tangent space. By aligning these two mappings, LTSA construct the low-dimensional embedding of the local tangent space of this manifold. LTSA preserves the neighbourhood information i.e. inter-point distance between neighbouring points of the high-dimensional space in the low-dimensional embedding as much as possible.

There are three fundamental steps of the LTSA algorithm for obtaining the low dimensional embeddings of the high-dimensional dataset. The first step is computing the $k$-nearest neighbours for each of the datapoints, the second step is constructing the tangent space at every point based on these neighbourhood informations and then the third step is optimizing the error function for aligning the tangent spaces.

The algorithm of LTSA is discussed below.

**Algorithm**

1. **Setting neighbourhoods**

   For each of the datapoints $x_i$, the $k$ nearest neighbours $x_{ij}$ are computed based on the distance between the points. This distance is generally considered to be Euclidean distance.

2. **Extracting local coordinates**

   For each of the datapoints $x_i$ and its $k$ neighbours $x_{ik}$, PCA is applied. From the result of PCA, we obtain an optimal linear fitting to the sample points of the neighbourhood. It is done by minimizing the following error

   $$\sum_{j=1}^{k_i} \left\| x_{ij} - x_i - Q_i \theta_j^{(i)} \right\|^2 \tag{2.7}$$

   where $x_{ij}$ is the coordinate of the $j$-th neighbour of datapoint $x_i$
   $Q_i$ is the tangent span of datapoint $x_i$
   $\theta_j^{(i)}$ is the local coordinate of the datapoints $x_i$ in tangent space
   $\|a\|$ represents the Euclidean norm of vector $a$

3. **Aligning local coordinates**

   We obtain $N$ sets of aligned local co-ordinates $\theta_i$. These local co-ordinates are aligned to get the global co-ordinates $T_i$. It is done by minimizing the global error

   $$\sum_{i=1}^{N} \left\| T_{ij} - c_i - L_i \theta_j^{(i)} \right\|^2 \tag{2.8}$$

   where $T$ is the set of all global co-ordinates and $\phi$ is the semidefinite matrix represented as

   $$\phi = \sum_{i=1}^{N} \frac{1}{k_i} S_i \phi_i S_i^T \tag{2.9}$$

### 2.7.6 Self-Organizing Map

A self-organizing map (SOM) or Kohonen map is a type of artificial neural network (ANN). It is also known as Kohonen map, named after its introducer Teuvo Kohonen of Finland (1980). It is an unsupervised learning method that produces a 'map' from high dimensional input space to discrete, low-dimensional representation. SOM uses a neighbourhood function to retain the structural properties of the input space which makes it different from the other ANN's. So it helps in low-dimensional data visualization. The Kohonen net is based on the working principle of a biological neural network and morphogenesis model.

There are two working modes in SOM, the training mode and the mapping mode. In training mode, SOM constructs the map using training data, a method called vector quantization. An internal model is developed in the training mode. So based on this internal model, SOM can classifies a new input datapoint in the mapping mode.

Nodes or neurons are the building component of a SOM. Each node is associated with a weight vector having same dimension as the input and a position in the map space. While mapping, SOM finds the node with the closest (least distance metric) weight vector to the data space vector. Initially, the weights of the neurons are small random values. But in that case, learning is slow. A faster approach is to select a better approximation of initial weights. This is done by choosing values sampled evenly from the subspace spanned by the two largest principal component eigenvectors.

In case of SOM's with a small number of nodes, it behave similar to $K$-means, while SOM's with large number of nodes rearrange data in a way that is fundamentally topological in nature. Large SOM's display emergent properties. In maps consisting of thousands of nodes, it is possible to perform cluster operations on the map itself.

### 2.7.7 Autoencoder

An autoencoder is a feed-forward neural network. It optimizes the identity function. It is done by minimizing the mean of squared error between the input and the output of the network. In best case scenario, it will be 0. It provides a mapping from a vector of values to the same vector. In case of dimensionality reduction, odd number of hidden layers are present. One of the hidden layer contains small number of network units. The network has two halves. The first half maps vectors from high to low-dimensional space and the second half performs the opposite action. $D$ nodes are present as input and output while the middle hidden layer has $d$ nodes.

The neural network is trained on the data-points $x_i$ having dimension $D$ to construct a mapping to the middle hidden layer having dimension $d$ such that in retains the structural information as much as possible. From this hidden middle layer, the low-dimensional representations $y_i$ having dimension $D$ is obtained. In case of linear mapping, linear activation functions are used while for non-linear mapping, sigmoid activation functions are used.

The main drawback of multilayer autoencoder is that its backpropagation method converges slowly. This occurs because of high number of connections. Another problem is that this convergence may reach to a local minima instead of a global one. These drawbacks can be overcomed by using a learning procedure. In the first step, the recognition layers of the network are trained one-by-one using Restricted Boltzmann Machines (RBM) having binary and stochastic nodes. The RBM's minimizes the contrastive divergence based on some unsupervised learning. Secondly, based on the inverse of the trained recognition layers, the reconstruction layers are formed. This process is called autoencoder unrolling. This reconstruction layers are fine-tuned using backpropagation.

### 2.7.8 Laplacian Eigenmap

The intrinsic geometry of the data are not taken into account by the linear dimensionality reduction techniques like PCA. On the other hand, Laplacian Eigenmap is a local features preserving method using spectral techniques for non-linear dimensionality reduction. The local features are based on the pairwise distances between neighbouring datapoints. The basic assumption is that the data lies in a low-dimensional manifold in a high-dimensional space. It represents high-dimensional data in low-dimension by preserving the local manifold properties.

Laplacian Eigenmaps constructs a graph based on the geometry of the dataset i.e. neighbourhood information of the dataset. Each node represents a datapoint and the edges connect the nearest neighbouring nodes. This graph is a discrete approximation of the low-dimensional manifold in the high-dimensional space. Similar to LLE, it constructs a low-dimensional embedding based on the objective of minimizing the distances between a datapoint and its $k$ nearest neighbours. It computes the weighted distance measure between the datapoints that is directly proportional to the closeness of the points in the low-dimensional space. Distance of the datapoints from its closest neighbour contributes more to the cost function than that of its second nearest neighbour. The goal is to minimize this cost function, ensuring that neighbouring points on the manifold are mapped as neighbouring points in the low-dimensional space also. Using spectral graph theory, this minimization is done which is an Eigen problem. Under mild conditions, Laplace-Beltrami operator has a countable spectrum that is a basis for square integrable functions on the manifold, for which the Eigen functions of the Laplace-Beltrami operator based on the manifold act as the embedding dimensions. With the number of points approximating to infinity, the graph Laplacian matrix converges to the Laplace-Beltrami operator with some non-restrictive assumptions. The main disadvantage of this technique is that it cannot embed test datapoints.

The algorithm of Laplacian Eigenmap is discussed below.

**Algorithm**

1. The neighbourhood graph $G$ is constructed where each nodes represents each datapoint and the nearest $k$ neighbours of each datapoints are connected by edges.

2. The weight of each edge is computed by using the Gaussian kernel function. The weight matrix $W$ is computed whose element $W_{ij}$ is the weight of the edge between nodes $i$ and $j$.

3. The optimization cost function needs to be minimized. The function is given below

$$\phi(Y) = \sum_{ij}(y_i - y_j)^2 w_{ij} \tag{2.10}$$

where $y_i$, $y_j$ are the two points in low dimensional space and $w_{ij}$ is the weight between them as computed in the $W$ matrix.

4. The above cost function can be reformulated as

$$\phi(Y) = \sum_{ij}(y_i - y_j)^2 w_{ij} = 2Y^T LY \tag{2.11}$$

where $L$ is the graph Laplacian of $W$ computed as $L = M–W$
$M$ is the diagonal matrix of $W$ having row sum as the diagonal entry i.e.

$$m_{ii} = \sum_j w_{ij} \tag{2.12}$$

5. This cost function boils down to the following generalised eigenvalue problem

$$Lv = \lambda Mv \tag{2.13}$$

The $d$ eigenvectors $v_i$ corresponding to the smallest non-zero eigenvalues form the low-dimensional data representation $Y$

### 2.7.9 Diffusion Map

Diffusion map is based on the relationship between diffusion and a random walk. The idea of diffusion came from the field of dynamical systems while the idea of random walk came from Markov chain. Diffusion map defines a Markov random walk on the connectivity graph of the datapoints. Based on the number of timesteps required for a random walk from one point to another, neighbourhood information of the datapoints are calculated. Based on these neighbourhood information, the diffusion distances are calculated. The pairwise diffusion distances between the datapoints in high-dimensional space are nearly preserved in the low-dimensional representation as well.

The algorithm of diffusion map is discussed below.

**Algorithm**

1. Construct the graph $G$ based on the dataset. Each node represents a datapoint while the weights of the edges between two nodes are the result of the Gaussian Kernel function between them. The weights are calculated as

$$w_{ij} = e^{-\frac{\left\|x_i - x_j\right\|^2}{2\sigma^2}} \tag{2.14}$$

2. The weight matrix $W$ computed, is normalized so that the new matrix $W_N$ is stochastic i.e. each row sum is 1. The normalization is performed by using the following equation

$$P_{ij}^{(1)} = \frac{w_{ij}}{\sum_k w_{ik}} \tag{2.15}$$

3. The forward transition probability matrix of a dynamic process is defined by the Markov matrix $W_N$. So the probability that in a single timestamp, a point goes from one datapoint to another is represented as the matrix $P^{(1)}$.

4. For $t$ timesteps, the forward probability matrix $P^{(t)}$ is given by $(P^1)^t$. The equation for diffusion distance based on the random walk forward probabilities $p_{ij}^t$ is given below.

$$D^{(t)}(x_i, x_j) = \sqrt{\sum_k \frac{(p_{ik}^{(t)} - p_{ij}^{(t)})^2}{\psi(x_k)^0}} \tag{2.16}$$

5. From the spectral theory on the random walk, the diffusion distance in high-dimensional space is preserved in the low-dimensional representation $Y$. The $d$ non-trivial principal eigenvectors are obtained by solving the eigenproblem

$$P_v^{(t)} = \lambda v \tag{2.17}$$

6. The low-dimensional data representation is given by

$$Y = \{\lambda_2 v_2, ..., \lambda_{d+1} v_{d+1}\} \tag{2.18}$$

where $\lambda_i$ are the eigenvalues and $v_i$ are their corresponding eigenvalues.

### 2.7.10 Maximum Variance Unfolding

Maximum Variance Unfolding (MVU), introduced by Saul et. al., is an effective heuristic method for dimensionality reduction. It produces a low-dimensional representation of the data by maximizing the variance of their embeddings while preserving the local distances of the original data. This method is based on a simple heuristic: maximizing the overall variance of the embedding while preserving the local distances between neighbouring observations. The weakness of MVU is that it may suffer from short-circuiting that hampers proper manifold unfolding.

MVU computes a neighbourhood graph based on the distance between the datapoints. It preserves the pairwise distances in the resulting graph. MVU tries to 'unfold' data manifolds 'explicitly'. The Euclidean distances between the

datapoints are maximized keeping neighbourhood graph distances intact. Semidefinite programming can be used in solving this optimization problem.

The algorithm of MVU is discussed below.

**Algorithm**

1. Construct the neighbourhood graph $G$ where each node represents a datapoint and each datapoint is connected with its nearest neighbouring datapoint by an edge.

2. It tries to maximize the Euclidean distance between the nearest neighbouring points in the low dimensional space with the restriction that distance between the neighbouring points are preserved. The optimization function is written as

$$Maximize \sum_{ij} \|y_i - y_j\|^2 \tag{2.19}$$

satisfying the condition that

$$\|y_i - y_j\|^2 = \|x_i - x_j\|^2 \ \forall \ (i,j) \ \epsilon \ G \tag{2.20}$$

3. This optimization problem boils down to a Semidefinite Programming problem (SDP). Let $M$ be a matrix having elements $m_{ij}$ representing the inner product of the low-dimensional dataset $Y$. The SDP is defined as follows. Maximize $trace(M)$ subject to:

   (a) $m_{ii} - 2m_{ij} + m_{ij} = \|x_i - x_j\|^2 \ \forall \ (i,j) \ \epsilon \ G$

   (b) $\sum_{ij} m_{ij} = 0$

   (c) $M \geq 0$

4. The $M$ matrix is obtained as the solution of the SDP. The low-dimensional data representation $Y$ is obtained from the $M$ matrix by performing a Singular Value Decomposition (SVD).

### 2.7.11 Kernel PCA

Kernel Principal Component Analysis (KPCA) is a modification on classical Principal Component Analysis (PCA). It is a combination of linear PCA and kernel trick i.e. using kernel function.

As Kernel PCA is a kernel-based method, the mapping performed by Kernel PCA relies on the choice of the kernel function $\kappa$. Kernel functions can be linear kernel, polynomial kernel or Gaussian kernel. In case of linear kernel, the Kernel PCA boils down to classical linear PCA. KPCA has an internal model. As a result, it can embed new test datapoints based on the information obtained from the embeddings of the train datapoints. In traditional PCA covariance matrix is used to compute principal eigenvectors whereas in Kernel PCA the kernel matrix itself is used. The kernel matrix is the inproduct of the datapoints in the high-dimensional space. It is constructed using the kernel function. PCA is applied on this kernel matrix and as the output, we get the non-linear mappings.

The algorithm of Kernel PCA is discussed below.

**Algorithm**

1. The kernel matrix $K$ of the datapoints $x_i$ is computed. The elements of the kernel matrix $K$ is given by

$$k_{ij} = \kappa(x_i, x_j) \tag{2.21}$$

   where '$\kappa$' is the kernel function used in this method

2. The kernel matrix $K$ is then centered by subtracting the means of the features. The new matrix is constructed by using the following formula

$$k_{ij} = k_{ij} - \frac{1}{n}\sum_{j} k_{ij} \tag{2.22}$$

3. The principal $d$ eigenvectors $v_i$ of the centered kernel matrix are computed.

4. The eigenvectors $a_i$ of the covariance matrix of the high-dimensional space is computed from the eigenvectors $v_i$ and their corresponding eigenvalues $\lambda_i$ of the centered kernel matrix by using the following relation

$$a_i = \frac{1}{\sqrt{\lambda_i}} X v_i \tag{2.23}$$

5. The low-dimensional data representation is obtained by projecting the data onto the eigenvectors of the covariance matrix $A$ having elements $a_i$. The low-dimensional projection $Y$ is computed by the following relation

$$y_i = \{\sum_{j=1}^{n} a_j^1 \kappa(x_i, x_j), ..., \sum_{j=1}^{n} a_j^d \kappa(x_i, x_j)\} \tag{2.24}$$

There are some disadvantages of kernel PCA. First of all, KPCA may not give desired results in some problems while using standard kernels as obtaining a good kernel is not trivial for a specific problem. Secondly, it may be difficult to handle the kernel matrix as its size is proportional to the square of the size of the dataset. In case of high number of sample datapoints, the kernel matrix becomes large sized. So it gets difficult to handle such a huge kernel matrix.

### 2.7.12 Locally Linear Coordination

Locally Linear Coordination (LLC) is a combination of both local and global property preserving techniques. At first, LLC computes a number of locally linear models. Based on these models, LLC computes the linear models and they are aligned globally. Expectation Maximization (EM) algorithm is used to compute different local linear models of the dataset. Then the local linear models are aligned to construct the low-dimensional embedding. This embedding is done by using a variant of LLE.

The algorithm of LLC is discussed below.

