A Study of the ISOMAP Algorithm and Its Applications in Machine Learning

Lucas Oliveira David

Department of Computing
Universidade Federal de Sao Carlos

Supervisor: Dr. Alexandre Luis Magalhes Levada

December 2015

Declaration

I hereby declare that I carried out the work entitled "A Study of the ISOMAP Algorithm and Its Applications in Machine Learning", which is being submitted to Universidade Federal de São Carlos, under the supervision of Dr. Alexandre Luis Magalhães Levada, as the partial fulfillment of the requirements for the award of the Degree of Bachelor in Computer Science in the Department of Computing. I solemnly declare that to the best of my knowledge, no part of this report has been submitted here or elsewhere in a previous application for award of a degree. All sources of knowledge used have been duly acknowledged.

....

Lucas Oliveira David, 407917 Date

Approval

"A Study of the ISOMAP Algorithm and Its Applications in Machine Learning," written by Lucas Oliveira David as the partial fulfillment of the requirements for the award of the Degree of Bachelor in Computer Science in the Department of Computing, has been approved and accepted by the following:

.....	.....
Dr. Alexandre Luis Magalhães Levada	Date
.....	.....
Dr. Heloisa de Arruda Camargo	Date
.....	.....
Dr. Ricardo Cerri	Date

Abstract

This project aims to study the foundations of nonlinear dimensionality reduction through manifold learning with the algorithm known as Isometric Feature Mapping (ISOMAP) and observe the application of the algorithm in practical experiments. The experiments were developed and executed in the following computational environment: Intel Core i7-4700MQ CPU 2.40GHz $\times$ 8, 16 GB of RAM. All the artifacts, (e.g., source-code, docs, experiments) can be found in the repository `https://github.com/lucasdavid/manifold-learning` and are licensed under the MIT License.

Contents

List of Figures

List of Tables

List of Abbreviations

DR	Dimensionality Reduction
ISOMAP	Isometric Feature Mapping
MDS	Multidimensional Scaling
ML	Machine Learning
SVM	Support Vector Machine

List of Symbols

$\mathbf{1}_n$	The column vector of 1's
$\mathbf{\Sigma}_X$	The covariance matrix associated with the matrix X
$\mathbf{H}$	The centering matrix
$G = (V, E)$	A graph represented by a tuple of vertices and arcs.
λ	The diagonal matrix containing the eigenvalues λ_i
M	A n-dimensional topological manifold
σ	Standard deviation
U_p	An open set of a manifold M
V_p	An open set of the Euclidean space $\mathbb{R}^n$
$\mathbf{X}_{n \times f}$	A data set X with n samples and f features

Chapter 1

Introduction

Throughout the years, machine learning techniques have grown popular between both academical and the corporative sectors. Their vast applications and promising results [6] indubitably contributed to our current scenario where not only computer scientists or mathematicians, but engineers, psychologists and many other groups have taken interest [7] on how to adapt and apply these studies to their own problems.

Machine learning can help us to understand large amount of data and take decisions based on it. To achieve this, however, we must first find ways to effectively (and efficiently) extract the information that lies within the data.

Many different machine learning algorithms have been developed during this century and the last one. Among those, many could successfully generalize low dimensional data [8]. In the other hand, problems of our world are often too complex and may be represented by high dimensional data. For example, images, sounds or text documents can be expressed as vectors of the $\mathbb{R}^n$, where each element corresponds to a pixel, wave signal or term, respectively. When analyzing these problems, we observed that many of the algorithms would often become unstable. Dimensionality reduction (or DR) then quickly became a key concept for minimizing the data size while maintaining its meaning.

Nowdays, dimensionality reduction has evolved into an extensive area. Being approached by many different perpectives, it can not only be applied to reduce the data size, specifically, but often employed in data preprocessing, visualization, noise reduction and many other purposes. Nonetheless, the area still presents

questions and great challenges to be solved. For example, it is still difficult to evaluate a reduction or even define generic metrics for it, as the "correctness" of a reduction is always associated with the specific problem domain in hand. Another issue is the difficulty attached to reduce nonlinearly distributed data sets.

In this work, we will focus on Isometric Feature Mapping (ISOMAP), a classic algorithm for Manifold Learning and highly regarded for nonlinear dimensionality reduction given its effectiveness when it has its pre-conditions met. First, the relevant background will be presented in order to contextualize the reader and provide concepts that are closely related to machine learning, dimensionality reduction and, of course, the ISOMAP algorithm. We will proceed to study linear dimensionality reduction, its algorithms, applications and limitations. Following, ISOMAP will be covered: its concept, computational complexity, extensions and limitations. Finally, experiments created during the project are presented for both observation and comparison.

Chapter 2

Relevant Background

2.1 Data set

In the context of Computer Science, very often our goal is to develop machines that can assist or automate the process of solving real-world problems. Firstly, however, we must find ways to express these problems numerically. This is where "data sets" come in.

Although the recurrent usage of the term "data set" in scientific work, there is not a clear definition established. It is possible, however, to observe the regular presence of four related features: grouping, content, relatedness and purpose [9]. For the scope of this work, the term data set is invariably associated with the idea of a collection of samples. Each sample is a sequence of features, where the *i-th* feature of all instances belong to a same set of symbols f_i.

Succinctly, let S be a set of samples and $F \coloneqq \{f_i \mid f_i \text{ is a set of symbols}\}$. Then, the dataset $\mathbf{X}$ is defined as:

$$\mathbf{X} \coloneqq [x_{ij}] \mid x_{ij} \in f_j, \forall i \in [1, |S|], \forall j \in [1, |F|]$$

Example of a canonical data set

The table bellow illustrates an example of data set, where each row represents a **sample**, and each column a **feature**.

	Sepal length	**Sepal width**	**Petal length**	**Petal width**	**Species**
1	5.1	3.5	1.4	0.2	I. setosa
2	4.9	3.0	1.4	0.2	I. setosa
3	4.7	3.2	1.3	0.2	I. setosa
...	...	...	...	...	...

Table 2.1: The first three samples of the Iris flower data set [5].

Iris flower is an example of data set broadly used in machine learning demonstrations, being usually interpreted as a classification problem where the feature *Species* will be learned from its adjacent features. In that scenario, *Species* is denominated **target feature**.

2.1.1 Data set as a collection of vectors in the $\mathbb{R}^n$

A data set can have each one of its nominal features enumerated, i.e., mapped to an element of $\mathbb{N}$. Such set could then be expressed as a collections of vectors in the $\mathbb{R}^n$. Consider the data set bellow:

	Age	**Gen.**	**TB**	**DB**	**Alk.**	**Sgpt**	**Sgot**	**TP**	**ALB**	**A/G**	**S**
1	65	Female	0.7	0.1	187	16	18	6.8	3.3	0.9	1
2	62	Male	10.9	5.5	699	64	100	7.5	3.2	0.74	1
3	62	Male	7.3	4.1	490	60	68	7	3.3	0.89	1
...	...	...	...	...	...	...	...	...	...	...	...

Table 2.2: The first three samples of the Indian Liver Patient Dataset (ILPD) [5].

Composed by 583 samples and 11 features, the data set ILPD has a nominal feature $Gender \coloneqq \{Male, Female\}$. *Gender* can, of course, be mapped on $\{0, 1\}$. ILPD can finally be expressed by the figure bellow:

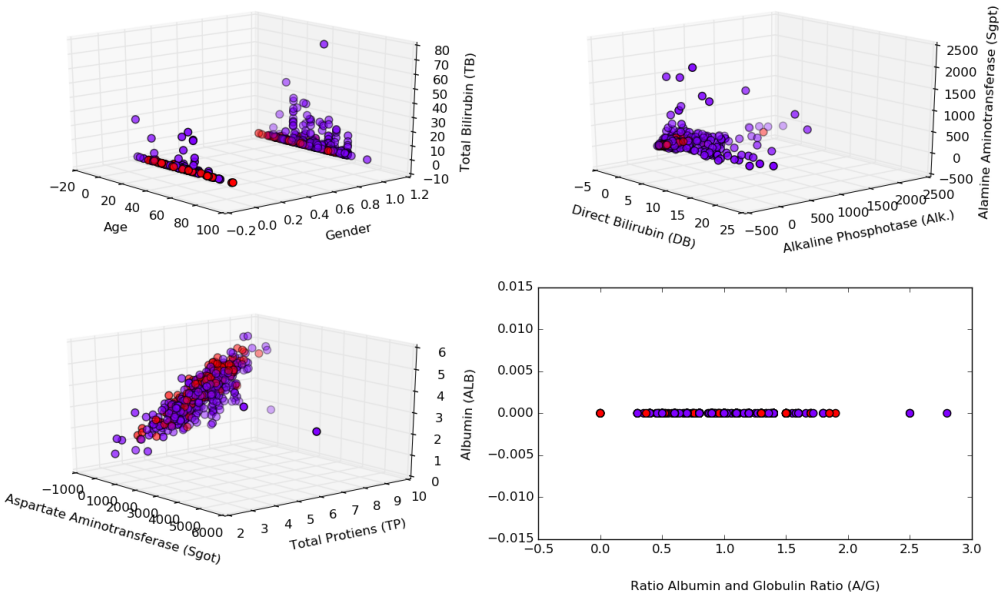


Figure 2.1: The data set ILPD's samples mapped onto the $\mathbb{R}^n$, where each of its features is an axis in one graph, except for $S \coloneqq \{1, 2\}$, which was represented by the vertices' colors.

As many graphs required to display the data set, it is quite difficult to identify a plausible distribution for ILPD. We define here our first encouragement towards the study of dimensionality reduction: the identification of the most significant features (i.e., that maximize variance) and plotting of those might result on simpler and more intuitive representations. Furthermore, it would also be interesting to combine the existing features to create new ones that are even more representative.

2.1.2 Modern Problems and Applications

Differently from Iris flower or ILPD data set, data sets associated with modern problems are often very dense, i.e., data sets containing many samples and/or features. Although the high number of samples is essentially benefic, a high number of features might be irrelevant or even unconstructive to the learning process [10]. The Leukemia data set, illustrated in the table 2.3, is an example of extremelly high dimensionality data set, as it is a sub set of the $\mathbb{R}^{7130}$.

	F1	F2	...	F7129	F7130
1	-1.46236	-0.645135	...	-0.959575	1
2	-0.664799	0.206146	...	-0.543433	1
3	-0.200487	0.379941	...	-0.896774	1
...	...	...	...	...	...
72	-0.455835	-0.071517	...	-0.068667	1

Table 2.3: The Leukemia data set, with 72 samples and 7130 features [5].

In many cases, there are indicatives that the data set lie near a lower-dimensional manifold embedded in the $\mathbb{R}^n$ [11]; that is, there is a smaller set of features which roughly express the information within. A second encouragement can then be set: it is possible that the data set might be shrunk by combining similar (linearly dependent) features or eliminating the ones that poorly contribute towards the learning process. In order to do this, one must be able to qualify the "contribution" of each feature or even identify dependencies between features.

2.2 Probability Theory

2.2.1 Feature Normalization and Standardization

Many of the methods ahead will require the data set to be centered in the origin. This is equivalent to remove the mean from each one of the data set's features.

Let $[\mathbf{X}]_{n\times f}$ be a data set, $\mathbf{X}_{.j}$ the j-th column of the matrix $\mathbf{X}$, μ_j the mean of $\mathbf{X}_{.j}$ and $\mathbf{1}_n$ the column vector of 1's, we build a **normalized** data set $\mathbf{X}'$ s.t. each column has zero mean:

$$\mathbf{X}'_{.j} = \mathbf{X}_{.j} - \mu_j \mathbf{1}_n$$

If σ_j is the standard deviation of $X_{.j}$, it is also possible to build the **standardized** X', where each column has zero mean and it is contracted by its standard deviation:

$$X'_{.j} = \frac{X_{.j} - \mu_j \mathbf{1}_n}{\sigma_j}$$

2.2.2 Centering Matrix

The symmetric matrix $\mathbf{H}$ is named the **centering matrix** when the multiplication of it by a matrix $\mathbf{X}$ produces the same effect of subtracting the mean of the components from each component of $\mathbf{X}$. $\mathbf{H}$ is defined as:

$$\mathbf{H} = \mathbf{I}_n - \frac{1}{n}\mathbf{1}\mathbf{1}^\top, \text{ where:}$$

1. $\mathbf{I}_n$ is the identity matrix of order n.
2. $\mathbf{1}$ is the column vector of 1's.

2.2.3 Variance

Variance is the measure which describes how far the samples in a given set X vary. For the scope of this project, only discrete probabilities will be considered. That is, if X represents a random variable with known distribution $P(x)$, where $P(x) = k \in \mathbb{R}^+, \forall x \in X$ and $\sum_{x \in X} P(x) = 1$, μ is the population mean of X and $\mathbf{1}_n$ is the column vector of 1's, then, for n samples of X [12]:

$$Var(X) = \frac{1}{n}(X - \mu\mathbf{1}_n) \cdot (X - \mu\mathbf{1}_n) = \frac{1}{n}\sum_{x \in X}(x - \mu)^2$$

Example 2.2.1. *If* $X = \{1, 2, -2, 4\}$ *and* $\mu = \frac{1}{n}\sum_{x \in X} x = \frac{1+2-2+4}{4} = 1.25$*, then*

$$\begin{aligned} var(X) &= \frac{1}{n}\sum_{x \in X}(x - \mu)^2 \\ &= \frac{(1 - 1.25)^2 + (2 - 1.25)^2 + (-2 - 1.25)^2 + (4 - 1.25)^2}{4} \\ &= 4.6875 \end{aligned}$$

Example 2.2.2. *The variance of the Sepal length feature* $\mathbf{X}_{.0}$ *in the Iris flower data set can be calculated as:*

$$\begin{aligned} var(X) &= \frac{1}{150}\sum_{i}(x_{i,0} - \mu)^2 \\ &= \frac{1}{150}[(5.1 - 5.84)^2 + (4.9 - 5.84)^2 + \cdots + (5.9 - 5.84)^2] \\ &= \frac{102.17}{150} = .681122 \end{aligned}$$

Let $[\mathbf{X}]_{n\times f}$ be a data set, μ_j the mean of the j-th column of $\mathbf{X}$ and $\mathbf{1}_n$ the column vector of 1's. If $var(\mathbf{X}_{.j}) = 0, \forall j \in [0, f)$, then all samples in $\mathbf{X}$ are the same.

Proof.

$$\begin{aligned} Var(\mathbf{X}_{.j}) &= 0 \\ \frac{1}{n}(\mathbf{X}_{.j} - \mu_j \mathbf{1}_n) \cdot (\mathbf{X}_{.j} - \mu_j \mathbf{1}_n) &= 0 \\ \mathbf{X}_{.j} - \mu_j \mathbf{1}_n &= 0 \\ \iff & \\ \mathbf{X}_{.j} &= \mu_j \mathbf{1}_n \end{aligned}$$

All elements in $\mu_j \mathbf{1}_n$ are the same. Therefore $\mathbf{X}_{ij} = \mu_j, \forall i \in [0, n)$. □

2.2.4 Covariance

The covariance measures the variance of two random variables in respect to each other. Formally, if X and Y are two given random variables with known mean population distribution μ_X and μ_Y, respectively, then

$$\sigma(X, Y) = \frac{1}{n}(X - \mu_X) \cdot (Y - \mu_Y)$$

Simply putting, the covariance of two random variables X and Y can be interpreted as one of the following behaviors:

- $\sigma(X, Y) > 0$ X tends to increase as Y increases.
- $\sigma(X, Y) < 0$ X tends to increase as Y decreases.
- $\sigma(X, Y) = 0$ X and Y are completely unrelated.