**Algorithm**

1. Models are constructed from a Mixture of $m$ Factor Analyzers (MoFA) using the EM algorithm or a Mixture of Probabilistic PCA models (MoPPCA).

2. The output of these models are datapoints represented as $z_{ij}$. Their corresponding responsibilities, represented as $r_{ij}$ where $j$ $\varepsilon\{$ 1, 2, . . . , m $\}$ for each datapoints $x_i$. $r_{ij}$ represents the significance of the datapoint $x_i$ corresponding to the model $j$, which is stochastic i.e. satisfying the condition

$$r_{ij} = 1 \tag{2.25}$$

3. The responsibility-weighted matrix $S$ is constructed from the local models and the corresponding responsibilities. The elements of $S$ are computed as

$$s_{ij} = r_{ij} z_{ij} \tag{2.26}$$

4. Let $W$ be the weight matrix which is computed by using the same method as used in LLE. Then the matrix $M$ is represented as

$$M = (I - W)^T (I - W) \tag{2.27}$$

5. LLC aligns the local models by solving the generalized eigenproblem

$$Av = \lambda Bv \tag{2.28}$$

where $A$ is the inproduct of $M^T$ and $B$ is the inproduct of $S$.

6. The $d$ principle eigenvectors of the solution provides a mapping from high-dimensional space to low-dimensional space. Let $V$ be the matrix of $d$ principle eigenvectors. The low dimensional datapoints are given by the following equation

$$Y = UV \tag{2.29}$$

# Chapter 3

# Locally Linear Embedding

## 3.1 General Idea

Locally-Linear Embedding (LLE) was proposed by Yang (2006). It was introduced almost concurrently with Isomap. The main advantage of LLE over Isomap is faster optimization which do not involve local minima. LLE makes best use of the sparse matrix algorithms. It gives better result in tackling many problems.

LLE assumes that the manifold has enough datapoints where every datapoint and its neighbours lie on or close to a locally linear patch. As a result, these datapoints can be approximately represented as a linear combination of its neighbouring datapoints. Based on the distance between the neighbouring datapoints, a weighted linear combination is obtained. The working principle behind LLE is that this linear combination is invariant under linear transformations. So, based on this linear combination, we get a mapping of the high-dimensional datapoints to its corresponding low-dimensional points.

LLE is an unsupervised learning algorithm that computes low-dimensional, neighbourhood-preserving embedding of high-dimensional inputs. LLE maps its high-dimensional input data into a low-dimensional space. It tries to find local

linear patches around each sample point in low-dimensional manifold embedded in high-dimension space. When such manifold exist, LLE performs effectively. LLE learns the global structure of non–linear manifolds by exploiting the local symmetries of the linear reconstructions.

LLE algorithm is mainly based on three major steps. Firstly, finding the nearest neighbours of each datapoints. Secondly, representing each datapoint as an approximate linear combination of its nearest neighbours and computing the reconstruction weights based on it. Finally, constructing the mapping from high-dimensional space to low-dimensional space by preserving the same linear combination of its neighbours.

The main disadvantages of LLE are as follows:

1. LLE cannot handle the non-uniform sample densities efficiently. As various regions differs in sample densities, the weights also drifts accordingly. This cannot be prevented.

2. LLE do not have any internal model, so it cannot embed new datapoints.

3. LLE is sensitive to its control parameter $K$ i.e. number of neighbours which is a fixed value.

4. LLE is very sensitive to noise. Even a small noise would cause failure in deriving low dimensional coordinates.

5. Eigen problem may not always be solvable.

6. LLE cannot ensure that two different data points in the high-dimensional space are embedded at different points in a lower-dimensional space.

## 3.2 Algorithm

1. LLE computes the $K$-nearest neighbours for each of the datapoints.

   LLE computes the barycentric coordinates of a point $X_i$ based on its $K$-nearest neighbours $X_j$. The original datapoint is represented as a weighted linear combination of its neighbours. The weight information is recorded in the weight matrix $W_{ij}$. The reconstruction error is given by the cost function $E(W)$.

   $$E(W) = \sum_i |X_i - \sum_j W_{ij} X_j|^2 \tag{3.1}$$

2. LLE compute the weights for each datapoint, so that particular datapoint can be expressed as a linear combination of its $K$-nearest neighbours.

   The amount of contribution the point $X_j$ in constructing the point $X_i$ is given by $W_{ij}$. Minimization of the cost function is done with two constraints:

   (a) If point $X_j$ is not a neighbour of the point $X_i$, then $W_{ij}$ is 0.

   (b) The rowsum of every row of the weight matrix is 1.

3. It computes the low-dimensional embedding of datapoints based on linear combination of its neighbours as obtained from the high-dimensional datapoints. This is done by using a eigenvector based optimization technique.

   (a) The high-dimensional data points having dimension $D$ are reduced to $d$ where $D >> d$. The weight matrix computed in step 2 is used to embed the datapoints in low-dimensional space. As a result, the mapping is obtained that retains the neighbourhood information. Each datapoint $X_i$ in high-dimension is mapped onto a datapoint $Y_i$ in low-dimension embedding.

(b) The embedding is done by minimizing the cost function which optimize the coordinates. A sparse $N \times N$ Eigenvalue problem is solved to handle the minimization problem. The maximum $d$ non-zero eigenvectors provides the mapping of the datapoints. The cost function, based on the error as measured using Euclidean distance is given below.

$$C(Y) = \sum_i |Y_i - \sum_j W_{ij} Y_j|^2 \tag{3.2}$$

## Complexity

The three stages of LLE algorithm are:

1. **Computing $K$-Nearest Neighbours** : Complexity is $O[Dlog(K)Nlog(N)]$

2. **Weight Matrix Computation** : Complexity is $O[DNK^3]$. A $K \times K$ linear equation is solved for each of the local neighbourhoods.

3. **Partial Eigenvalue Decomposition** : Complexity is $O[DN^2]$. The $d$ largest eigenvalues are selected and their corresponding eigenvectors are used for embedding the high dimensional datapoints.

   The overall complexity of standard LLE is

$$O[Dlog(K)Nlog(N)] + O[DNK^3] + O[DN^2]$$

# Chapter 4

# Proposed Modification

## 4.1 General Idea

The original LLE algorithm remains very effective while embedding any manifold structure. But in case of real datasets, it tend to overlook the geometry of the manifold and thereby various errors are introduced due to wrong distance considered in the least $K$ distances. Our approach holds in heart the basic LLE algorithm but modifies the distance calculation by replacing the Euclidean distance measure with other distance measures like diffusion-distance, commute-time distance and biharmonic distance. This causes the algorithm to take better embedding with lesser errors and standard deviations and also increases the stability of the manifold embedding. Here, stability of the manifold learning is meant by correct embedding of the manifold with greater variation of the manifold or value of $K$. Euclidean distance computes the shortest distance between two points. It is applied in LLE assuming that the $K$ nearest points will be in euclidean space but it may not be true in case of real dataset. So, instead of using shortest path measure, we used other distance measure which are based on the principle of random walk and timestep.

## 4.2 Distance Measures

LLE is based on the principle of embedding high dimensional data into lower dimension. The points in the higher dimension are considered to be in Euclidean space locally. For LLE, the embedding is done based on the distance between the datapoints. In classical LLE, Euclidean distance is used as the distance measure. But Euclidean distance may not always give the desirable results for all datasets.

All distances are not a metric. The distance measure needs to meet certain conditions so that it can me considered as a metric. These conditions are discussed below.

A distance function $d : D \times D \to \mathbb{R}$ must satisfy the following properties:

1. **Symmetry** : $d(x, y) = d(y, x)$
2. **Non-negativity** : $d(x, y) \geq 0$
3. **Identity of indiscernibles** : $d(x, y) = 0 \iff x = y$
4. **Triangle inequality** : $d(x, z) + d(z, y) \geq d(x, y)$

The important properties of distance measures are:

1. **Metric** : Satisfies all the three conditions of distance measure.
2. **Locally Isotropic** : Approximates to geodesic distance when two points are close to each other.
3. **Globally shape-aware** : Gives idea about global shape when two points are far from each other.
4. **Simple computation** : Easy to compute the distance between a pair of points .

Apart from Euclidean distance, there are different distance metric like geodesic distance, diffusion-distance, commute-time distance, biharmonic distance, Maholanobis distance etc. Instead of using Euclidean distance, these distance can be tried out. So we modified the classical LLE by using diffusion-distance, commute-time distance and biharmonic distance in place of Euclidean distance and studied their performance on different datasets. The details of diffusion-distance, commute-time distance and biharmonic distance are given in the next section.

### 4.2.1 Euclidean Distance

Euclidean distance is the measure of distance between two points $x$ and $y$ in Euclidean space. It is calculated as the length of straight line joining points $x$ and $y$. For $n$-dimensional space, Euclidean distance between two points $x(x_1, x_2, \ldots, x_n)$ and $y(y_1, y_2, \ldots, y_n)$ is calculated as:

$$d_E(x, y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + ... + (x_n - y_n)^2} \tag{4.1}$$

Euclidean distance satisfies all the conditions for being a metric. The proof is given below.

**Proof of Euclidean distance to be a Metric**

1. $d_E(x, y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + ... + (x_n - y_n)^2}$
   $d_E(x, y) = \sqrt{(y_1 - x_1)^2 + (y_2 - x_2)^2 + ... + (y_n - x_n)^2} = d_E(y, x)$

2. From the equation of Euclidean distance, it is obvious that the distance is non-negative

3. From the equation of Euclidean distance, it is evident that if $x = y$ , then $d_E(x, y) = 0$ Now if $d_E(x, y) = 0$ , then the all the squared terms must be individually 0 which implies $x = y$

4. As per Cauchy-Schwarz inequality, $|p.q| \leq |p|.|q|$. So,

$$|p+q|^2 = |p|^2 + 2|p.q| + |q|^2 \leq |p|^2 + 2|p|.|q| + |q|^2 = (|p| + |q|)^2$$

Therefore $(|p+q|^2 \leq |p| + |q|)^2$.

Let $x$, $y$ and $z$ be three points. From the above inequality

$$d_(x,z) = |x-z| = |x-y+y-z| \leq |x-y| + |y-z| = d(x,y) + d(y,z)$$

The advantages of Euclidean distance is that it is easy to compute. It computes the shortest distance between two points. The bottleneck of Euclidean distance is that it cannot compute distance along the surface.

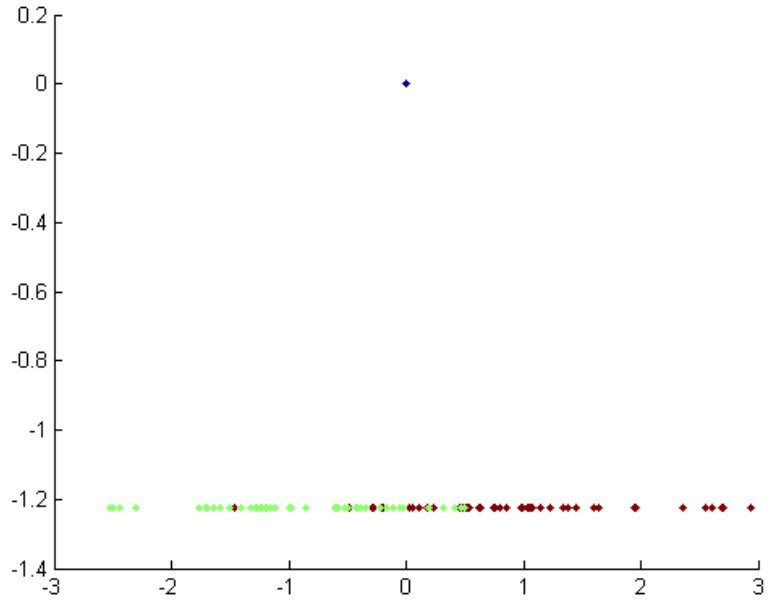


Figure 4.1: Embedding of IRIS dataset using classical LLE (Euclidean distance)

### 4.2.2 Diffusion Distance

The diffusion distance between two points is the similarity of two points at time $t$ in the observation space with the connectivity between them. Diffusion distance is dependent on the parameter $t$. This time parameter determines the time period for which the path lengths are calculated. If the value of $t$ is large, it gives a distance measure capturing global properties more efficiently than local properties. But when $t$ is small, local properties are better captured.

The diffusion distance between two points $x$ and $y$ is calculated as:

$$d_D(x,y)^2 = \sum_{k=1}^{\infty} e^{-2t\lambda_k}(\phi_k(x) - \phi_k(y))^2 \tag{4.2}$$

where $\phi_k$ and $\lambda_k$ are the $k$-th eigenvector and eigenvalue of Laplace-Beltrami operator respectively. The eigenvalues are non-negative and in increasing order.

Diffusion distance satisfies all the conditions for being a metric. The proof is given below.

**Proof of diffusion distance to be a Metric**

1. As diffusion map is an embedding into the Euclidean space, the diffusion distance inherits some of the metric properties i.e. symmetry, non-negativity and the triangle inequality.

2. The only property remaining is the 'identity of indiscernibles'. Here, it is obvious that if $x = y$ then $\phi_k(x) = \phi_k(x) \implies d_B(x,y) = 0$. Now, if $d_B(x,y) = 0$, then $\phi_k(x) = \phi_k(x)$. In this case, $x \neq y$ if and only if $x$ and $y$ have exact same neighbours and proportional weights. Formally, it can be

written as

$$W_{ik} = \alpha W_{jk},\ \ where\ \alpha > 0\ \forall\ k = 1, 2, ..., n \longleftrightarrow d_D(x_i, x_j) = 0; x_i \neq x_j$$

The main advantages of diffusion distance are:

1. It can be computed very quickly.
2. Points closer to one another are highly connected in the graph, so may help in forming clusters.
3. It is sensitive to noise as the distance between two datapoints is dependent on all possible paths of length $t$.

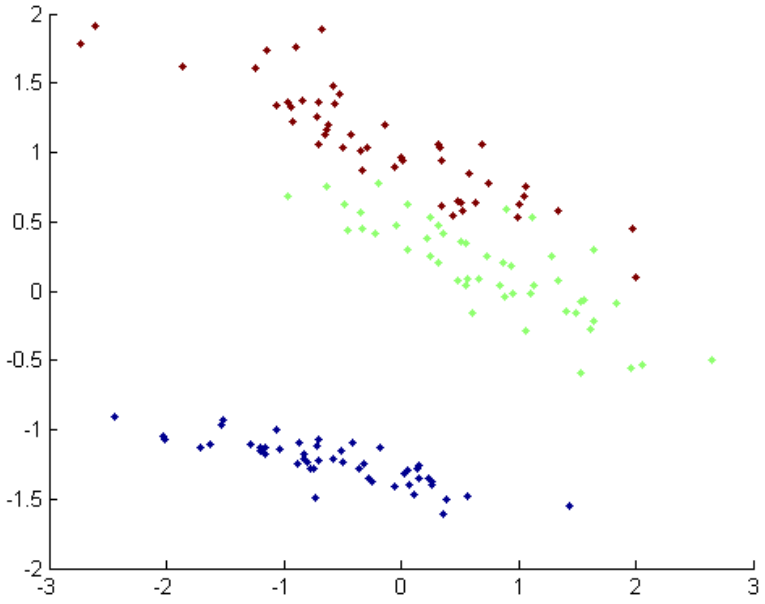


Figure 4.2: Embedding of IRIS dataset using modified LLE (Diffusion distance

### 4.2.3 Commute-time Distance

Commute-time distance was proposed by Fouss et al. (2006), Yen et al. (2007), Qiu and Hancock (2007). It is the distance measure based on the random walk on the similarity graph. Consider two vertices on a graph. The average time taken by a random walker to walk from one vertex to the other and back is represented as the commute-time distance. With the increase in number of paths between two nodes, the lengths of those paths decrease. As a result, the commute-time distance also decreases. Commute-time distance considers the closeness between two nodes based on the connectivity within the graph. So, two datapoints of the same cluster should have very small commute-time distance between them. On the other hand, commute-time distance between two datapoints in different clusters should be relatively large. This property of the commute-time distance makes it well suited for clustering.

The distance calculation for commute-time distance is given below:

$$d_C(x,y)^2 = \sum_{k=1}^{\infty} (\phi_k(x) - \phi_k(y))^2 / \lambda_k \tag{4.3}$$

where $\phi_k$ and $\lambda_k$ are the $k$-th eigenvector and eigenvalue of Laplace-Beltrami operator respectively.

It can also be written as

$$d_C(x,y)^2 = g_C(x,x) + g_C(y,y) - 2g_C(x,y) \tag{4.4}$$

where $g_C(x,y)$ is the Green's function of the Laplacian for the commute-time distance which is written as

$$g_C(x,y)^2 = \sum_{k=1}^{\infty} \phi_k(x)\phi_k(y)/\lambda_k \tag{4.5}$$

Commute-time distance satisfies all the conditions for being a metric. The proof is given below.

**Proof of commute-time distance to be a Metric**

1. It is symmetric as interchanging the roles of $x$ and $y$ does not change the the equation for computing biharmonic distance.

2. From the equation of commute-time distance in the continuous domain, it is clear that the distance is non-negative

3. It is trivial from the equation of commute-time distance that $x = y \implies \phi_k(x) = \phi_y(y) \implies d_B(x, y) = 0$. We have to show that $x = y \impliedby d_B(x, y) = 0$. Now, if $\lambda_k \neq 0$, it can be clearly said that $\phi_k(x) - \phi_y(y) = 0 \implies \phi_k(x) = \phi_y(y)$. Now, $\phi_k$ is an orthonormal basis, so every $L_2$ integrable function $f$ can be written as $\sum_{k=1}^{\infty} \langle f, \phi_k \rangle \phi_k(x)$ for which the equality $f(x) = f(y)$ holds. Since no function distinguishes between $x$ and $y$ it holds $x = y$

4. Let the inequality be

$$\|a - c\| \leq \|a - b\| + \|b - c\| \iff$$

$$\sqrt{\sum_{k=1}^{n}(a_k - c_k)^2} \leq \sqrt{\sum_{k=1}^{n}(a_k - b_k)^2} + \sqrt{\sum_{k=1}^{n}(a_k - b_k)^2} \tag{4.6}$$

   Now, for $n = \infty$ and $a_k = \frac{\phi_k(x)}{\sqrt{\lambda_k}}, b_k = \frac{\phi_k(y)}{\sqrt{\lambda_k}}, c_k = \frac{\phi_k(z)}{\sqrt{\lambda_k}}$ the triangle inequality condition is satisfied.

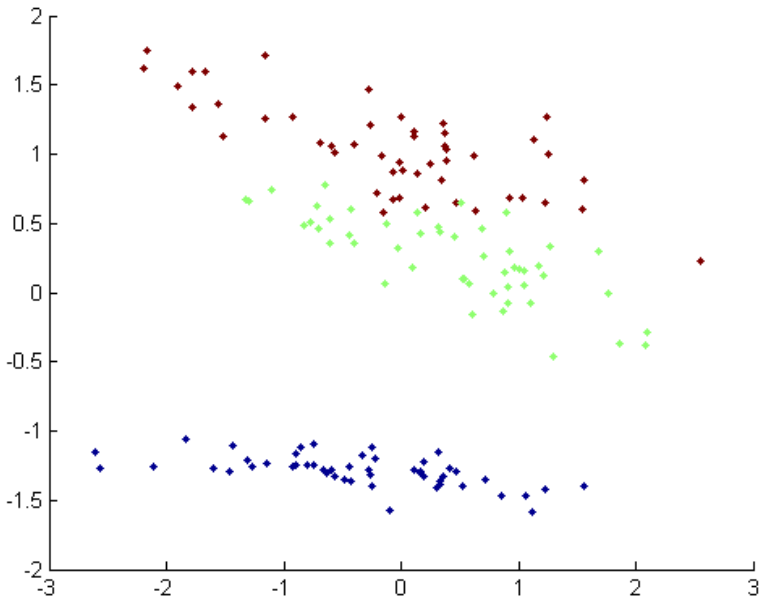


Figure 4.3: Embedding of IRIS dataset using modified LLE (Commute-time Distance)

The main disadvantages of commute time distance are:

1. Harmonic Green's function is singular at the diagonal.

2. It depends on the conformal structure of the surface.

### 4.2.4 Biharmonic Distance

Biharmonic distance was introduced by Y. Lipman, R. M. Rustamov, T. A. Funkhouser (2010). It is based on the idea of taking into account both the local and global properties of the distance. This is done by eigenvalue normalization related to the biharmonic differential equation. The distance calculation is as follows:

#### 4.2.4.1 Continuous domain

The equation for calculating the biharmonic distance in continuous domain is given below.

$$d_B(x,y)^2 = \sum_{k=1}^{\infty} (\phi_k(x) - \phi_k(y))^2 / \lambda_k^2 \tag{4.7}$$

where $\phi_k$ and $\lambda_k$ are the $k$-th eigenvector and eigenvalue of Laplace-Beltrami operator respectively. It can also be written as

$$d_B(x,y)^2 = g_B(x,x) + g_B(y,y) - 2g_B(x,y) \tag{4.8}$$

where $g_B(x,y)$ is the Green's function of the biharmonic operator $\Delta^2$

$$g_B(x,y)^2 = \sum_{k=1}^{\infty} \phi_k(x)\phi_k(y)/\lambda_k^2 \tag{4.9}$$

#### 4.2.4.2 Discrete domain

The equation for calculating the biharmonic distance in discrete domain is given below.

$$d_B(v_i,v_j)^2 = g_d(i,i) + g_d(j,j) - 2g_d(j,j) \tag{4.10}$$

where $g_d(x,y)$ is the discrete Green's function of the Bi-Laplacian. Dicretization is based on discrimination of the Laplace-Beltrami differential operator on meshes

$$K = \sum_{j \epsilon N_1(i)} (cot\alpha_{ij} + cot\beta_{ij})(u_i - u_j)/A_i \tag{4.11}$$

where $A_i$ is the Voronoi area at the $i$-th mesh vertex, $\alpha_{ij}$ and $\alpha_{ij}$ are the angles connecting $i$ and $j$, $A$ is a diagonal matrix where $A_{ii} = A_i$, $l_d$ is the matrix of linear transformation. Now, $l_d = A^{-1}l_c$ where $l_c$ is the conformal discrete Laplacian without the inverse area terms. Finally, $g_d = l_c A^{-1} l_c$

Biharmonic distance satisfies all the conditions for being a metric. The proof is given below.

**Proof of biharmonic distance to be a Metric**

1. It is symmetric as interchanging the roles of $x$ and $y$ does not change the the equation for computing biharmonic distance.

2. From the equation of biharmonic distance in the continuous domain, it is clear that the distance is non-negative.

3. It is trivial from the equation of biharmonic distance that $x = y \implies \phi_k(x) = \phi_y(y) \implies d_B(x, y) = 0$. We have to show that $x = y \impliedby d_B(x, y) = 0$. Now, if $\lambda_k \neq 0$, it can be clearly said that $\phi_k(x) - \phi_y(y) = 0 \implies \phi_k(x) = \phi_y(y)$. Now, $\phi_k$ is an orthonormal basis, so every $L_2$ integrable function $f$ can be written as $\sum_{k=1}^{\infty} \langle f, \phi_k \rangle \phi_k(x)$ for which the equality $f(x) = f(y)$ holds. Since no function distinguishes between $x$ and $y$ it holds $x = y$.

4. Let the inequality be

$$\|a - c\| \leq \|a - b\| + \|b - c\| \iff$$

$$\sqrt{\sum_{k=1}^{n} (a_k - c_k)^2} \leq \sqrt{\sum_{k=1}^{n} (a_k - b_k)^2} + \sqrt{\sum_{k=1}^{n} (a_k - b_k)^2} \tag{4.12}$$

   Now, for $n = \infty$ and $a_k = \frac{\phi_k(x)}{\lambda_k}, b_k = \frac{\phi_k(y)}{\lambda_k}, c_k = \frac{\phi_k(z)}{\lambda_k}$ the triangle inequality condition is satisfied.

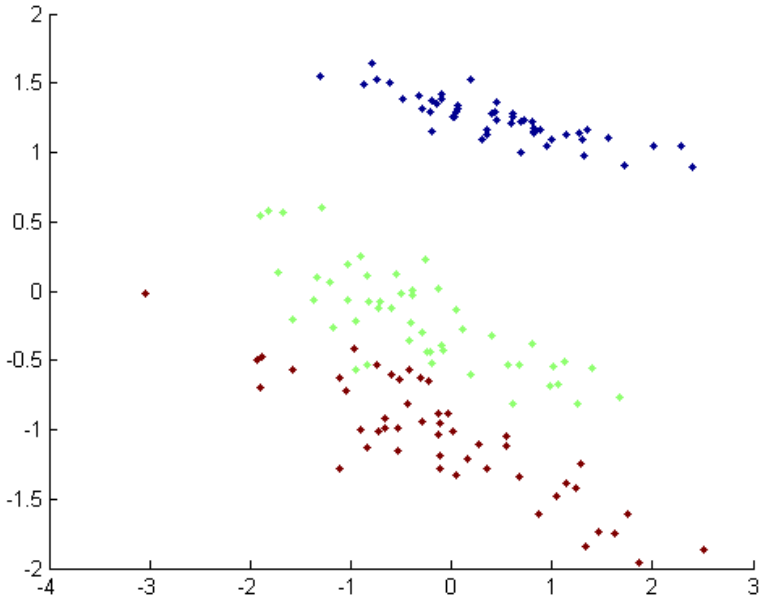


Figure 4.4: Embedding of IRIS dataset using modified LLE (Biharmonic Distance)

# Chapter 5

# Statistical Analysis

Experimental results are an integral part of establishing a theory. In this section, we will describe the experimental set-up, steps of experiments, experimental parameters and datasets. Initially, we will mention the basic experimental setup and discuss them in details as we go progressing ahead. Along with the experiments, the results obtained are reported. The inferences are described along with the results. The experiments are done in two phases. The details of these experiments are discussed in the later sections.

1. Experiment-I

   In this type of experiments, for each dataset, for each valid embedding, the misclassification fractions are computed. Based on this misclassification values, all the testings are done.

2. Experiment-II

   In this type of experiments, for each dataset, the minimum misclassification ratio for each distance-measure methods are recorded. Based on this misclassification values, all the testings are done.

## 5.1 Dataset

For our experiments, we have used 16 datasets in total. 9 of them are selected from UCI machine learning repository while the 'Two-Norm' dataset is obtained from the Delve dataset. Among remaining 6 datasets, 3 are real datasets while rest 3 are derived datasets. The 3 real datasets are MNIST dataset for hand-written numbers, YALE face dataset and AT&T face dataset. Remaining 3 are constructed by applying PCA algorithm on the original MNIST, YALE and AT&T dataset. For these 3 synthetic datasets, the top 10% principal eigenvectors are selected. As a result, in these reduced datasets, the number of attributes is 10% of that of the mother dataset. So, for AT&T and YALE face dataset, the original attribute is 1024 whereas in the reduced dataset, attribute is 103. On the other hand, for MNIST dataset, the original attribute is 784 whereas in the reduced dataset, attribute is 79. The number of samples, number of attributes, and number of classes of these 16 datasets are given in the columns #sample, #attribute and #class of the table-5.1. Each row represents a dataset. The details of these datasets are described in the Appendix.

## 5.2 Experiments

### 5.2.1 Steps of Experiment

Experimentation is the most important part of this study. The goal of these experiments is to compare the performances of Locally Linear Embedding in classification of real-datasets. In LLE, for computing neighbours, four different distance measures are used i.e. Euclidean distance (original LLE), diffusion distance, commute-time distance and biharmonic distance (modified LLE). So we get four different distance-measure methods which will be considered as test methods.

Table 5.1: Description of Dataset

| Dataset | #sample | #attribute | #class |
|---|---|---|---|
| AT&T (face) | 400 | 1024 | 40 |
| Breast_cancer | 457 | 9 | 2 |
| Diabetes | 513 | 8 | 2 |
| Haberman | 205 | 3 | 2 |
| Heart | 181 | 13 | 2 |
| Ionosphere | 235 | 34 | 2 |
| Iris | 150 | 4 | 3 |
| Liver | 231 | 5 | 2 |
| MNIST (number) | 2105 | 784 | 10 |
| Reduced_at&t | 400 | 103 | 40 |
| Reduced_mnist | 2105 | 79 | 10 |
| Reduced_yale | 165 | 103 | 15 |
| Thyroid | 143 | 5 | 2 |
| Two-Norm | 400 | 20 | 2 |
| Wine | 120 | 13 | 2 |
| YALE (face) | 165 | 1024 | 15 |

The three major experimental steps of this study are:

1. **Dimensionality Reduction**

2. **Classification**

3. **Statistical Testing**

The In the dimensionality reduction stage, the high-dimensional datasets are embedded into low-dimensional space by the help of Locally Linear Embedding

technique. After this stage, the low dimensional datasets are classified by using the $k$-Nearest Neighbour classification method. Based on the misclassification values obtained from the previous stage, five non-parametric statistical tests are performed. The results of these statistical tests are analyzed to compare the performance of the test methods. The internal steps and the sequence of experimentations are given in the following flow chart (Figure-5.1).

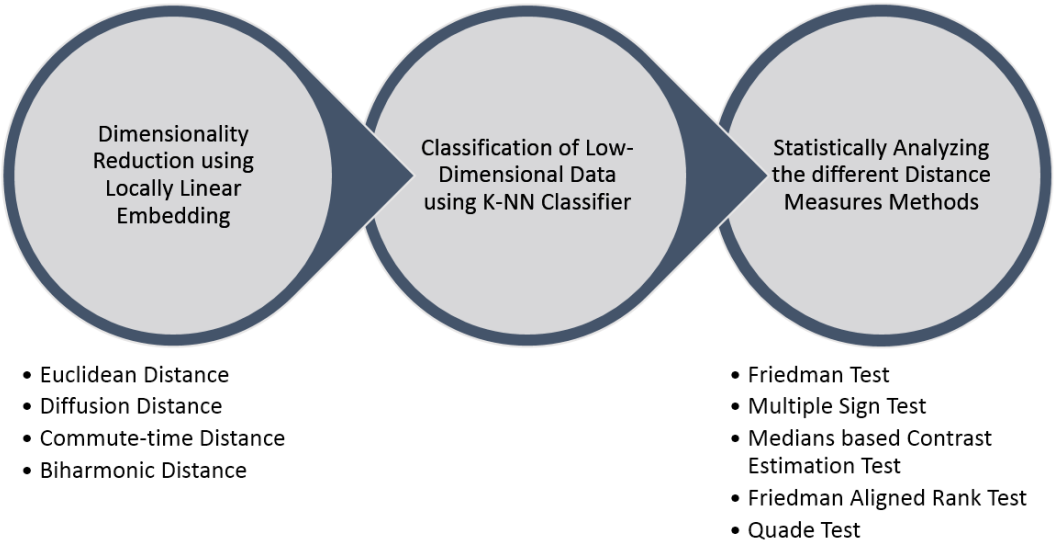


Figure 5.1: Sequence of Experimentations

### 5.2.2 Experimental Setup

In this section, we discuss about the setup of the experiments. The experimental setup describes about how the experiments were carried out and what were the parameters used for experiments and their respective values. The three main stages of the experimentation are dimensionality reduction, classification and statistical testing. The details of the experimental parameters used for these three major steps are described individually in the following sections.