Remark 2.2.1. *For a random variable X, $Var(X) = \sigma(X, X)$.*

Covariance Matrix of Features in a Centered Data Set

Let $\mathbf{X}$ be a data set, $\mathbf{X}'$ the data set $\mathbf{X}$ with its features centered ($\mathbf{X}' = \mathbf{HX}$), and $\mathbf{X}'_{.j}$ the *j-th* feature column of the centered data set $\mathbf{X}'$, the covariance between

each pair of features can be represented by the matrix:

$$
\begin{aligned}
\mathbf{\Sigma}_X &= [\sigma_{xy}]_{n\times n} \\
&= \begin{bmatrix} \sigma(\mathbf{X}'_{.0}, \mathbf{X}'_{.0}) & \sigma(\mathbf{X}'_{.0}, \mathbf{X}'_{.1}) & \cdots & \sigma(\mathbf{X}'_{.0}, \mathbf{X}'_{.n-1}) \\ \sigma(\mathbf{X}'_{.1}, \mathbf{X}'_{.0}) & \sigma(\mathbf{X}'_{.1}, \mathbf{X}'_{.1}) & & \sigma(\mathbf{X}'_{.1}, \mathbf{X}'_{.n-1}) \\ \vdots & & & \vdots \\ \sigma(\mathbf{X}'_{.n-1}, \mathbf{X}'_{.0}) & \sigma(\mathbf{X}'_{.n-1}, \mathbf{X}'_{.1}) & \cdots & \sigma(\mathbf{X}'_{.n-1}, \mathbf{D}_{.n-1}) \end{bmatrix} \\
&= \frac{1}{n}(\mathbf{X}')^\top \mathbf{X}' \\
&= \frac{1}{n}(\mathbf{HX})^\top \mathbf{HX} \\
&= \frac{1}{n}\mathbf{X}^\top \mathbf{H}^\top \mathbf{HX} \\
&= \frac{1}{n}\mathbf{X}^\top \mathbf{HX}
\end{aligned}
$$

2.3 Numerical Analysis

2.3.1 Eigenvalues and Eigenvectors of a Matrix

Given a matrix $\mathbf{A} \neq 0 \in \mathbb{R}^{2n}$, a vector $\mathbf{v} \in \mathbb{R}^n$ is said to be an **eigenvector** of $\mathbf{A}$ if the multiplication $\mathbf{Av}$ does not change the direction of $\mathbf{v}$; that is:

$$\exists \lambda \in \mathbb{R} \mid \mathbf{Av} = \lambda \mathbf{v}, \text{ where}$$

λ is the **eigenvalue** associated to the eigenvector $\mathbf{v}$.

2.3.2 Spectral Decomposition of a Matrix

If $[\mathbf{A}]_{n\times n}$ is a symmetric matrix of rank n and admits n pairs of eigenvalues $\lambda = diag(\lambda_0, \lambda_1, ...\lambda_{n-1})$ and eigenvectors $[\mathbf{V}]_{n\times n} = [\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_{n-1}]$, such that $\mathbf{V}$ is an orthogonal matrix (i.e., $\mathbf{V}^\top \mathbf{V} = \mathbf{I}_n$), then

$$\mathbf{AV} = \mathbf{V}\lambda$$

λ is a diagonal matrix, where λ_{ii} is the eigenvalue associated to the eigenvector $\mathbf{v}_i$ [13]. Furthermore, the columns of $\mathbf{V}$ are linear independent, hence $\mathbf{V}$ is

invertible.

$$\begin{aligned}\mathbf{AV} &= \mathbf{V}\lambda \\ \mathbf{AVV}^\top &= \mathbf{V}\lambda\mathbf{V}^\top \\ \mathbf{A} &= \mathbf{V}\lambda\mathbf{V}^\top\end{aligned}$$

2.3.3 Singular Value Decomposition

Let $[\mathbf{A}]_{n\times n}$ be a matrix, then $\exists \mathbf{U} \in \mathbb{R}^{m\times m}, \mathbf{V} \in \mathbb{R}^{n\times n}$ and $\mathbf{\Sigma} = diag(\sigma_0, \ldots, \sigma_{n-1})$ conditioned to $\sigma_i \geq \sigma_{i+1} \geq 0, \forall\sigma \in [0, n)$ s.t. [14]

$$\mathbf{M} = \mathbf{U\Sigma V}^\top$$

Decomposition of Symmetric Matrices

If $\mathbf{A} = \mathbf{U\Sigma V}^\top$, $\mathbf{AA}^\top = \mathbf{U\Sigma^2 U}$ and $\mathbf{A}^\top\mathbf{A} = \mathbf{V\Sigma^2 V}$.

Proof.

$$\begin{aligned}\mathbf{A}^\top\mathbf{A} &= (\mathbf{U\Sigma V}^\top)^\top(\mathbf{U\Sigma V}^\top) \\ &= \mathbf{V\Sigma}^\top\mathbf{U}^\top\mathbf{U\Sigma V}^\top \\ &= \mathbf{V\Sigma\Sigma V}^\top \\ &= \mathbf{V\Sigma^2 V}^\top\end{aligned}$$

□

Proving $\mathbf{AA}^\top = \mathbf{U\Sigma^2 U}^\top$ is analogous to the above.

2.4 Topology

2.4.1 Manifolds

Intuitively, n-dimensional topological manifolds are sets that are "locally Euclidean" [15]. In other words, they can be decomposed into sub sets that can be mapped to the $\mathbb{R}^n$.

Formally, a set M is said to be a **n-dimensional topological manifold** $\iff$ M is a paracompact Hausdorff topological space $\mid \forall p \in M, p \in U_p$, where

U_p is an open set that is homeomorphic to an open set V_p of the Euclidean space $\mathbb{R}^n$ [15].

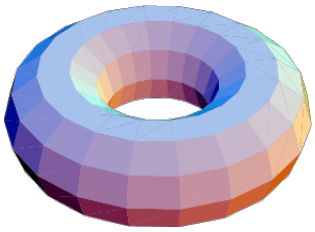


Figure 2.2: The Torus, often studied in topology, it is a manifold that can be mapped to the $\mathbb{R}^2$ [1].

From now on in this report, we will use the word manifold to refer to a n-dimensional topological manifold.

Charts

The pair (U_i, ϕ_i) is called a **coordinate chart** or **chart** on M if $U \in M$ and ϕ_i is a **homeomorphism** such that $\phi_i(U_i) = V_i \subseteq \mathbb{R}^n$ [16].

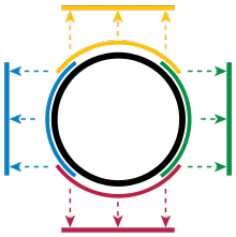


Figure 2.3: Charts mapping four regions of a circle to different open sets [2].

Atlas

A set $A = \{(U_i, \phi_i)\}_{i \in A}$ is said to be an **atlas** on a manifold M if $\cup_{i \in A} U_i = M$ [16].

Example 2.4.1. *The $\mathbb{R}^n$ is, directly, a manifold.*

[1] From "Torus," by E. W. Weisstein, *MathWorld–A Wolfram Web Resource.* Available at: mathworld.wolfram.com/Torus.html.

[2] From "Manifold," *Wikipedia - The Free Encyclopedia.* Available at: wikipedia.org/wiki/Manifold.

Example 2.4.2. *A n-dimensional sphere is a manifold. Earth, in special, is a 3-dimensional sphere and its stereographic projection (figure 2.4) is its mapping to the* $\mathbb{R}^2$ *[1].*

Figure 2.4: Stereographic projection applied to Earth [1].