#### 5.2.2.1 LLE

We have selected Locally Linear Embedding (LLE) as the method of dimensionality reduction. The parameters required for this LLE algorithm are $d$ and $K$. Here, $K$ represents the number of points to be considered as nearest neighbours whereas $d$ represents the low dimension to which the high-dimensional data needs to be embedded. For our experiments, we selected 6 different $K$-values and 8 different $d$-values. The $d$-values ranges from 2 to 99 while the $K$-value ranges from 7 to 30. The $d$-values and $K$-values chosen for the experiments are given below in the table-5.2. In each case except 'Breast cancer' dataset, all 6 of the $K$-values are

Table 5.2: Set of $d$-values & $K$-values for LLE

| d-values | 2 | 3 | 15 | 25 | 40 | 55 | 77 | 99 |
|---|---|---|---|---|---|---|---|---|
| K-values | 7 | 12 | 15 | 20 | 25 | 30 | - | - |

considered while all the $d$-values that are not more than the actual dimension of the dataset are taken. Experiments are performed for all valid $(d, k)$ pairs for each datasets. In case of 'Breast cancer' dataset, the last 5 $K$-values are selected. The total number of experiments performed on each of the datasets is product of corresponding number of $d$-values and $K$-values. The details of the set of $d$-values & $K$-values for each dataset is given in the table-5.3

#### 5.2.2.2 $k$-NN Classification

$k$-nearest neighbouring ($k$-NN) is a popular classification technique. The only parameter required in this classification algorithm is $k$. In this method, the $k$ nearest neighbours of the unlabelled test datapoint are computed. By applying the 'majority wins' rule, the majority class label of those neighbouring datapoints is assigned to the test datapoint.

Table 5.3: Set of $d$-values & $K$-values for each dataset used for LLE

| Dataset | #d | #K | #exp |
|---|---|---|---|
| AT&T (face) | 2, 3, 15, 25, 40, 55, 77, 99 | 7, 12, 15, 20, 25, 30 | 48 |
| Breast_cancer | 2, 3 | 12, 15, 20, 25, 30 | 10 |
| Diabetes | 2, 3 | 7, 12, 15, 20, 25, 30 | 12 |
| Haberman | 2, 3 | 7, 12, 15, 20, 25, 30 | 12 |
| Heart | 2, 3 | 7, 12, 15, 20, 25, 30 | 12 |
| Ionosphere | 2, 3, 15, 25 | 7, 12, 15, 20, 25, 30 | 24 |
| Iris | 2, 3 | 7, 12, 15, 20, 25, 30 | 12 |
| Liver | 2, 3 | 7, 12, 15, 20, 25, 30 | 12 |
| MNIST (number) | 2, 3, 15, 25, 40, 55, 77, 99 | 7, 12, 15, 20, 25, 30 | 48 |
| Reduced_at&t | 2, 3, 15, 25, 40, 55, 77, 99 | 7, 12, 15, 20, 25, 30 | 48 |
| Reduced_mnist | 2, 3, 15, 25, 40, 55, 77 | 7, 12, 15, 20, 25, 30 | 42 |
| Reduced_yale | 2, 3, 15, 25, 40, 55, 77, 99 | 7, 12, 15, 20, 25, 30 | 48 |
| Thyroid | 2, 3 | 7, 12, 15, 20, 25, 30 | 12 |
| Two-Norm | 2, 3, 15 | 7, 12, 15, 20, 25, 30 | 18 |
| Wine | 2, 3 | 7, 12, 15, 20, 25, 30 | 12 |
| YALE (face) | 2, 3, 15, 25, 40, 55, 77, 99 | 7, 12, 15, 20, 25, 30 | 48 |

This algorithm is very easy to implement. But the classification results can be sensitive to this $k$-value. It is difficult to decide the value of $k$. Generally, the value of $k$ is selected by various heuristic techniques. For our experiment, value of $k$ is assigned to be 9. We have used 5-fold cross validation and each experiment is run for 5 times, so we get 25 results and those are averaged. These are done for each experiments of each dataset. The misclassification fraction is noted for each experiments of each datasets.

The $k$-NN classification algorithm is given below.

**Algorithm**

1. For a test data-point, compute its distance with every other trained data-points. This distance measure is generally considered to be the Euclidean distance but it can be other distance measures as well.

2. Among all the data-points, select the nearest $k$ neighbouring data points i.e. the $k$ data-points closest to the test data point.

3. Record the class labels of these $k$-nearest data-points.

4. Calculate the count of class labels of these neighbouring data points.

5. The class label having majority count then gets assigned to the test data-point as the class label.

#### 5.2.2.3 Statistical Testing

Here, the tests are done to analyze the performances of the distance measures, their interrelationships or co-dependencies. These tests compares the performances of different algorithms which gives us an idea about which algorithm is performing better. Experiments are categorized into two types. Type-I experiments consider all the valid embedding's of each dataset and compares each of them across all the distance-measure methods for ranking. On the other hand, Type-II experiments consider only the 'minimum' misclassification fraction of each dataset as its representative value and compares each of those values across all the distance-measure methods. Five different statistical testing are performed on these results and the distance-measure methods are analyzed based on these tests and their performances are evaluated.

## 5.3 Results

### 5.3.1 Type-I Experiments

For the Type-I experiment, for each dataset, for each valid embedding, the misclassification fractions are computed. The distance-measures methods are ranked from 1 to $n$ accordingly. The method having least misclassification value performs best and is ranked 1, second best is ranked 2 while the one performing worst is ranked $n$. In this way, for each dataset, ranks are computed. Here, we have 4 distance-measure methods, so they are ranked from 1 to 4. In the table-5.4 is the example of ranking for the MNIST dataset.

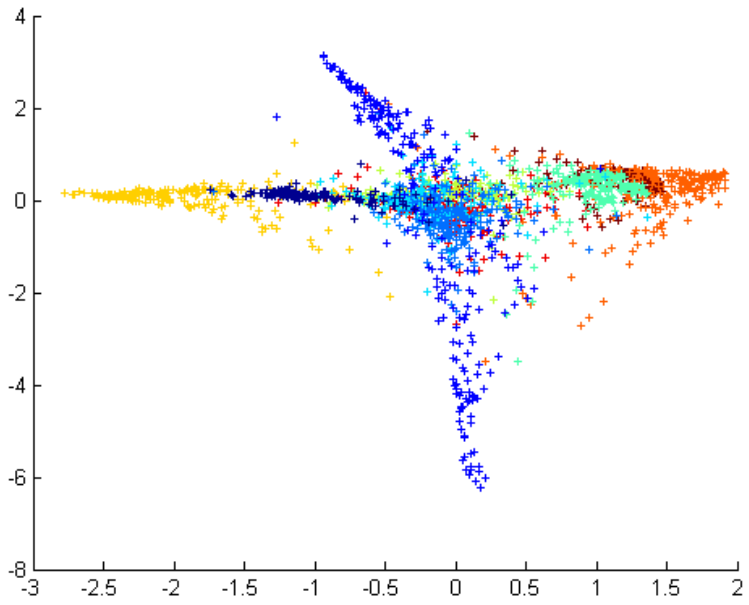


Figure 5.2: Embedding of MNIST dataset using classical LLE (Euclidean distance)

In the table-5.4, row-1, row-2, row-3 and row-4 signifies the number of cases where different distances ranked 1, 2, 3 and 4 respectively. The total is the weighted sum rank of each of the distance-measure methods.

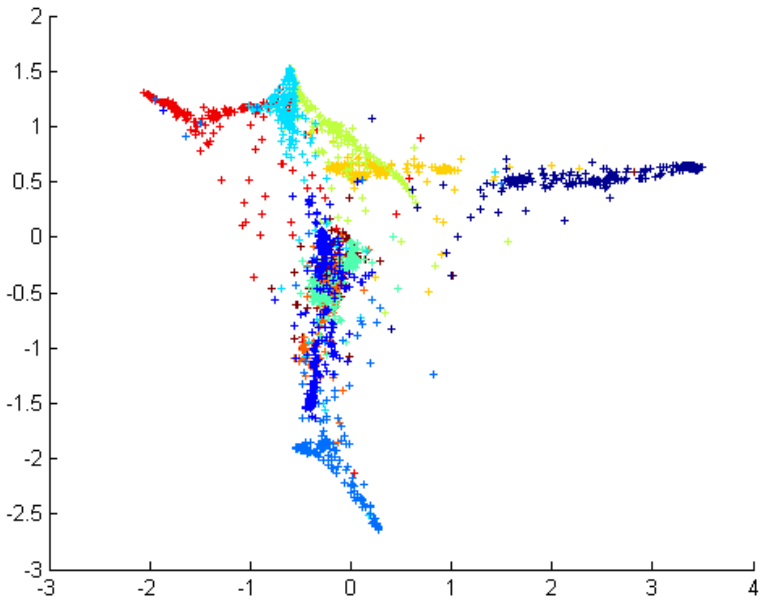


Figure 5.3: Embedding of MNIST dataset using modified LLE (Diffusion distance)

It is calculated as:

$$T_i = \sum_{j=1}^{n} j c_{ij} \tag{5.1}$$

where $c_{ij}$ is the number of $j$-th ranked cases in $i$-th distance-method

Average rank is calculated as

$$A_i = T_i / \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} \tag{5.2}$$

where $n$ is the number of distance-measures methods

From the table-5.4, we get the total rank and the average rank. Based on these average rank, the final ranking of the test methods are done. It is done in increasing order of their corresponding average rank. The distance-measure method with highest average rank is ranked 4 while that having least average rank is ranked 1. In case of a tie, the mean of the rank is given to all the tied test-methods.

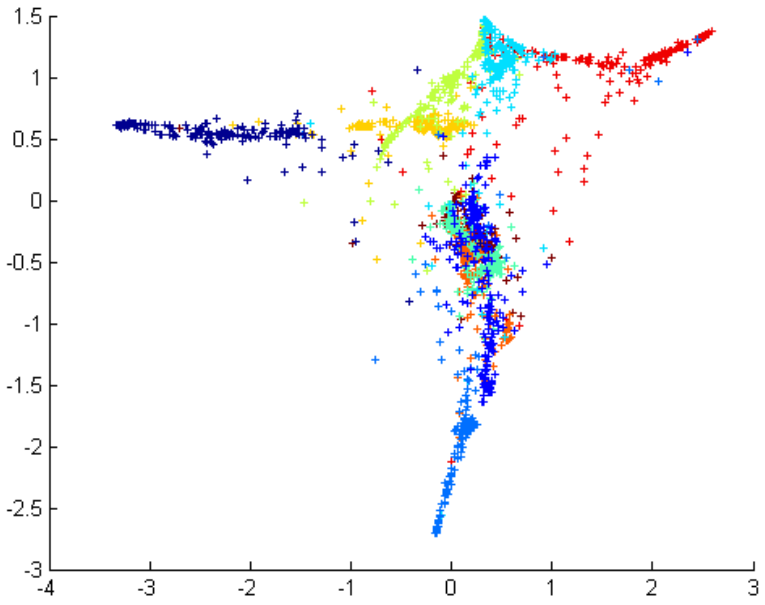


Figure 5.4: Embedding of MNIST dataset using modified LLE (Commute-time distance)

Table 5.4: Rank distribution across different distance measures for all 48 embedding of MNIST dataset

| MNIST | EC | DF | CT | BI |
|---|---|---|---|---|
| 1 | 7 | 8 | 19 | 14 |
| 2 | 3 | 21 | 16 | 8 |
| 3 | 4 | 13 | 11 | 20 |
| 4 | 34 | 6 | 2 | 6 |
| Total ($T_i$) | 161 | 113 | 92 | 114 |
| Average ($A_i$) | 3.354 | 2.354 | 1.917 | 2.375 |
| Rank | 4 | 2 | 1 | 3 |

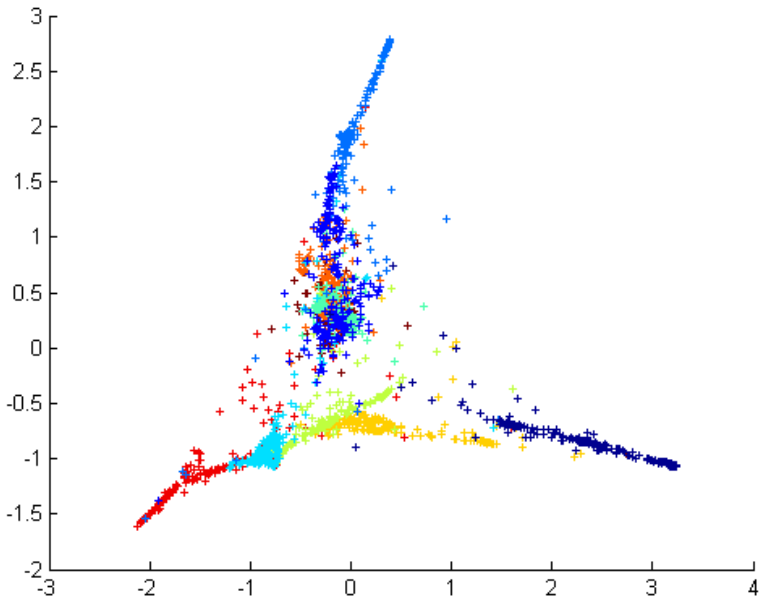


Figure 5.5: Embedding of MNIST dataset using modified LLE (Biharmonic distance)

So, the method having best average performance is ranked 1 while the one having worst average performance is ranked 4. These experiments are done on each of the datasets. The ranks of all the four distance-measure methods for each of the datasets are computed individually. The detailed individual results of all the datasets are given in the Appendix. The individually computed ranks of the different distance measures for different datasets are recorded row-wise in the table-5.5

Now, to evaluate the overall performance of each test methods across all datasets, the combined ranking of all the datasets are needs to be computed. In the table-5.6, row-1, row-2, row-3 and row-4 signifies the number of cases where different distances ranked 1, 2, 3 and 4 respectively. The total is weighted sum rank of each

Table 5.5: Experiment-wise ranking based on Friedman Test

| Dataset | EC | DF | CT | BI |
|---|---|---|---|---|
| AT&T (face) | 1 | 4 | 2 | 3 |
| Breast_cancer | 1 | 4 | 2 | 3 |
| Diabetes | 4 | 3 | 2 | 1 |
| Haberman | 4 | 1 | 2 | 3 |
| Heart | 3 | 2 | 1 | 4 |
| Ionosphere | 3 | 4 | 2 | 1 |
| Iris | 1 | 4 | 2 | 3 |
| Liver | 4 | 2 | 3 | 1 |
| MNIST (number) | 4 | 2 | 1 | 3 |
| Reduced_at&t | 1 | 4 | 2 | 3 |
| Reduced_mnist | 4 | 2 | 3 | 1 |
| Reduced_yale | 1 | 3 | 2 | 4 |
| Thyroid | 3 | 1 | 2 | 4 |
| Two-Norm | 1 | 3 | 2 | 4 |
| Wine | 2 | 3 | 1 | 4 |
| YALE (face) | 1 | 4 | 2 | 3 |

of the distance-measure methods. It is calculated as:

$$T_i = \sum_{j=1}^{n} j c_{ij} \tag{5.3}$$

where $c_{ij}$ is the number of $j$-th ranked cases in $i$-th distance-method
The average ranks for $n$ distance-measure methods are calculated as

$$A_i = T_i / \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} \tag{5.4}$$

The average rank gives the weighted combined rank of each distance-measure methods across all the datasets. The total and average rank are computed in the similar way as it was done for the individual datasets. The detailed results of the average performance of each of the distance-methods across all datasets are given in the table-5.6.