2.4.2 Embedding

When talking about the relationship between two topological objects, such as two vector spaces, manifolds or graphs (section 2.5); it is interesting to imagine a mapping from one object to the other that somehow preserves its original properties.

Let A and B be two topological objects of the same type, a function $\phi\colon A \to B$ is an embedding of A into B if ϕ is an isomorphism which preserves the original properties of A [17], where such properties are relative to the type of the objects at hand. Shortly, A is said to be *embedded* in B.

2.5 Graph Theory

2.5.1 Graphs

Let G be the pair (V, E). G is defined as a **graph** [18], where

1. V is a set of objects called **vertices**.
2. E is a family of elements $e_i \in V \times V$ called arcs.

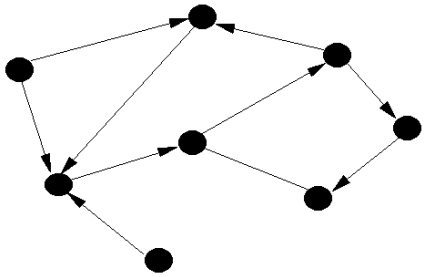


Figure 2.5: An example of a graph [3].

Basic Concepts [18]

Multiplicity If $G = (V, E)$ and $(x, y) \in E \subset V \times V$, the multiplicity $m_g^+(x, y)$ is defined to be the number of arcs with initial endpoint x and terminal endpoint y. Furthermore:

1. $m_G^-(x, y) = m_G^+(y, x)$
2. $m_G(x, y) = m_G^+(x, y) + m_G^-(x, y)$

Degree If $G = (V, E)$ is a graph and $v \in V$, the degree $d(v)$ of v is defined as $d(v) = 2n_s + n_n$, where n_s is the number of arcs self-incident at v [19] (i.e., arc with v as initial endpoint and terminal endpoint) and n_n is the number of arcs incident at v.

Adjacency Matrix If $V = \{v_1, v_2, \ldots, v_n\}$ and $G = (V, E)$, define the adjacency matrix $\mathbf{A} = [a_{ij}]_{n \times n}$ associated with graph G, where $a_{ij} = m_G^+(v_i, v_j)$.

Further Specifications

Undirected graph Let $G = (V, E)$ be a graph and $e_k \in E \mid e_k = (a, b) \in E$. The element $e_i = [a, b]$ can be defined as the **edge** that links a to b without specifying direction. Finally, define the **undirected graph** G' as (V, E'), where $E' = \{e_i\}$ is the set of edges created from E [18].

Figure 2.6 shows the graph **Les Miserables**, where each vertex is a character and each arc links two characters that have shared stage at some point during the play. The size of each vertex is a result of its own multiplicity.

[3] From "OR-Notes," by J. E. Beasley. Available at: brunel.ac.uk/ mastjjb/jeb/or/graph.html.

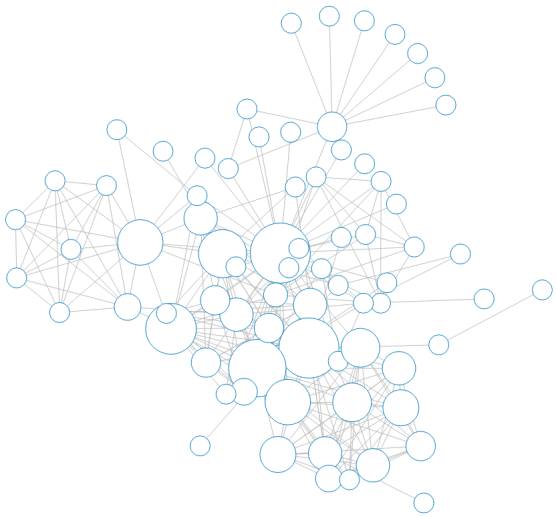


Figure 2.6: The **Les Miserables** graph [4].

Complete graph A graph $G = (V, E)$ is said to be complete if

$$\forall (x, y) \in E, x \neq y, m_G(x, y) \geq 1$$

Remark 2.5.1. *Let $G = (V, E)$. G is called the complete graph K_n if $E = V \times V$. That is,*

$$\forall (x, y) \in V \times V, \exists e \in E \mid e = (x, y)$$

Weighted graph Let $G = (V, E)$ be a graph and $w \colon E \to \mathbb{R} \mid w(e) = w_e$ be the weight associated with arc e. G is said to be a weighted graph.

Euclidean graph If $G = (V, E)$ and $W = \{w_e, \forall e \in E\} \subset \mathbb{R}$, G is said to be an Euclidean graph if w_e corresponds to the euclidean distance between the vertices connected by e in a given vector space.

Tree Let G be the graph (V, E) such that G is connected and admits no cycles. G is said to be a **tree** [18].

[4] From "Graph Theory," by L. David, 2012. Available at: comp-ufscar.github.io/graph-theory.

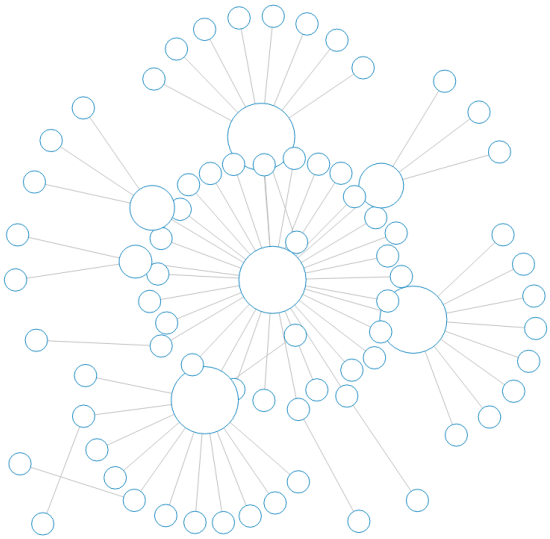


Figure 2.7: A tree extracted (a subgraph) from the **Les Miserables** graph [5].

[5]From "Graph Theory," by L. David, 2012. Available at: comp-ufscar.github.io/graph-theory.

2.5.2 Related Problems

Nearest-Neighbor Search

Let $G = (V, E)$ be a weighted graph, where $w\colon E \to \mathbb{R}$ is the weight or **length** of the arc e, and $n\colon E \to \{0, 1\}$ is a definition of **nearness** in G. The nearest-neighbor search is a optimization problem that consists of finding a subgraph $G' = (V, F \subseteq E) \mid f \in F \iff n(f) = 1$. In other words, to find a subgraph where each arc connects two vertices if and only if these vertices are "close".

K-Nearest-Neighbor Search (K-NN)

Consider $k \in \mathbb{N}$. K-NN will result in a subgraph G' s.t. each vertex v is connected at most to k other vertices and the sum of weights of the arcs incident on v is minimum:

```
def nearest_neighbors(V, E, w, k):
    neighbors = set()
    for v in V:
        vs = V - {v}
        # Sort vertices by their how close they are to v.
        vs = sort(vs, keys=[w((v, u)) for u in vs])
        # Keep only k-first vertices.
        vs = vs[0:k]
        neighbors.add(vs)
    return V, neighbors
```

Listing 1: K-Nearest Neighbors Algorithm.

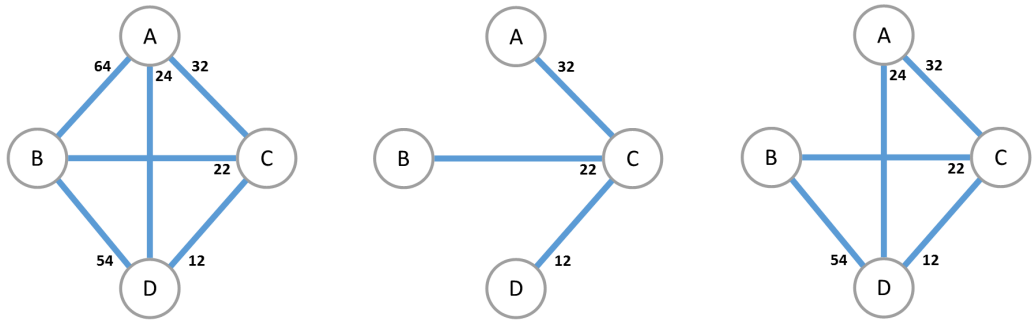
ϵ-Nearest Neighbor Search (ϵ-NN)

Fixed $\epsilon \in \mathbb{R}$, ϵ-NN will find the subgraph $G' = (V, E')$ s.t. each arc in E' has associated weight $w(e) \leq \epsilon$:

```
def nearest_neighbors(V, E, w, epsilon):
    neighbors = set()
    for v in V:
        vs = {u for u in V - {v} if w(v, u) > epsilon}
        neighbors.add(vs)
    return V, neighbors
```

Listing 2: ϵ-Nearest Neighbors Search Algorithm.

Example 2.5.1. *If $k = 1$ and $\epsilon = 60$, the graph G, the sub-graph G' found from K-Nearest neighbor algorithm and the sub-graph G'' found from the ϵ-Nearest neighbor are defined as follows:*



(a) G (b) G' (c) G''

Figure 2.8: The original graph G and the results of K-NN and ϵ-NN, respectively.

Single-pair Shortest-path Problem [20]

Let $G = (V, E)$ be a weighted graph, $w\colon E \to \mathbb{R}$ and $w(e)$ the weight associated with the arc e. Furthermore, consider the paths in $P_{x \to y} = \{(x, v_1, v_2, \ldots, y) \mid v_i \in V, \forall i \in (0, n)\}$ and their weights, defined as $w(p) = \sum_0^{n-1} w(p_i, p_{i+1})$.

Under the conditions above, a path $p \in P$ is said the **shortest-path** between x and $y \iff w(p) \leq w(q), \forall q \in P$. Additionally, the shortest-path weight between two vertices x and y is defined as:

$$\sigma(x, y) = \begin{cases} \min\{w(p(x, y))\}, & \text{if } p(x, y) \text{ exists,} \\ \infty, & \text{otherwise.} \end{cases}$$

The shortest-path between a vertex $x_0 \in X$ and all other vertices can be expressed by the tree $S = (V, F), F \subseteq E$ named **shortest-path tree**.

Dijkstra's Algorithm

Let $G = (V, E)$ be a directed weighted graph and $v_0 \in V$ the initial vertex, the shortest-path from v_0 to all other vertices can be found through Dijkstra's algorithm:

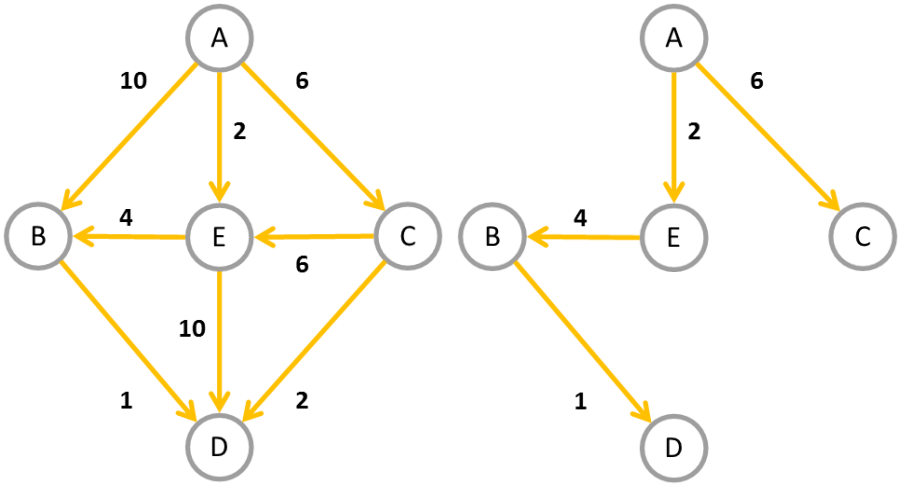
```
def dijkstra(V, E, w):
    predecessor = {v: None for v in V}
    d = {v: infinity for v in V}
    S = {}
    Q = priority_queue(V, d)

    while len(Q) > 0:
        u = Q.pop_min()
        for v in V:
            if d[v] > d[u] + w(u, v):
                d[v] = d[u] + w(u, v)
                predecessor[v] = u
    return (predecessor, d)
```

Listing 3: Dijkstra's algorithm [20].

Remark 2.5.2. *Dijkstra's algorithm is based on two strategies: greedy, when extracting the vertex u such that $d(u)$ is minimum, and dynamic programming, during edge relaxation (comparison and update of the $d(v)$ and $predecessor(v)$ values).*

Example 2.5.2. *Let G be the weighted graph as defined in figure 2.9a. The shortest-path between A and all other vertices is described by the tree S illustrated in figure 2.9b.*



(a) The weighted graph G. (b) The shortest-path tree S.

All-pairs Shortest-path Problem

Similarly from the previous problem, all-pairs shortest-path is also a search problem that aims to find the predecessors of each vertex $v_i \in V$. The difference is that there is no constraint regarding the initial vertex, i.e., one is interested in finding the shortest routes from every vertex to every other vertex in the graph.

Clearly, all-pairs shortest-path can be solved by executing the Dijkstra's Algorithm $|V|$ times, using every $v_i \in V$ as initial vertex.

Floyd-Warshall Algorithm

Another solution for the all-pairs shortest-path problem is Floyd-Warshall algorithm, which is entirely based on dynamic programming:

```
def floyd_warshall(V, E, w):
    d = {}
    for v in V:
        for u in V - {v}:
            d[v, u] = w[v, u]

        predecessor[v, u] = None

    for k in V:
        for v in V:
            for u in V:
                if d[u, v] > d[u, k] + d[k, v]:
                    d[u, v] = d[u, k] + d[k, v]
                    predecessor[u, v] = k
      return predecessors, d
```

Listing 4: Floyd-Warshall Algorithm [21].

Now, matrix d contains the weights sum for the paths between all vertices of the graph. These paths can be constructed by:

```
        def path(i, j, predecessor):
                if predecessor[i, j] == None:
                        return [i, j]
                l = path(i, predecessor[i, j])
                r = path(predecessor[i, j], j)
                return l.append(r)
```

Listing 5: Algorithm for path reconstruction from list of predecessors [21].

2.6 Machine Learning

Learning it is Improvement of an agent's performance over a specific environment through acquisition of experience [22]. Machine learning relates to the area in Computer Science which aims to extend the human learning concept to machines. That is, the implementation of machines that can imitate human learning [23]. Practically, this usually translates into creating machines that are able to recognize patterns in an environment and interpret those using concepts related to artificial intelligence. Such interpretation can create a model which might eventually be used to predict new patterns, take actions and/or solve domain problems.

Based on how the learning phase of a problem is, ML algorithms are mostly divided into one of the following categories:

Supervised A direct feedback is presented during the learning phase [22]. For the instances where the ME problem relies on a data set, the learning task uses the labeled training samples (i.e., samples that present the **target feature**) to synthesize the model that attempts to generalize the relationship between the feature vectors and the target variable [24].

Unsupervised Infer hidden structures from the data set without direct feedback, such as known labels for the samples [24].

Semi-supervised Most commonly, it is given by the extension of either supervised or unsupervised learning to include the other paradigm, resulting in a combination of both [25].

Reinforcement Through iterative exploration, the learner is positively or negatively reinforced for its actions. The learner's goal is, ultimately, maximize the cumulative reward gained [24].

Regarding the nature of what is being learned, ML algorithms can be separated between:

Classification Each sample $\mathbf{x}$ of the data set $\mathbf{X}$ is associated with a label $y_{\mathbf{x}} \in \mathbf{Y}$, regardless if this label is known or unknown. Moreover $|\mathbf{Y}| < \infty$. The ML algorithm goal is to create a model which can infer a label to a given sample.