Table 5.6: Summarised ranking table based on Friedman Test

| All Data | EC | DF | CT | BI |
|---|---|---|---|---|
| 1 | 7 | 2 | 3 | 4 |
| 2 | 1 | 4 | 11 | 0 |
| 3 | 3 | 4 | 2 | 7 |
| 4 | 5 | 6 | 0 | 5 |
| Total | 38 | 46 | 31 | 45 |
| Average | 2.375 | 2.875 | 1.9375 | 2.8125 |
| Rank | 2 | 4 | 1 | 3 |

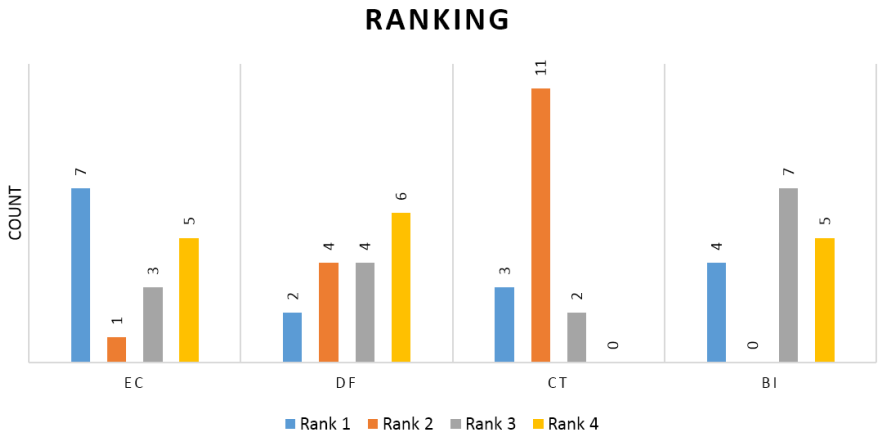


Figure 5.6: Friedman Test result for Experiment-I

Based on the results as represented in the table-5.6, it is observed that among all the four distance-measure algorithms, commute-time distance gives the best result. The average rank for commute-time is least, followed by Euclidean distance, biharmonic distance and diffusion distance. The average rank for commute-time is better than that of second placed Euclidean distance by a considerable margin. On the other hand, the average rank of biharmonic distance and diffusion distance across different datasets are close, so it may be concluded that they have similar performance across these datasets.

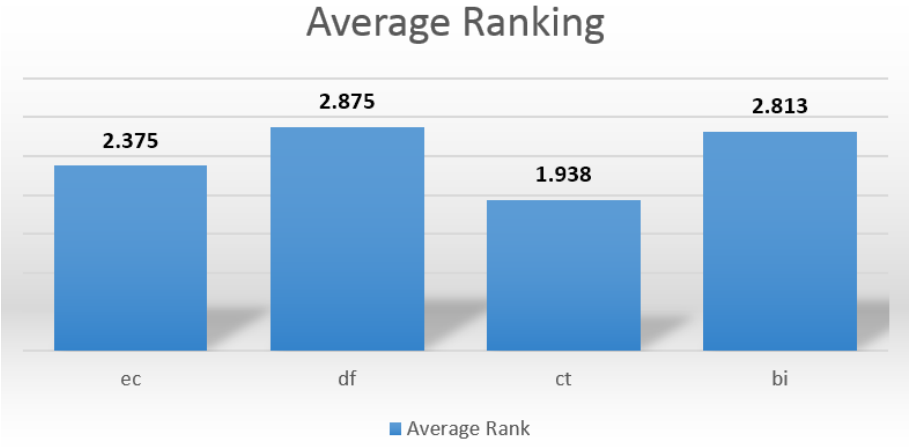


Figure 5.7: Ranking based on Friedman Test for Experiment-I

### 5.3.2 Type-II Experiments

In this type-II experiments, for each dataset, the Minimum Misclassification Ratio (MMR) for each distance-measure methods are recorded. Based on these misclassification values, all the statistical testing are performed and the relative performance of each distance-measure methods are analyzed.

In our statistical analysis, we have performed five statistical tests namely Friedman Ranking test, Multiple-sign test, Median Based Contrast Estimation Test,

Friedman Aligned Rank test and Quade test. For each of these statistical tests, based on the misclassification value for each distance-measure methods for all datasets, the distance-measure methods are ranked. After performing all these tests, we give a comparative study of all the distance-measure methods based on the results of these tests. The table-5.7 gives the misclassification ratio of the different datasets across all distance measures.

Table 5.7: MMR of datasets across different distance measures

| Dataset | EC | DF | CT | BI |
|---|---|---|---|---|
| AT&T (face) | 0.114 | 0.129 | 0.143 | 0.131 |
| Breast_cancer | 0.303 | 0.300 | 0.331 | 0.290 |
| Diabetes | 0.304 | 0.287 | 0.298 | 0.278 |
| Haberman | 0.233 | 0.218 | 0.248 | 0.223 |
| Heart | 0.320 | 0.333 | 0.338 | 0.328 |
| Ionosphere | 0.138 | 0.153 | 0.122 | 0.160 |
| Iris | 0.033 | 0.001 | 0.000 | 0.000 |
| Liver | 0.345 | 0.359 | 0.361 | 0.354 |
| MNIST (number) | 0.088 | 0.086 | 0.086 | 0.073 |
| Reduced_at&t | 0.166 | 0.185 | 0.169 | 0.179 |
| Reduced_mnist | 0.089 | 0.087 | 0.088 | 0.085 |
| Reduced_yale | 0.348 | 0.380 | 0.372 | 0.354 |
| Thyroid | 0.048 | 0.053 | 0.026 | 0.044 |
| Two-Norm | 0.041 | 0.042 | 0.045 | 0.040 |
| Wine | 0.262 | 0.278 | 0.293 | 0.270 |
| YALE (face) | 0.251 | 0.276 | 0.261 | 0.269 |

#### 5.3.2.1 Friedman Rank Test

The Friedman test is a non-parametric statistical test developed by the U.S. economist Milton Friedman. It is similar to the parametric repeated measures two-way analysis of variance, commonly known as ANOVA. The main objective of Friedman test is to detect differences in treatment across multiple test attempts. This procedure involves ranking each sample of results together, then considering the values of ranks by test methods.

The Friedman test is used for one-way repeated measures analysis of variance by ranks. The ranking is done based on the misclassification values of each dataset across all test-methods. Then Friedman Test Statistics is done on these values. Based on these Friedman test results, the relative performance of the distance-measure methods are compared and analyzed. The method of computing the Friedman statistics is given below.

**Method**

1. The $\{x_{ij}\}_{n\times k}$ matrix is given where each row represent each sample of results and each column represent each test methods. The element $x_{ij}$ represents the original result of the $i$-th sample of the $j$-th method. This original result is converted to rank matrix $\{s_{ij}\}_{n\times k}$

2. The ranking of the test-methods is done for each row. The best performing test method gets rank 1, second best gets rank 2 and so on with the last one getting rank $n$. In case of a tie, the mean rank is given to the tied methods. In this way, the rank matrix is populated.

3. Compute

$$r_j = \frac{1}{n}\sum_{i=1}^{n} r_{ij} \tag{5.5}$$

4. Compute

$$r_m = \frac{1}{nk}\sum_{i=1}^{n}\sum_{j=1}^{k} r_{ij} \tag{5.6}$$

5. Compute

$$SS_t = n\sum_{j=1}^{k}(r_j - r_m)^2 \tag{5.7}$$

6. Compute

$$SS_e = \frac{1}{n(k-1)}\sum_{i=1}^{n}\sum_{j=1}^{k} r_{ij} - {r_m}^2 \tag{5.8}$$

7. The test statistics $Q$ is given as

$$Q = \frac{SS_t}{SS_e} \tag{5.9}$$

8. When $n > 10$ or $k > 4$, the probability distribution of $Q$ can be approximated by that of a chi-squared distribution but if $n$ and $k$ are small, the approximation becomes poor.

9. When $n$ or $k$ is large, the Friedman statistics is computed as

$$\chi_F^2 = \frac{12n}{k(k+1)}(\sum_j r_j^2 - \frac{k(k+1)^2}{4}) \tag{5.10}$$

   Here, $n$ is the number of sample results i.e. number of datasets

   $k$ is the number of test methods

   $R_j$ is the average rank of $j$-th method

While when $n$ and $k$ are small, exact critical values are computed. In our experiment, $n = 16$ and $k = 4$ are large, so we will follow the chi-square distribution.

The table-5.8 gives the information about the rank of each test methods for each test samples based on the Friedman test. In our experiment, the total number of test methods is 4, so the ranks ranges from 1 to 4. Rank 1 indicates the best performing test method while rank 4 indicates the worst performing test method. Each row of the table represents each of the 16 datasets while each column represents each distance-measure methods. The total rank is the sum of all ranks for each distance-measure method and average rank is the mean rank of each distance-measure method.

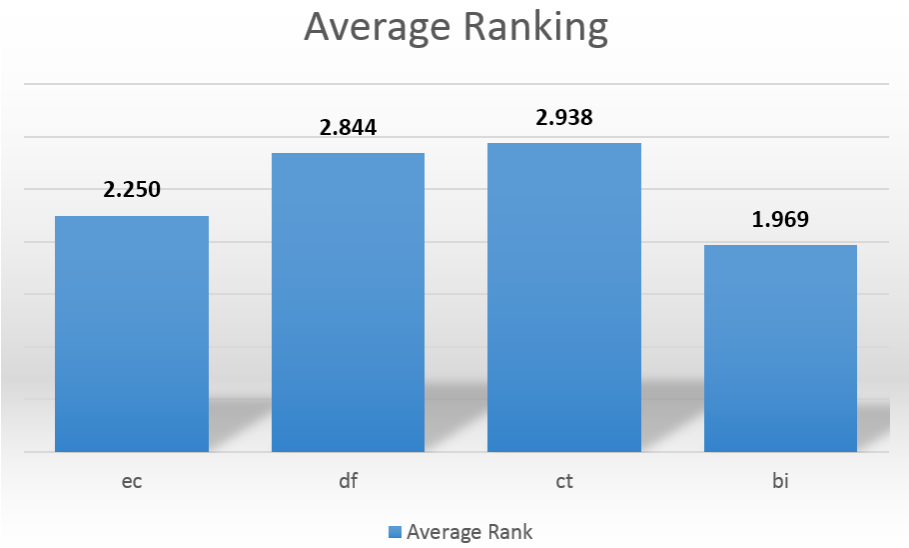


Figure 5.8: Ranking based on Friedman Test for Experiment-II

Under the null hypothesis, all test methods are assumed to be performing similarly. So the rank of each test methods should be similar to one another. The average rank of each test methods across all samples give a good estimate of the performance of the test methods. As per the Friedman ranking system, the table-5.8 is generated. From this table, it is evident that biharmonic distance performs best, followed by Euclidean distance and diffusion distance while

Table 5.8: Rank matrix based on Friedman Test

| Dataset | EC | DF | CT | BI |
|---|---|---|---|---|
| AT&T (face) | 1 | 2 | 4 | 3 |
| Breast_cancer | 3 | 2 | 4 | 1 |
| Diabetes | 4 | 2 | 3 | 1 |
| Haberman | 3 | 1 | 4 | 2 |
| Heart | 1 | 3 | 4 | 2 |
| Ionosphere | 2 | 3 | 1 | 4 |
| Iris | 4 | 3 | 1.5 | 1.5 |
| Liver | 1 | 3 | 4 | 2 |
| MNIST (number) | 4 | 2.5 | 2.5 | 1 |
| Reduced_at&t | 1 | 4 | 2 | 3 |
| Reduced_mnist | 4 | 2 | 3 | 1 |
| Reduced_yale | 1 | 4 | 3 | 2 |
| Thyroid | 3 | 4 | 1 | 2 |
| Two-Norm | 2 | 3 | 4 | 1 |
| Wine | 1 | 3 | 4 | 2 |
| YALE (face) | 1 | 4 | 2 | 3 |
| Total | 36 | 45.5 | 47 | 31.5 |
| Average | 2.250 | 2.844 | 2.938 | 1.969 |
| Rank | 2 | 3 | 4 | 1 |

commute-time distance performs worst. The average rank of biharmonic distance and Euclidean distance are close, so their performances can be said to be are similar across all datasets together. Same conclusion can inferred for diffusion distance and commute-time distance.

#### 5.3.2.2 Multiple-sign Test

In this method, one of the test-method is selected as the control method. With respect to the control method, the performance of the other test methods are evaluated. The null hypothesis is that the control method performs best. The steps of this method are discussed below.

**Method**

1. The $\{x_{ij}\}_{n\times k}$ matrix is given where each row represent each sample of results and each column represent each test methods. The element '$x_{ij}$' represents the original result of the $i$-th sample of the $j$-th method. This original result is converted to signed matrix $\{s_{ij}\}_{n\times k}$

2. The signed matrix $\{s_{ij}\}_{n\times k}$ is computed for each row. This matrix indicates the relative performance of each test-method with the control method. It is computed by subtracting the original value of test-method from the control method of a particular sample result. The sign and the magnitude of the differences are recorded in the table.

3. For each test methods, the total number of positive and negative 'differences' with respect to the control method are counted and noted accordingly in the 'plus' and 'minus' rows of the table.

4. Let the null hypothesis be such that control method performs better than the other test-methods. This null hypothesis will be rejected if the number of minus signs is less than the number of plus signs whereas if the number of minus signs is greater than the number of plus signs, then the null hypothesis will be accepted.

The table-5.9 contains the original misclassification ratio of the different datasets across all test-methods. In the parenthesis, the sign of the difference between values of test methods and control method are recorded. The sign is computed by subtracting the original value of test-method from the control method for each sample result. If the value of test method is greater than that of control method, the sign is '-' and if the value of test method is less than that of control method, then the sign is '+'. But if the value is 0, then the sign will be '='.