Regression Samples of the data set $\mathbf{X}$ follow a regression function f. The goal is to learn such function. A reasonable manner to think of it is to imagine the samples of $\mathbf{X}$ being associated with continuous labels [26].

ML algorithms often represent the end-point of the learning process, receiving the result of reduction algorithms as input and learning from them instead of the raw data. In this section, we will study a small set of supervised ML algorithms. Later, we will use them to simulate how the reductions produced by ISOMAP are interpreted by "real-world" ML algorithms.

2.6.1 K-Nearest Neighbors

One might consider using the concepts of locality and proximity to perform classification or regression. K-Nearest Neighbors (K-NN) can be employed here, being possibly one of the most straight forward methods for prediction [27].

K-NN Classifier

Let $\mathbf{X}$ be a data set, $\mathbf{Y}$ the array of labels possibly assumed by samples in $\mathbf{X}$, $\mathbf{s}$ a unlabeled sample and $k \geq 1$ a fixed integer. The K-NN classifier will determine $\mathbf{s}$'s class based on the most common class shared between $\mathbf{s}$'s closest neighbors: if $\mathbf{X}_s \subset \mathbf{X}$ is the set of neighbors of $\mathbf{s}$, consider $E_{\mathbf{X}_s}[i]$ the probability of a sample in $\mathbf{X}_s$ being associated to the label i. Then,

$$y(\mathbf{s}) = \arg\max_{i \in Y} E_{\mathbf{X}_s}[i]$$

K-NN Regressor

Let $\mathbf{X}$ be a data set, $\mathbf{Y}$ the array of instants of the regression function, $\mathbf{s}$ a unlabeled sample and $k \geq 1$ a fixed integer. The K-NN regressor attempts to generalize the regression function by averaging the instants in $\mathbf{Y}$ associated with $\mathbf{s}$'s nearest neighbors [26], given by $N_k(s)$:

$$y(\mathbf{s}) = \frac{1}{k} \sum_{\mathbf{x} \in N_k(\mathbf{s})} y_{\mathbf{x}}$$

In order to increase the accuracy in both classification and regression, the importance of the label/instant associated to each neighbor can be weighted by

the distance d between it and $\mathbf{s}$, where d is defined based on the context of the problem (although the Euclidean distance is broadly employed [27]).

2.6.2 Support Vector Machine

Contained in the supervised learning category, Support Vector Machine (SVM) is a powerful tool, often used in classification and regression problems.

Intuitively, the SVM classifier attempts to find a hyperplane that separates the samples in a data set into two different groups: positives and negatives (i.e., binary classification). Furthermore, the hyperplane is placed such that the distance between the **support vectors** (the closest samples) and it are maximized.

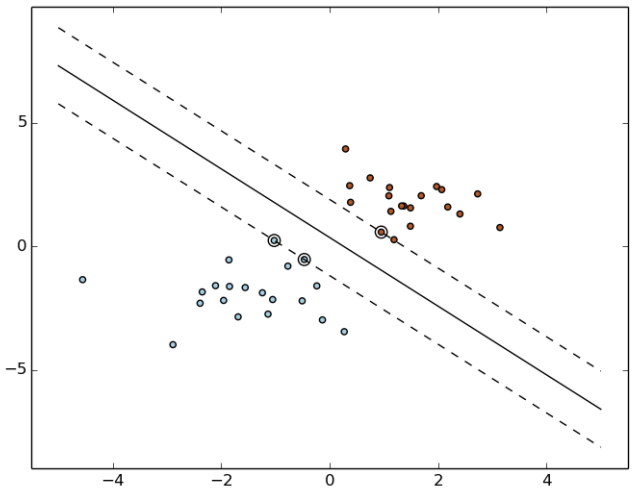


Figure 2.10: A SVM classifier projecting a hyperplane that perfectly separates two classes of samples [2]

.

More elaborately, given any linearly separable data set $\mathbf{X}$ containing samples from two distinguished classes $\{-1, +1\}$, consider the vector $\mathbf{w}$ and the hyperplane $d = \{\mathbf{x} \mid \mathbf{w} \cdot \mathbf{x} + b = 0\}$ (represented in fig. 2.10 by the contiguous line). The class

$y_i = y(x_i)$ to which a given sample $\mathbf{x}_i$ belongs can be defined as:

$$y(x_i) = \begin{cases} +1, \text{if } \mathbf{w} \cdot \mathbf{x}_i + b > 0 \\ -1, \text{otherwise.} \end{cases}$$

To prevent samples from falling into the margin or being misclassified [28], reinforce that for any positive sample $\mathbf{x}_+$, $\mathbf{w} \cdot \mathbf{x}_+ + b \geq 1$. Similarly for $\mathbf{x}_-$ samples, $\mathbf{w} \cdot \mathbf{x}_- + b \leq 1$. These both constraints can be expressed as

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 \geq 0 \tag{2.1}$$

Notice that $y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 = 0 \iff \mathbf{x}_i$ is a **support vector**.

The width (the distance between the two margins) of the street is the vector $(\mathbf{x}_+^0 - \mathbf{x}_-^0)$ projected onto the vector $\mathbf{w}$, where $\mathbf{x}_+^0$ is a positive support vector and $\mathbf{x}_-^0$ is a negative one.

$$\begin{aligned} width &= (\mathbf{x}_+^0 - \mathbf{x}_-^0) \cdot \frac{\mathbf{w}}{\|\mathbf{w}\|} \\ &= \frac{\mathbf{x}_+^0 \cdot \mathbf{w} - \mathbf{x}_-^0 \cdot \mathbf{w}}{\|\mathbf{w}\|} \\ &= \frac{1 - b - (-1 - b)}{\|\mathbf{w}\|} = \frac{2}{\|\mathbf{w}\|} \end{aligned} \tag{2.2}$$

As the goal is to maximize the width, while still respecting the constraint (2.1).

$$\max width = \max \frac{2}{\|\mathbf{w}\|} \equiv \min \frac{1}{2}\|\mathbf{w}\|^2 \tag{2.3}$$

Which can be solved using standard quadratic programming.

SVM for non-separable data sets (soft margins)

To deal with non-separable data sets, it is possible to introduce the variables ξ_i to the optimizing equation (2.3) and to the decision rule (2.1) [28]. This represents a trade-off between maximum margin/distance of the misclassified samples from the decision boundary. The optimizing equation and the decision rule are updated to:

$$\min \frac{1}{2}\|\mathbf{w}\|^2 + C \sum_1^m \xi_i, \text{ constrained to:}$$
$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i, \xi_i \geq 0$$

Such trade-off can be adjusted through the parameter C. Notice that small values for C might result in misclassification, whereas high values can produce overfitting.

Dependency over the dot product

Equation 2.3 does not explicitly illustrates how the model generated depends on the dot product between the training samples. The implications of such fact will be discussed in the next section. For now, let us convert 2.3 to its dual form. By the Lagrange multipliers method [29]:

$$L = \frac{1}{2}\|\mathbf{w}\|^2 - \sum \alpha_i [y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1] \tag{2.4}$$

$$\frac{\partial L}{\partial \mathbf{w}} = \mathbf{w} - \sum \alpha_i y_i \mathbf{x}_i = 0 \implies \mathbf{w} = \sum \alpha_i y_i \mathbf{x}_i \tag{2.5}$$

$$\frac{\partial L}{\partial b} = -\sum \alpha_i y_i = 0 \implies \sum \alpha_i y_i = 0 \tag{2.6}$$

Applying (2.5) and (2.6) on (2.4), our problem of minimizing (2.3) constrained by (2.1) becomes maximizing L, subject to (2.6):

$$\begin{aligned} L &= \frac{1}{2}\sum \alpha_i y_i \mathbf{x}_i \cdot \sum \alpha_j y_j \mathbf{x}_j - \sum \alpha_i y_i \mathbf{x}_i \cdot \sum \alpha_j y_j \mathbf{x}_j - \sum \alpha_i y_i b + \sum \alpha_i \\ &= \sum \alpha_i - \frac{1}{2}\sum\sum \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j) \end{aligned}$$

w and b can easily be found from (2.5) and $\alpha_i[y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1] = 0$ (for any $i \mid \alpha_i \neq 0$), respectively.