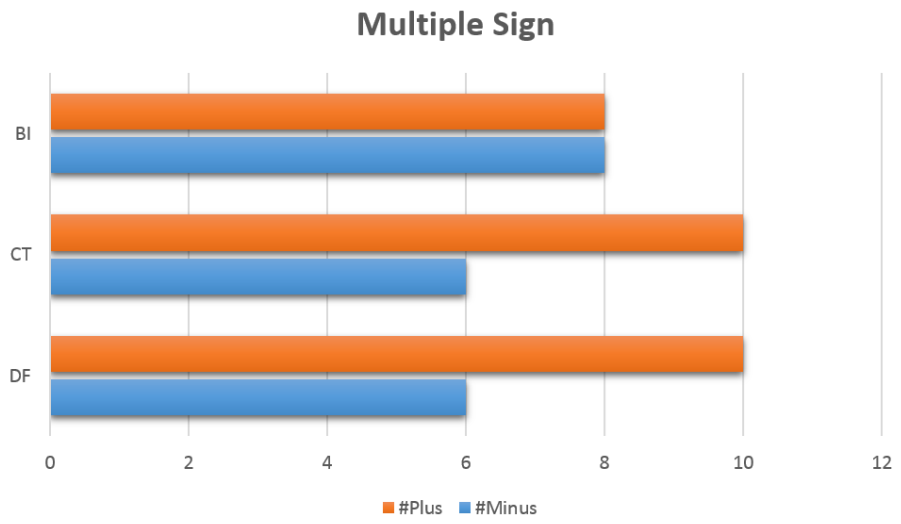


Figure 5.9: Ranking based on Multiple Sign Test for Experiment-II

In our set of experiments, total number of datasets is 16. We have selected Euclidean distance-measure method as the control method. Our null hypothesis is that the Euclidean distance-measure method performs best. Based on our null hypothesis, for the control method to be performing best, it should have at least 9 'minus' signs. From the table-5.9, it is evident that in case of commute-time distance and diffusion distance, Euclidean distance is clearly performing better. But in case of biharmonic distance, both are tied at same result. In this situation,

Table 5.9: Comparison of MMR of datasets across different distance measures based on Multiple Sign Test with EC as Control Method

| Dataset | EC | DF | CT | BI |
|---|---|---|---|---|
| AT&T (face) | 0.114 | 0.129 (-) | 0.143 (-) | 0.131 (-) |
| Breast_cancer | 0.303 | 0.300 (+) | 0.331 (-) | 0.290 (+) |
| Diabetes | 0.304 | 0.287 (+) | 0.298 (+) | 0.278 (+) |
| Haberman | 0.233 | 0.218 (+) | 0.248 (-) | 0.223 (+) |
| Heart | 0.320 | 0.333 (-) | 0.338 (-) | 0.328 (-) |
| Ionosphere | 0.138 | 0.153 (-) | 0.122 (+) | 0.160 (-) |
| Iris | 0.033 | 0.001 (+) | 0.000 (+) | 0.000 (+) |
| Liver | 0.345 | 0.359 (-) | 0.361 (-) | 0.354 (-) |
| MNIST (number) | 0.088 | 0.086 (+) | 0.086 (+) | 0.073 (+) |
| Reduced_at&t | 0.166 | 0.185 (-) | 0.169 (-) | 0.179 (-) |
| Reduced_mnist | 0.089 | 0.087 (+) | 0.088 (+) | 0.085 (+) |
| Reduced_yale | 0.348 | 0.380 (-) | 0.372 (-) | 0.354 (-) |
| Thyroid | 0.048 | 0.053 (-) | 0.026 (+) | 0.044 (+) |
| Two-Norm | 0.041 | 0.042 (-) | 0.045 (-) | 0.040 (+) |
| Wine | 0.262 | 0.278 (-) | 0.293 (-) | 0.270 (-) |
| YALE (face) | 0.251 | 0.276 (-) | 0.261 (-) | 0.269 (-) |
| #plus | | 6 | 6 | 8 |
| #minus | | 10 | 10 | 8 |
| Performance | Control Method | Poor | Poor | Equal |

it is difficult to identify any one of them to be performing better than the other. So it may be inferred safely that both biharmonic distance and Euclidean distance performs similar.

#### 5.3.2.3 Contrast Estimation based on Median Test

In this technique, we compare between each pair of distance-measure methods across all the datasets. Here, we try to get an estimate of the relative performances between two methods of each pair of test methods across all datasets. To do so, we compute the difference between the median of all pairs of test methods. Based on these medians, we try to conclude which one of the method performs better than the other methods. The steps of this technique are stated below.

**Method**

1. The difference between performances of all possible pair of test-methods are computed for each sample results. For $n$ test-methods, we get $n(n-1)/2$ test-method pairs. For each test-method pair, the difference between the performances of two methods are computed and recorded in the table. They are represented as

$$D_{ij}^k = x_j^k - x_i^k \tag{5.11}$$

   where $i$, $j$ are two methods compared and $k$ is the dataset. $x_i^k$ represents the performance of the $i$-th test method for the $k$-th dataset

2. For each pair of dataset, the median of the performance of that pair across all dataset is computed. The median for method $D_{ij}$ is represented as $Z_{ij}$ which is computed as

$$Z_{ij} = \begin{cases} median(D_{ij}^k) & where\ i < j \\ 0 & where\ i = j \\ -Z_{ij} & where\ i > j \end{cases} \tag{5.12}$$

3. Now the estimator for each methods are computed as

$$M_i = \sum_{i=1}^{n} Z_{ij} \tag{5.13}$$

4. The contrast estimator for each test-method pairs are computed as

$$M_{ij} = m_i - m_j \tag{5.14}$$

5. Based on these $M_{ij}$ values, the contrast estimation table is populated. Now, if $M_{ij}$ is greater than or less than 0, then it can be concluded that method-$i$ performs better or poorer than method-$j$ respectively. If $M_{ij} = 0$, then none of them can be said to be performing better than the other. So, both methods are assumed to be having same performance.

In our experiments, there are 4 test-methods. So there are 6 possible test-method pairs. Based on these 6 pairs, the difference of performance for each of the 16 datasets are computed and the table-5.10 is populated.

In table-5.10, the column $D_{ij}$ represents the performance difference between $i$-th and $j$-th test-methods. Here, first, second, third and fourth methods are the Euclidean, diffusion, commute-time and biharmonic distance-measures respectively. The medians $Z_{ij}$ along each columns $D_{ij}$ are calculated individually and are recorded in the row 'median'.

Then the estimators $m_i$ for each of the test-methods are computed individually. The results obtained are recorded in the table-5.11. From these results, the contrast estimation $m_i$ for each pair of test-methods are computed and the values are recorded in the table-5.12. These values are based on the medians of the performances of each test-methods.

Table 5.10: Pairwise comparison of MMR of datasets across different distance measures

| Dataset | $D_{12}$ | $D_{13}$ | $D_{14}$ | $D_{23}$ | $D_{24}$ | $D_{34}$ |
|---|---|---|---|---|---|---|
| AT&T (face) | 0.015 | 0.029 | 0.017 | 0.014 | 0.002 | -0.012 |
| Breast_cancer | -0.003 | 0.028 | -0.013 | 0.031 | -0.010 | -0.041 |
| Diabetes | -0.017 | -0.006 | -0.026 | 0.011 | -0.009 | -0.020 |
| Haberman | -0.015 | 0.015 | -0.010 | 0.030 | 0.005 | -0.025 |
| Heart | 0.013 | 0.018 | 0.008 | 0.005 | -0.005 | -0.010 |
| Ionosphere | 0.015 | -0.016 | 0.022 | -0.031 | 0.007 | 0.038 |
| Iris | -0.032 | -0.033 | -0.033 | -0.001 | -0.001 | 0.000 |
| Liver | 0.014 | 0.016 | 0.009 | 0.002 | -0.005 | -0.007 |
| MNIST (number) | -0.002 | -0.002 | -0.015 | 0.000 | -0.013 | -0.013 |
| Reduced_at&t | 0.019 | 0.013 | 0.033 | -0.016 | -0.006 | 0.010 |
| Reduced_mnist | -0.002 | -0.001 | -0.004 | 0.001 | -0.002 | -0.003 |
| Reduced_yale | 0.032 | 0.024 | 0.006 | -0.008 | -0.026 | -0.018 |
| Thyroid | 0.005 | -0.022 | -0.004 | -0.027 | -0.009 | 0.018 |
| Two-Norm | 0.001 | 0.005 | -0.001 | 0.003 | -0.002 | -0.005 |
| Wine | 0.016 | 0.031 | 0.008 | 0.015 | -0.008 | -0.023 |
| YALE (face) | 0.025 | 0.010 | 0.018 | -0.015 | -0.007 | 0.008 |
| Median | 0.009 | 0.007 | 0.003 | 0.002 | -0.006 | -0.009 |

Table 5.11: Estimator value for each of the Test-methods

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| $m_1$ | 0.019 | $m_2$ | -0.013 | $m_3$ | -0.017 | $m_4$ | 0.012 |

Table 5.12: Pairwise comparison of datasets across different distance measures by means of Contrast Estimation based on Medians.

| | EC | DF | CT | BI |
|---|---|---|---|---|
| EC | 0.000 | 0.008 | 0.009 | 0.002 |
| DF | -0.008 | 0.000 | 0.001 | -0.006 |
| CT | -0.009 | -0.001 | 0.000 | -0.007 |
| BI | -0.002 | 0.006 | 0.007 | 0.000 |

From the table-5.12, it is evident that Euclidean distance-method gives the best result as it gives positive contrast estimation value when compared with biharmonic distance-method, diffusion distance-method and commute-time distance-method individually. Similarly, the second best is biharmonic distance-method. When it is compared with diffusion distance-method and commute-time distance-method, it gives positive contrast estimation value. So it performs better than both of them. Now, the third best method is the diffusion distance-method followed by the commute-time distance-method.

#### 5.3.2.4 Friedman Aligned Rank Test

This method is similar to the Friedman rank test with some variations. In the Friedman test, the ranking is done across all test-methods for each dataset individually. So for $k$-methods and for each of the $n$-datasets, ranks range from 1 to $k$. In this ranking scheme, the ranking is done only within a particular dataset. After doing the intra-set ranking, the cumulative ranks are averaged and the conclusion is drawn based on that result. But for small number of datasets, averaging may not give a good result. This disadvantage is handled smartly in case of aligned ranking.

In the Friedman aligned ranking technique, the individual datasets across all test-methods are not ranked. Instead, all the $n * k$ results from $n$-datasets and $k$-methods are considered for ranking together. They are ranked from 1 to $nk$ based on the aligned result of each dataset. This aligned ranking is done based on the aligned observations. These aligned observation are nothing but the difference between the original performance and the average performance across all test-methods for a particular dataset. Thus the original results are converted to the aligned results. The steps of this technique are stated below.

**Method**

1. The $\{x_{ij}\}_{n\times k}$ matrix is given where each row represent each sample of results and each column represent each test methods. The element '$x_{ij}$' represents the original result of the $i$-th sample of the $j$-th method. This original result is converted to aligned result matrix $\{a_{ij}\}_{n\times k}$

2. Ranking is done for all the elements of the matrix. The best performing test method gets rank 1, second best gets rank 2 and so on. In case of a tie, the mean rank is given to the tied methods. In this way, the rank matrix is populated. Here, total results is $nk$, so they are ranked from 1 to $nk$. This aligned result is converted to rank matrix $\{r_{ij}\}_{n\times k}$

3. The mean performance of each row are computed as

$$x_j = \frac{1}{k}\sum_{i=1}^{k} r_{ij} \tag{5.15}$$

   where $x_{ij}$ is result of $i$-th test-method of the $j$-th dataset.

4. From each result of a row, subtract the mean value of that row from the original result. The new result $y_{ij}$ is computed as

$$y_{ij} = x_{ij} - x_j \tag{5.16}$$

5. Now based all the $n * k$ $y_{ij}$ values, they are ranked in their increasing order. The $y_{ij}$ having minimum value is ranked 1 and the $y_{ij}$ having maximum value is ranked nk. In case of a tie, the average rank is given to all the tied contenders.

6. The average rank of each test-methods are computed. The test-method having least average rank is concluded to be the best performing method.

7. The Friedman Aligned test statistics is computed as

$$T = \frac{(k-1)[\sum_{j=1}^{n} R_j^2 - kn^2(kn+1)^2/4]}{[kn(kn+1)(2kn+1)/6] - [\sum_{i=1}^{n} R_i^2/k]} \tag{5.17}$$

where $k$ is number of test-methods
'n' is the number of datasets
'$R_j$' is the total rank of the $j$-th dataset and
'$R_i$' is the total rank of the $i$-th test-method.

In our experiments, there are 16 datasets and 4 test-methods. So, total number of results is 64. The average results of each of the datasets are computed individually. These average results are subtracted from all the four corresponding results of that particular dataset. Now, based on these new values so obtained, all the results are ranked from 1 to 64 in increasing order of their value. The result having minimum value will be ranked 1 while the result with maximum value will be ranked 64. In case of a tie, the average rank is given to all of the tied results. The aligned ranks are specified after delimiter '|' . Based on the above method, the table-5.13 is generated.

The total rank and the average rank gives us a good idea about the performance of the test-methods based on aligned rankings. From the average rank as obtained from table-5.13, it is evident that biharmonic distance-measure method performs best. It is closely followed by Euclidean distance-measure method. In

Table 5.13: Comparison of MMR of datasets across different distance measures along with ranks based on Friedman Aligned Test

| Dataset | Mean | EC | DF | CT | BI |
|---|---|---|---|---|---|
| AT&T (face) | 0.129 | -0.015\|5 | 0.000\|32 | 0.014\|58 | 0.002\|37 |
| Breast_cancer | 0.306 | -0.003\|19 | -0.006\|20 | 0.025\|63.5 | -0.016\|1.5 |
| Diabetes | 0.292 | 0.012\|56.5 | -0.005\|22.5 | 0.006\|49.5 | -0.014\|6.5 |
| Haberman | 0.231 | 0.003\|41 | -0.013\|8.5 | 0.018\|62 | -0.008\|19 |
| Heart | 0.330 | -0.010\|11.5 | 0.003\|41 | 0.008\|51.5 | -0.002\|27 |
| Ionosphere | 0.143 | -0.005\|22.5 | 0.010\|54 | -0.021\|1 | 0.017\|60 |
| Iris | 0.009 | 0.025\|63 | -0.008\|17.5 | -0.009\|15 | -0.009\|16 |
| Liver | 0.355 | -0.010\|11.5 | 0.004\|44.5 | 0.006\|49.5 | -0.001\|29.5 |
| MNIST (number) | 0.083 | 0.005\|47 | 0.003\|41 | 0.003\|41 | -0.010\|12.5 |
| Reduced_at&t | 0.175 | -0.009\|15 | 0.010\|54 | -0.006\|20 | 0.004\|44.5 |
| Reduced_mnist | 0.087 | 0.002\|37 | 0.000\|32 | 0.001\|34.5 | -0.002\|27 |
| Reduced_yale | 0.364 | -0.016\|3.5 | 0.017\|60 | 0.008\|51.5 | -0.010\|12.5 |
| Thyroid | 0.043 | 0.005\|47 | 0.010\|54 | -0.017\|2 | 0.001\|34.5 |
| Two-Norm | 0.042 | -0.001\|29.5 | 0.000\|32 | 0.003\|41 | -0.002\|27 |
| Wine | 0.276 | -0.014\|6.5 | 0.002\|37 | 0.017\|60 | -0.006\|21.5 |
| YALE (face) | 0.264 | -0.013\|8.5 | 0.012\|56.5 | -0.003\|24.5 | 0.005\|47 |
| Total | | 430 | 606.5 | 624.5 | 419 |
| Average | | 26.875 | 37.906 | 39.031 | 26.188 |
| Rank | | 2 | 3 | 4 | 1 |

third and fourth place are commute-time distance-measure method and diffusion distance-measure method respectively.