Now, as we finally plug (2.5) back into our decision rule, it becomes clear that both training and prediction phases depend only on the dot product between the sample vectors [29]:

$$y(\mathbf{x}_k) = \begin{cases} +1, \text{if } \sum \alpha_i y_i \mathbf{x}_i \cdot \mathbf{x}_k + b \geq 0 \\ -1, \text{otherwise.} \end{cases}$$

Kernel functions

"The kernel function represent the dot product of both vectors projected onto the new space". [29]

Some data sets are not linearly separable, as previously mentioned . They can, however, be projected to a different vector space, where they hopefully will be. To achieve this, a transformation ϕ is applied on both vectors. The dot product between those is then calculated in the new space and the value is used during training and prediction:

$$L = \sum \alpha_i - \frac{1}{2} \sum \sum \alpha_i \alpha_j y_i y_j \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j)$$

Remark 2.6.1. *Practically, Let* $k : (\mathbb{R}^n, \mathbb{R}^n) \to \mathbb{R} \mid k(\mathbf{u}, \mathbf{v}) = \phi(\mathbf{u}) \cdot \phi(\mathbf{v})$*, then:*

$$L = \sum \alpha_i - \frac{1}{2} \sum \sum \alpha_i \alpha_j y_i y_j k(\mathbf{u}, \mathbf{v})$$

$$y(\mathbf{x}_k) = \begin{cases} +1, \textit{ if } \sum \alpha_i y_i k(\mathbf{x}_i, \mathbf{x}_k) + b \geq 0 \\ -1, \textit{ otherwise.} \end{cases}$$

Which entails, given a known function k*, there is no need to explicitly know which are the projections of* u *and* v *onto the new vector space. This is commonly known as the* ***kernel trick****.*

The first set of axis $\{x\}$ in the figure bellow illustrates the non-linearly separable distribution of positive and negative samples in the $\mathbb{R}$. The second basis $\{x, y = x^2\}$ represents the same samples being projected to the $\mathbb{R}^2$.

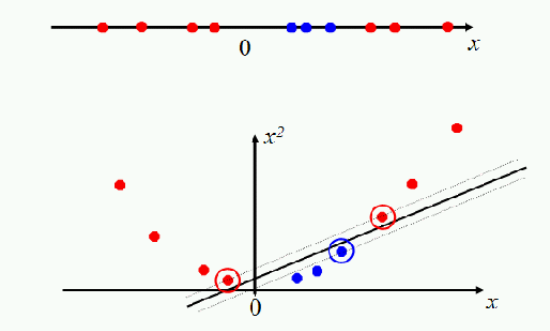


Figure 2.11: Projection of samples from the $\mathbb{R}$ to the $\mathbb{R}^2$, allowing SVM to find a hyperplane that perfectly separates both classes [3].

Many different kernels exist. Two which are frequently used among all of them [3] are:

- RBF: $\exp(-\frac{\|x-x'\|^2}{2\sigma^2})$
- Polynomial: $(u^\top v + c)^d$.

Multi-class Classification

SVMs are constructed over the concept of binary separation, but not all problems are binary. In classification, this reflects data sets that have their samples separated into more than two classes. The Iris flower data set is an example of this, when predicting the samples' *species*.

There are many different approaches for multi-class classification [30]. For the scope of this work, consider only the following:

One-vs-All (OVA) n binary classifiers are built, where n is also the number of classes in the data set. For the classifier $c^j, j \in [1, n]$, all samples of the jth class are taken as positive samples, at the same time that all the other samples are considered negative.

If $c^j(\mathbf{x})$, the j-th classification of sample $\mathbf{x}$, is defined as:

$$c^j(\mathbf{x}) = \sum_{i=1}^{m} \alpha_i^j y_i \mathbf{x} \cdot \mathbf{x}_i + b$$

Then positive values for $c^j(\mathbf{x})$ indicate that the sample $\mathbf{x}$ belongs to the jth class. Additionally, greater $c^j(\mathbf{x})$ values imply on further distance from the hyperplane (i.e., $c^j(\mathbf{x})$ can also be interpreted as a **confidence value**) and the sample x should be assigned to the class which holds greatest confidence [31]. Shortly, classification is given by

$$y(\mathbf{x}) = \arg\max_j c^j(\mathbf{x})$$

All-vs-All (AVA) Also known as all-pairs or one-vs-one, a classifier c_{ij} is built for each pair of classes (i, j), resulting in a total of $n(n-1)$ classifiers. c_{ij} responds with positive values for samples of the i-th class and negative values for samples of the j-th class. Classification can be done by simply counting the class most frequently associated with $\mathbf{x}$:

$$y(\mathbf{x}) = \arg\max_i \sum_{j=1}^{n} c_{ij}(\mathbf{x})$$

Or equivalently, but only using $\frac{n(n-1)}{2}$ classifiers,

$$y(\mathbf{x}) = \arg\max_i \sum_{j=1}^{n} \frac{j-i}{|j-i|} c_{\min(i,j)\,\max(i,j)}(\mathbf{x})$$

2.6.3 Evaluating Supervised Learners

Machine Learning algorithms might be susceptible to data noise, incorrect configuration or even random factors, which would eventually decrease the generated model's accuracy. Additionally, there is also the change of **overfitting**, the event where a learner performs sufficiently well over the learned data, but fails to make correct predictions over yet-unseen data. In order to evaluate this accuracy, models are quite often tested after trained.

Considering that testing with the same data used for training will most likely produce unreliable results, a simple way to test a learner is to separate the labeled data set into two chunks, where the first is used for training. The second chunk is then be given to the learner, which attempts to predict the samples. Finally, the predictions made by the learner would be compared with the actual labels.

Confusion Matrix and Prediction Accuracy When Classifying

To evaluate classification models, one way to visualize the incorrect predictions made by the learner is a **confusion matrix**, where the item a_{ij} is the number of times that a sample of the class i was classified as being of the class j.

	a	b	c	d
a	12	3	2	0
b	7	12	2	4
c	0	4	54	8
d	6	0	1	23

Table 2.4: Example of confusion matrix for a data-set with four different classes.

We define the **prediction accuracy** of a ML model as the number of correct predictions divided by the total number of predictions made:

$$accuracy = \frac{\sum_i c_{ii}}{\sum_i \sum_j c_{ij}}$$

Where **C** is a confusion matrix.

Naturally, a diagonal matrix **C** represents that all samples were correctly classified, being the best possible outcome (no errors).

Prediction Accuracy When Regressing

The coefficient of determination, which can be interpreted as how much data variance is explained by a prediction model, can easily be used to evaluate regression models.

Let $\mathbf{X}$ be a data set, $\mathbf{y}_i$ an observation of the regression function associated with the sample $\mathbf{x} \in \mathbf{X}$, $\mathbf{y}_i^p$ the actual value predicted by the regression model and $\bar{\mathbf{y}}$ the average of all observations, then:

$$R^2 = 1 - \frac{\sum_i (y_i^p - y_i)^2}{\sum_i (y_i^p - \bar{\mathbf{y}})^2}$$

Cross Validation

When the estimator at hand accepts different settings ("hyperparameters"), such as k in K-NN or C in the SVM, there is still a change of overfitting on the testing set, as these parameters can always be tweaked for optimal performance. In order to avoid this, a third chuck, called validation set, can be divided from the original data set and used to "validate" the learner current settings.