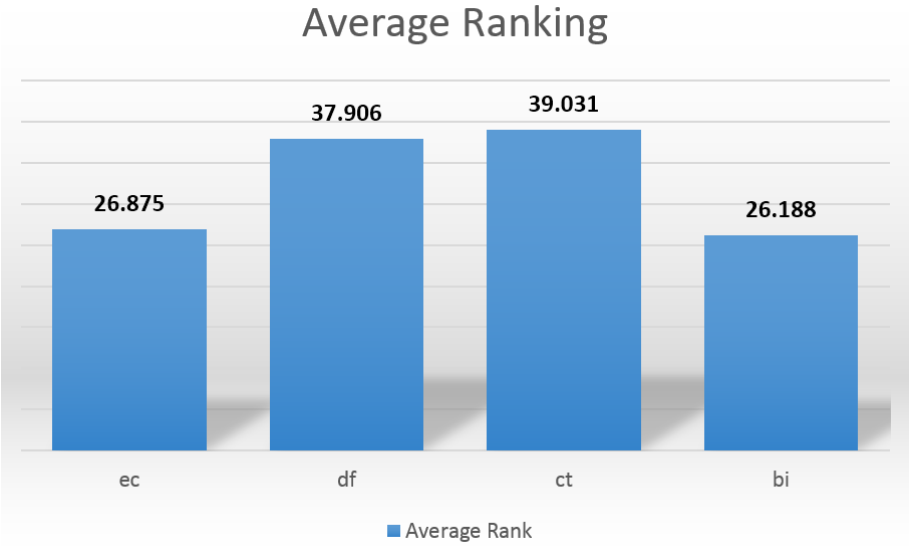


Figure 5.10: Ranking based on Friedman Aligned Test for Experiment-II

#### 5.3.2.5 Quade Test

In the Friedman test, all datasets considered, are assumed to be having same importance in evaluating the test methods. This may not give the best accuracy while concluding the performance ranking of the test-method. So some weightage is given to the dataset so that the ranking is scaled based on the relevance of each dataset. The Quade test is used to implement this. It take into consideration the observed differences in performances of the test-methods. Thus, it gives a weighted ranking to all the sample results. The method of Quade Test is described below.

**Method**

1. The $\{x_{ij}\}_{n\times k}$ matrix is given where each row represent each sample of results and each column represent each test methods. The element $x_{ij}$ represents the original result of the $i$-th sample of the $j$-th method.

2. The range of each dataset is computed as the difference between maximum

value and minimum value of dataset across all test-methods. The range $G_j$ of the $j$-th dataset is given as

$$G_j = max\{x_{ij}\} - min\{x_{ij}\} \tag{5.18}$$

where $x_{ij}$ is result of $i$-th test-method of the $j$-th dataset.

3. Based on these $G_j$ value, all datasets are ranked in their increasing order. The dataset having minimum $G_j$ value is ranked 1 and the one having maximum $G_j$ value is ranked $n$. The rank of the $j$-th dataset is denoted as $R_j$.

4. Each sample result is ranked from 1 to $k$ in their increasing order. The minimum valued result is ranked 1 while the maximum valued result is ranked $k$. The rank of the result of the $i$-th test-method of the $j$-th dataset is represented as $r_{ij}$.

5. The average rank within a dataset is $(k+1)/2$. The difference between the rank of a sample result within a dataset and its average rank is computed. This value is multiplied with the corresponding rank of that dataset to obtain a new score $S_{ij}$. This score gives us an idea about the relative significance of each dataset. The equation to compute $S_{ij}$ is written as

$$S_{ij} = R_j[r_{ij} - (k+1)/2] \tag{5.19}$$

6. Another score $W_{ij}$ is computed that do not adjust the average performance of the dataset. It is the product of the rank of a sample result of a dataset and its corresponding dataset rank. The equation to compute $W_{ij}$ is

$$W_{ij} = R_j * r_{ij} \tag{5.20}$$

7. $S$ and $W$ are computed for each test methods. They are nothing but the sum of the $S_{ij}$ and $W_{ij}$ value along each test method. The equations for

evaluating $S$ and $W$ are given below

$$S_i = \sum_{j=1}^{n} S_{ij} \tag{5.21}$$

and

$$W_i = \sum_{j=1}^{n} W_{ij} \tag{5.22}$$

8. The Quade test statistics $T$ is given as

$$T = \frac{(n-1)B}{(A-B)} \tag{5.23}$$

$$\text{where } A = \frac{nk(n+1)(2n+1)(k^2-1)}{72}$$

$$\text{and } B = \frac{1}{n}\sum_{j=1}^{k} S_j^2$$

In our experiment, total number of dataset is 16 and total number of test-methods is 4. So the dataset rank $R_i$ ranges from 1 to 16 while the test-method ranks ranges from 1 to 4. Based on the above method, the table-5.14 is computed.

This table-5.14 contains the the range and rank of each dataset along with $S$-value and $W$-value of each test-method for each dataset. Based on the range of each dataset across all test-methods, they are ranked from 1 to 16. In the test-method columns, the value before the delimiter '|' represents the $S_{ij}$ value while the value after delimiter '|' represents the $W_{ij}$ value of the $i$-th test-method of the $j$-th dataset. The cumulative sum of all the $S$-values and $W$-values about all the columns are given in the last two rows i.e. $S$ and $W$ rows. Based on these $S$-value and $W$-value, the test-methods are ranked. Lesser the value, better is the rank. From the table-5.14, it is observed that the average rank of biharmonic distance-measure method is the best followed by Euclidean distance-measure method. In

third place is a tie between commute-time distance-measure method and diffusion distance-measure method.

Table 5.14: Comparison of MMR of datasets across different distance measures along with the ranks based on Quade Test

| Dataset | Range | Rank | EC | DF | CT | BI |
|---|---|---|---|---|---|---|
| AT&T (face) | 0.029 | 10 | -15\|10 | -5\|20 | 15\|40 | 5\|30 |
| Breast_cancer | 0.041 | 16 | 8\|48 | -8\|32 | 24\|64 | -24\|16 |
| Diabetes | 0.026 | 8 | 12\|32 | -4\|16 | 4\|24 | -12\|8 |
| Haberman | 0.030 | 11 | 5.5\|33 | -16.5\|11 | 16.5\|44 | -5.5\|22 |
| Heart | 0.018 | 5 | -7.5\|5 | 2.5\|15 | 7.5\|20 | -2.5\|10 |
| Ionosphere | 0.038 | 15 | -7.5\|30 | 7\|45 | -22.5\|15 | 22.5\|60 |
| Iris | 0.033 | 14 | 21\|56 | 5.5\|42 | -14\|21 | -14\|21 |
| Liver | 0.016 | 4 | -6\|4 | 2\|12 | 6\|16 | -2\|8 |
| MNIST (number) | 0.015 | 3 | 4.5\|12 | 0\|7.5 | 0\|7.5 | -4.5\|3 |
| Reduced_at&t | 0.019 | 6 | -9\|6 | 9\|24 | -3\|12 | 3\|18 |
| Reduced_mnist | 0.004 | 1 | 1.5\|4 | -0.5\|2 | 0.5\|3 | -1.5\|1 |
| Reduced_yale | 0.032 | 13 | -19.5\|13 | 19.5\|52 | 6.5\|39 | -6.5\|26 |
| Thyroid | 0.027 | 9 | 4.5\|27 | 13.5\|36 | -13.5\|9 | -4.5\|18 |
| Two-Norm | 0.005 | 2 | -1\|4 | 1\|6 | 3\|8 | -3\|2 |
| Wine | 0.031 | 12 | -18\|12 | 6\|36 | 18\|48 | -6\|24 |
| YALE (face) | 0.025 | 7 | -10.5\|7 | 10.5\|28 | -3.5\|14 | 3.5\|21 |
| S | | | -37 | 44.5 | 44.5 | -52 |
| W | | | 2.228 | 2.827 | 2.827 | 2.118 |
| Rank | | | 2 | 3.5 | 3.5 | 1 |

## 5.4 Discussion

In our experiments, we have performed some non-parametric statistical testing to compare the performances of the four distance-measure methods. Each of the statistical testing used here have different significance.

The Friedman test used gives us an idea about the performances of each algorithms compared individually for each dataset and then taking the mean of that result to give the final conclusion. The two variants of Friedman test i.e. Friedman Aligned rank test and Quade test also does the same thing with some modifications.

Friedman aligned rank test align the sample results by taking the difference of the results and its mean. Based on these aligned results, the aligned ranks are computed. On the other hand, Quade test gives weightage to each dataset based on the range of data across all test methods. This weightage denotes the significance of the datasets in analyzing the test method's performance. This original value is multiplied with the weightage to get the weighted value. Now, all the values are of same importance, so all the datasets across all test methods are ranked together.

The other two test i.e. Multiple-sign test and Contrast estimation test works differently. In multiple-sign test, one test method is selected as the control method. All the other test method are compared individually with this control method. The null hypothesis is that the control method performs better than the other methods. Based on the test results, it is concluded whether the null hypothesis is accepted or rejected.

In the Contrast estimation test, all test methods are compared with each other. For each pair of tests, the performances are analyzed and concluded which one is better. As per the outcome of these pair-wise comparison tests, the test-method giving positive contrast estimation value for all the pair-wise tests is concluded to be the best performing test-method.

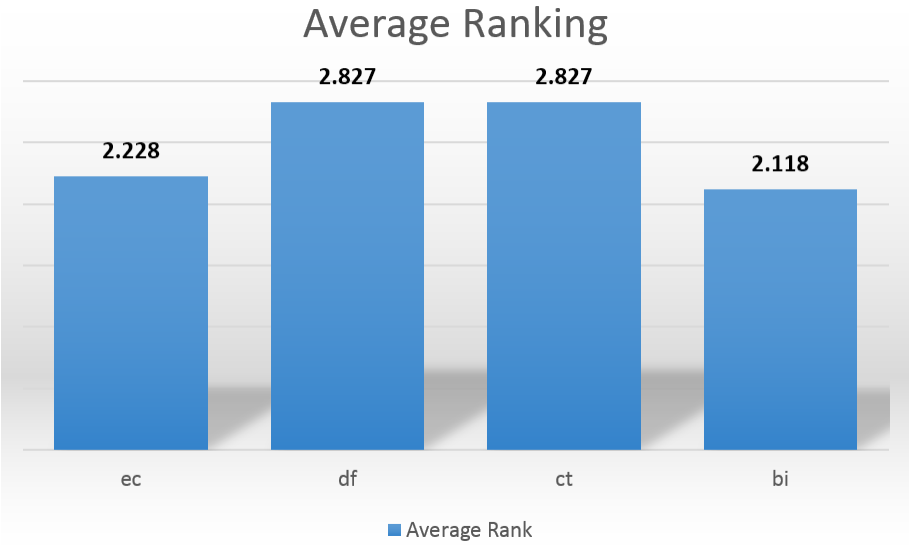


Figure 5.11: Ranking based on Quade Test result for Experiment-II

# Chapter 6

# Conclusion

## 6.1 Summary

The main goal of our work was to study whether the performance of Locally Linear Embedding can be enhanced while classifying the real datasets by using some different distance measures, other than traditional Euclidean distance measure. The high-dimensional datasets are embedded into low-dimensional space by LLE, thus reducing the complexity of classification. The distance measure in classical LLE is Euclidean distance, which is replaced with diffusion distance, commute-time distance and biharmonic distance.

For our study, we selected 16 datasets as our working database on which the dimensionality reductions, classifications and statistical tests are performed one after another. In our experiment, LLE is used as the dimensionality reduction method. The reduced datasets are classified using $k$-Nearest Neighbour classifier. Based on the misclassification results, the performance of all these four distance measures are compared using five non-parametric statistical tests i.e. Friedman Test, Multiple Sign Test, Contrast Estimation Test, Friedman Aligned Test and Quade Test.

Table 6.1: Summary of all the Statistical Test results

| Test | EC | DF | CT | BI |
|---|---|---|---|---|
| Friedman Test (Exp-I) | 2 | 4 | 1 | 3 |
| Friedman Test (Exp-II) | 2 | 3 | 4 | 1 |
| Multiple Sign Test | Control Method | Poor | Poor | Equal |
| Contrast Estimation Test | 1 | 3 | 4 | 2 |
| Friedman Aligned Test | 2 | 3 | 4 | 1 |
| Quade Test | 2 | 3.5 | 3.5 | 1 |

The table-6.1 gives the summary of the statistical test results and the ranks of the distance-measure methods in those tests. From the above table, it is evident that Euclidean distance performs best in one test (Contrast Estimation Test), biharmonic distance comes first in three tests (Friedman Test (Exp-II), Friedman Aligned Test and Quade Test) while commute-time distance performs best in one test (Friedman Test (Exp-I). On the other hand, Euclidean distance performs better than biharmonic distance in two cases (Contrast Estimation Test and Friedman Test (Exp-I)) whereas biharmonic distance gives better performance in three cases (Friedman Test (Exp-II), Friedman Aligned Test and Quade Test). The result of the other test i.e. Multiple Sign Test is inconclusive. Here, both Euclidean distance and biharmonic distance performs better than both commute-time distance and diffusion distance individually. Now, but when Euclidean distance and biharmonic distance are compared with one another, inference cannot be drawn in favour of any one of them. Although commute-time distance is the best performing method based on Friedman Test (Exp-I), both commute-time distance and diffusion distance measures lag behind considerably from Euclidean distance and biharmonic distance measures in respect of performances, considering all the tests together.

## 6.2 Future Work

To extend the ideas in this thesis, further studies are planned. They are discussed in brief below.

The first study is to extend this work of applying different distance measure on other linear and non-linear dimensionality reduction techniques. Some other non-linear dimensionality reduction techniques like Laplacian Eigenmap, Isomap, LTSA and MVU also computes the nearest neighbours in a similar approach to that of LLE. In these techniques, our distance measure methods i.e. diffusion distance, commute-time distance and biharmonic distance can be applied instead of the traditional Euclidean distance measure. The outcomes of those experiments can be extensively studied and analyzed.

The second study is to generalize the value of the 'number of nearest neighbour' parameter i.e. $K$ used in LLE. It is an important parameter as the earlier studies have shown that with the change in value of $K$, the result varies considerably. The performance of LLE is quiet sensitive to the pre-decided value of the parameter $K$. So an adaptive approach for selecting the $K$ value can be tried. In that case, the value of $K$ will be dynamic and will vary for each datapoint based on the local geometry of the dataset. Here also, this adaptive approach can be applied to other non-linear dimensionality reduction techniques like LTSA.

Finally, this work can be extended by continuing the experimentation on more number of real dataset so that a more concluding inference can be drawn. Beside, it can be investigated in the future how the effectiveness of this proposed method can be improved in terms of computational cost.

# Bibliography

[1] Arunasakthi K, KamatchiPriya L, A Review On Linear and Non-Linear dimensionality reduction Techniques, MLAIJ (2014)

[2] Balasubramanian M, Schwartz EL, Tenenbaum JB, de Silva V, Langford JC (2002), The Isomap algorithm and topological stability, Science 295:7

[3] Belkin M, Niyogi P (2003), Laplacian eigenmaps for dimensionality reduction and Laplacian eigenmaps for dimensionality reduction and data representation, Neural Computation 15(6):1373–1396

[4] Bengio et al,"Out-of-Sample Extensions for LLE, Isomap, MDS, Eigenmaps, and Spectral Clustering", In Advances in Neural Information Processing Systems (2004)

[5] Bjorn Hafner, Biharmonic Distance

[6] C.O.S. Sorzano, J. Vargas, A. Pascual-Montano, A survey of dimensionality reduction techniques

[7] D. DeMers and G. Cottrell, Non-linear dimensionality reduction, In Advances in Neural Information Processing Systems, volume 5, pages 580–587, San Mateo, CA, USA, 1993. Morgan Kaufmann

[8] De Ridder D, Kouropteva O, Okun, Oleg (2003), Supervised locally linear embedding, In: Proceedings of ICANN, pp 333–341

[9] Gashler M and Martinez T., Temporal Nonlinear Dimensionality Reduction, In Proceedings of the International Joint Conference on Neural Networks IJCNN'11, pp. 1959–1966, 2011

[10] G.H. Golub and W. Kahan, "Calculating the singular values and pseudo-inverse of a matrix", J. SIAM Series B 2:2 (1965), pp. 205–224

[11] G.N. Ramadevi, K. Usharani, Study on Dimensionality Reduction Techniques and Applications, CSEA (2012)

[12] H. Hoffmann, Kernel PCA for novelty detection, Pattern Recognition, 40(3):863–874, 2007

[13] I.K.Fodor, A Survey of Dimension Reduction Techniques, UCRL-ID-148494

[14] I.T. Jolliffe, Principal Component Analysis, Springer-Verlag, New York, 1989

[15] James A. Albano and David W. Messinger, Euclidean Commute Time Distance Embedding and its Application to Spectral Anomaly Detection, Proc. of SPIE Vol. 8390 83902G

[16] John A. Lee, Michel Verleysen, Nonlinear Dimensionality Reduction, Springer, 2007

[17] John W. Sammon, Jr, A Nonlinear Mapping for Data Structure Analysis, IEEE Transactions on Computers 1969

[18] Joshua B. Tenenbaum, Vin de Silva, John C. Langford, A Global Geometric Framework for Nonlinear Dimensionality Reduction

[19] Juan M. Banda, Rafal A. Angryk, Petrus C. Martens, Quantitative Comparison of Linear and Non-Linear Dimensionality Reduction Techniques for Solar Image Archives, Proceedings of the Twenty-Fifth International Florida Artificial Intelligence Research Society Conference

[20] J. Wang, Z. Zhang, and H. Zha, Adaptive manifold learning, In Advances in Neural Information Processing Systems, volume 17, pages 1473–1480, Cambridge, MA, USA, 2005, The MIT Press

[21] Kilian Q. Weinberger, Lawrence K. Saul, An Introduction to Nonlinear Dimensionality Reduction by Maximum Variance Unfolding, American Association for Artificial Intelligence (2006)

[22] Lawrence K. Saul, Sam T. Roweis, An Introduction to Locally Linear Embedding

[23] L.J.P. van der Maaten, E.O. Postma, H.J. van den Herik, Dimensionality Reduction: A Comparative Review, Elsevier (2008)

[24] L.K. Saul, K.Q. Weinberger, J.H. Ham, F. Sha, and D.D. Lee, Spectral methods for dimensionality reduction, In Semisupervised Learning, Cambridge, MA, USA, 2006, The MIT Press

[25] L. Vandenberghe and S. Boyd, Semidefinite programming, SIAM Review, 38(1):49–95, 1996

[26] Luh Yen, Denis Vanvyve, Fabien Wouters, Francois Fouss, Michel Verleysen, Marco Saerens, Clustering using a random walk based distance measure, ESAAN/2005 proceedings

[27] M.E. Wall, A. Reichtsteiner and L.M. Rocha, "Singular value decomposition and principal component analysis", In A Practical Approach to Microarray Data Analysis (D.P. Berrar, W. Dubitzky, and M. Granzoweds.), pp. 91–109, Kluwer, Norwell, MA, 2003

[28] Michail Vlachos, Carlotta Domeniconi, Dimitrios Gunopulos, George Kollios, Nick Koudas, Non-Linear Dimensionality Reduction Techniques for Classification and Visualization, SIGKDD (2002)

[29] Mikhail Belkin and Partha Niyogi, Laplacian Eigenmaps and Spectral Techniques for Embedding and Clustering, Advances in Neural Information Processing Systems 14, 2001, p. 586–691, MIT Press

[30] M. Penrose, Random Geometric Graphs, Oxford University Press, 2003

[31] Ronald R. Coifman, Stephane Lafon, Diffusion maps, Appl. Comput. Harmon. Anal. 21 (2006) 5–30

[32] Salvador Garcia, Alberto Fernandez, Julian Luengo, Francisco Herrera, Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power, Information Sciences 180 (2010) 2044–2064

[33] S. T. Roweis and L. K. Saul, Nonlinear dimensionality reduction by locally linear embedding, Science 290, 2323-2326 (2000)

[34] T. Cox and M. Cox, Multidimensional Scaling (Chapman & Hall, London, 1994)

[35] T. Kohonen, Self-organization and Associative Memory, Springer-Verlag, Berlin, 1988

[36] Ulrike von Luxburg, Agnes Radl, Matthias Hein, Hitting and Commute Times in Large Random Neighbourhood Graphs, JMLR (2014)

[37] Ulrike von Luxburg, Agnes Radl, Matthias Hein, Getting lost in space: Large sample analysis of the commute distance

[38] Varini C, Degenhard A, Nattkemper TW (2006), ISOLLE: LLE with geodesic distance, Neurocomputing 69(13–15):1768–1771

[39] X. He and P. Niyogi, Locality preserving projections, In Advances in Neural Information Processing Systems, volume 16, page 37, Cambridge, MA, USA, 2004. The MIT Press

[40] Yaron Lipman, Raif Rustamov, and Thomas Funkhouser, Biharmonic distance, ACM Transactions on Graphics, 29(3), June 2010

[41] Zhang Z, Zha H (2005), Principal manifolds and nonlinear dimensionality reduction via tangent space alignment, SIAM J Science Computation 26(1):313–338

# Chapter 7

# Appendix

## 7.1 Floyd-Warshall Algorithm

Floyd–Warshall algorithm is an algorithm for finding shortest paths in a weighted graph with positive or negative edge weights without any negative cycles. This algorithm compares all possible paths through the graph between each pair of vertices. It does so by incrementally improving an estimate on the shortest path between two vertices, until the estimate is optimal. The time complexity and space complexity of this algorithm is $O(n^3)$. The algorithm is described below.

Let $d_{ij}^k$ be the length of the shortest path from $i$ to $j$ such that all the intermediate vertices on the path are in the set $\{1, 2, ..., k\}$. For each pairs of vertices, the shortest path could be either a path that only uses vertices in the set $\{1, 2, ..., k\}$ or a path that goes from $i$ to $k+1$ and then from $k+1$ to $j$. Now, the shortest path from $i$ to $j$ contains vertex $k$. So it contains a subpath from $i$ to $k$ and $k$ to $j$. Each subpath can only contain intermediate vertex $\{1, 2, ..., k-1\}$. They are represented as $d_{ik}^{k-1}$ and $d_{kj}^{k-1}$. So the path length is $d_{ik}^{k-1} + d_{kj}^{k-1}$. Now, the shortest path is $d_{ij}^k = \min(d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1})$. Using this equation recursively, the shortest path between all the pairs of points are computed.

## 7.2 Datasets

The features of the 16 datasets are given in details below.

**AT&T** : The 1024 features are the 32x32 image pixels.

**Breast_cancer** : The 9 features are clump thickness, uniformity of cell size, uniformity of cell shape, marginal adhesion, single epithelial cell size, bare nuclei, bland chromatin, normal nucleoli, mitosis.

**Diabetes** : The 8 features are number of times pregnant, plasma glucose concentration a 2 hours in an oral glucose tolerance test, diastolic blood pressure, triceps skin fold thickness, 2-Hour serum insulin, body mass index, diabetes pedigree function, age, class variable.

**Haberman** : The 3 features are age of patient at time of operation, patient's year of operation, number of positive auxiliary nodes detected.

**Heart** : The 13 features are age, sex, chest pain type, resting blood pressure, serum cholesterol, fasting blood sugar, resting electrocardiographic results, maximum heart rate achieved, exercise induced angina, old-peak = ST depression induced by exercise relative to rest, the slope of the peak exercise ST segment, number of major vessels colored by fluoroscopy, thal.

**Ionosphere** : The 34 features are the radar data collected by a system in Goose Bay, Labrador.

**Iris** : The 4 features are sepal length, sepal width, petal length, petal width.

**Liver** : The 5 features are mcv mean corpuscular volume, alkaline phosphatase, sgpt alamine aminotransferase, sgot aspartate aminotransferase, gamma-glutamyl transpeptidase.

**MNIST** : The 784 features are the 28x28 image pixels.

**Reduced_at&t** : The 103 features are top 10% principal eigenvectors of the AT&T dataset computed by applying PCA algorithm.

**Reduced_mnist** : The 79 features are top 10% principal eigenvectors of the MNIST dataset computed by applying PCA algorithm.

**Reduced_yale** : The 103 features are top 10% principal eigenvectors of the YALE dataset computed by applying PCA algorithm.

**Thyroid** : The 5 features are collected from Garavan Institute, Sydney, Australia.

**Two-Norm** : This dataset is obtained from the Delve datasets.

**Wine** : The 13 features are Alcohol, Malic acid, ash, alkalinity of ash, magnesium, total phenols, flavonoids, nonflavanoid. phenols, proanthocyanins, color intensity, hue, OD280/OD315 of diluted wines, proline.

**YALE** : The 1024 features are the 32x32 image pixels.

## 7.3 Results

Detailed result of Experiment-I for each of the datasets individually is given in the table-7.1

Table 7.1: Rank distribution based on Experiment-I across different distance measures for all embeddings of each of the 16 datasets

| | **AT&T** | | | | **Breast_cancer** | | | |
|---|---|---|---|---|---|---|---|---|
| | EC | DF | CT | BI | EC | DF | CT | BI |
| 1 | 38 | 4 | 2 | 4 | 8 | 0 | 0 | 2 |
| 2 | 4 | 13 | 18 | 13 | 1 | 1 | 6 | 2 |
| 3 | 3 | 15 | 14 | 16 | 1 | 3 | 2 | 4 |
| 4 | 3 | 16 | 14 | 15 | 0 | 7 | 1 | 2 |
| Total | 67 | 139 | 136 | 138 | 13 | 39 | 22 | 26 |
| Average | 1.396 | 2.896 | 2.833 | 2.875 | 1.300 | 3.900 | 2.200 | 2.600 |
| | | | | | | | | |
| | **Diabetes** | | | | **Haberman** | | | |
| | EC | DF | CT | BI | EC | DF | CT | BI |
| 1 | 4 | 3 | 4 | 1 | 6 | 3 | 7 | 8 |
| 2 | 1 | 5 | 5 | 1 | 6 | 6 | 4 | 8 |
| 3 | 2 | 2 | 1 | 7 | 3 | 9 | 7 | 5 |
| 4 | 5 | 2 | 2 | 3 | 9 | 6 | 6 | 3 |
| Total | 32 | 27 | 25 | 36 | 63 | 66 | 60 | 51 |
| Average | 2.667 | 2.250 | 2.083 | 3.000 | 2.625 | 2.750 | 2.500 | 2.125 |

| | **Heart** | | | | **Ionosphere** | | | |
|---|---|---|---|---|---|---|---|---|
| | EC | DF | CT | BI | EC | DF | CT | BI |
| 1 | 4 | 3 | 4 | 1 | 6 | 3 | 7 | 8 |
| 2 | 1 | 5 | 5 | 1 | 6 | 6 | 4 | 8 |
| 3 | 2 | 2 | 1 | 7 | 3 | 9 | 7 | 5 |
| 4 | 5 | 2 | 2 | 3 | 9 | 6 | 6 | 3 |
| Total | 32 | 27 | 25 | 36 | 63 | 66 | 60 | 51 |
| Average | 2.667 | 2.250 | 2.083 | 3.000 | 2.625 | 2.750 | 2.500 | 2.125 |

| | **Iris** | | | | **Liver** | | | |
|---|---|---|---|---|---|---|---|---|
| | EC | M1 | M2 | M3 | EC | M1 | M2 | M3 |
| 1 | 4 | 1 | 4 | 3 | 3 | 3 | 4 | 2 |
| 2 | 4 | 3 | 5 | 0 | 1 | 2 | 1 | 8 |
| 3 | 3 | 4 | 0 | 5 | 2 | 7 | 2 | 1 |
| 4 | 1 | 4 | 3 | 4 | 6 | 0 | 5 | 1 |
| Total | 25 | 35 | 26 | 34 | 35 | 28 | 32 | 25 |
| Average | 2.083 | 2.917 | 2.167 | 2.833 | 2.917 | 2.333 | 2.667 | 2.083 |

| | **MNIST** | | | | **Reduced_at&t** | | | |
|---|---|---|---|---|---|---|---|---|
| | EC | DF | CT | BI | EC | DF | CT | BI |
| 1 | 7 | 8 | 19 | 14 | 22 | 6 | 13 | 7 |
| 2 | 3 | 21 | 16 | 8 | 8 | 14 | 9 | 17 |
| 3 | 4 | 13 | 11 | 20 | 15 | 13 | 13 | 7 |
| 4 | 34 | 6 | 2 | 6 | 3 | 15 | 13 | 17 |
| Total | 161 | 113 | 92 | 114 | 95 | 133 | 122 | 130 |
| Average | 3.354 | 2.354 | 1.917 | 2.375 | 1.979 | 2.771 | 2.542 | 2.708 |

| | **Reduced_mnist** | | | | **Reduced_yale** | | | |
|---|---|---|---|---|---|---|---|---|
| | EC | DF | CT | BI | EC | DF | CT | BI |
| 1 | 3 | 14 | 7 | 18 | 30 | 5 | 6 | 7 |
| 2 | 4 | 13 | 13 | 12 | 6 | 14 | 20 | 8 |
| 3 | 3 | 12 | 8 | 9 | 6 | 17 | 9 | 16 |
| 4 | 32 | 3 | 4 | 3 | 6 | 12 | 13 | 17 |
| Total | 144 | 88 | 103 | 81 | 84 | 132 | 125 | 139 |
| Average | 3.524 | 2.095 | 2.452 | 1.929 | 1.750 | 2.750 | 2.604 | 2.896 |

| | **Thyroid** | | | | **Two-Norm** | | | |
|---|---|---|---|---|---|---|---|---|
| | EC | DF | CT | BI | EC | DF | CT | BI |
| 1 | 2 | 4 | 4 | 4 | 7 | 5 | 3 | 3 |
| 2 | 3 | 5 | 5 | 5 | 4 | 2 | 8 | 4 |
| 3 | 4 | 3 | 4 | 1 | 2 | 7 | 3 | 6 |
| 4 | 3 | 0 | 3 | 6 | 5 | 4 | 4 | 5 |
| Total | 32 | 23 | 31 | 34 | 41 | 46 | 44 | 49 |
| Average | 2.667 | 1.917 | 2.583 | 2.278 | 2.778 | 2.556 | 2.444 | 2.722 |

| | **Wine** | | | | **YALE** | | | |
|---|---|---|---|---|---|---|---|---|
| | EC | DF | CT | BI | EC | DF | CT | BI |
| 1 | 5 | 3 | 2 | 2 | 35 | 3 | 5 | 5 |
| 2 | 2 | 0 | 9 | 1 | 6 | 7 | 22 | 13 |
| 3 | 2 | 6 | 0 | 4 | 3 | 25 | 12 | 8 |
| 4 | 3 | 3 | 1 | 5 | 4 | 13 | 9 | 22 |
| Total | 27 | 33 | 24 | 36 | 72 | 144 | 121 | 143 |
| Average | 2.250 | 2.750 | 2.000 | 3.000 | 1.500 | 3.000 | 2.521 | 2.979 |