Sometimes, partitioning the data set into three subsets might be malign to the learning process, as the model will be constructed only considering a small, random portion of the samples (when the data set does not have too many samples, for instance). This event is known as **underfitting**. Fortunately, cross validation can be used to deal with such cases. When k-fold cross-validating [32], the data set can be partitioned into k folds. For each fold k_i, a model is trained with all folds, except for k_i. The model is then tested over k_i. Finally, the score reported by the cross-validation method is the average accuracy when testing over all folds.

Grid Search

As a learner hyperparameters strongly affect how the generated model will perform, one cannot choose them arbitrarily. Furthermore, when learners have a high number of hyperparameters, it becomes difficult to manually define a "good" setting for them. For example, SVM requires C and which *kernel* - and the kernel's own parameters, of course - it should use.

Grid Search is a traditional method used to find the learner's hyperparameters that maximize accuracy optimize the generalization of learning model over

a specific data set [33]. Given a set of possible parameters, it will exhaustively search for the combination of those that generate the best outcome, which might be evaluated by a validation method such as cross-validation.

2.6.4 Examples of Learning

Coffee Selling Rate

The figure bellow illustrates the distribution of coffee sales per time of the day in a particular coffee-shop [4]. Clearly being a regression problem, a machine learning algorithm must create a model that appropriately generalizes the distribution observed. Such model will eventually be used to predict the selling rate in the following days.

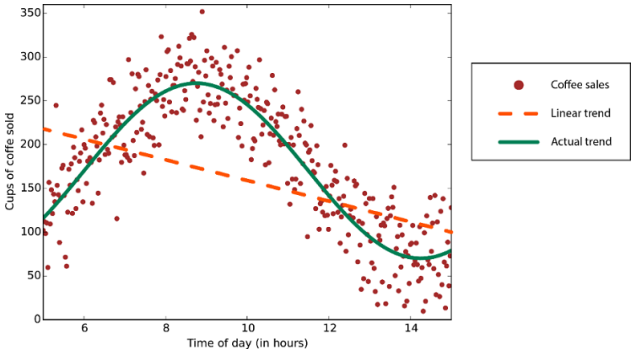


Figure 2.12: graphic representation of a data set generalization by a linear (orange) and a nonlinear model (green) [4].

The orange line and the green arc represent two different models. The orange line, which represents the linear model, clearly does not generalize the data set appropriately, once it induces an error much larger than necessary [4]. The nonlinear model, i.e., the green arc, was capable of generalize the data inducing a smaller error.

Iris Flower

Consider the Iris flower data set as defined in 2.1. In order to predict the feature *Species* of a given sample, one could train a Support Vector Machine classifier.

Through GridSearch, it was found that the SVM algorithm with $C = 100$, $gamma = .01$ and rbf kernel is capable of finding a model yielding .99% accuracy. The samples misclassified during the test phase were represented by the confusion matrix bellow.

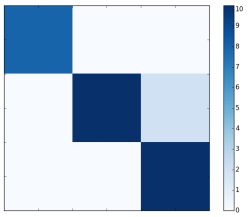


Figure 2.13: Confusion matrix of a SVM with $C = 100$, $gamma = .01$ and rbf kernel when predicting samples from the Iris flower data set.

Chapter 3

Linear Dimensionality Reduction

As presented in the previous sections, data sets with many features may present a series of issues: difficult visualization, high performance requirements, noise etc. In this section, it will be discussed methods related with linear dimensionality reduction, i.e., the shrinking of data sets by transformation and/or removal of features, while minimizing information loss.

Consider the synthetic data set $\mathbf{K}$. $\mathbf{K}$ has its samples expressed by two similarly scaled dimensions. It is clear, however, that the samples follow a very particular distribution:

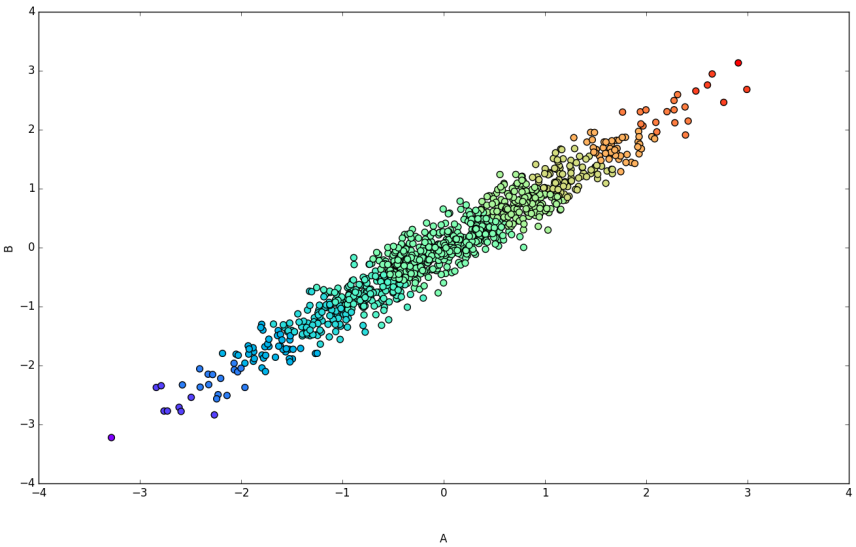


Figure 3.1: The data set $\mathbf{K} \in \mathbb{R}^n \times \mathbb{R}^n$.

Additionally, something interesting can be observed when analyzing the covariance matrix of K: as it is not a diagonal matrix, the variance of x from its mean somehow correlates with the variance of y [34].

	x	**y**
x	1.26682132	1.29158697
y	1.29158697	1.40358478

Table 3.1: Covariance between the components of **K**.

3.1 Principal Component Analysis

As in K, some data sets follow certain distributions that are majorly contained in a few orthogonal components, where a component is the result of a linear combination of the original features.

Principal Component Analysis (PCA) is a statistical technique that attempts to transform a n-dimensional data set $\mathbf{X}$ into a m-dimensional data set $\mathbf{Y}$, where, hopefully, $m \ll n$. Furthermore, the dimensions of $\mathbf{Y}$ will necessarily be orthogonal components aligned with the direction in which the variance of samples in $\mathbf{X}$ is maximum, commonly referred to as **principal components** [35]. In figure 3.2, the orange and purple arrows are the principal components of the data set **K**.

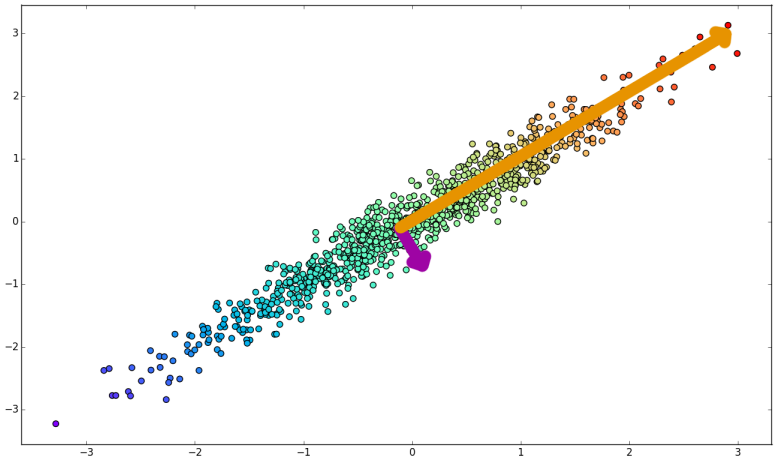


Figure 3.2: The principal components of **K** (orange and purple arrows).

3.1.1 Study of the PCA Algorithm

Let $\mathbf{D}$ be a dataset with n samples and f features and $\mathbf{X} = \mathbf{HD}$, where $\mathbf{H}$ is the centering matrix. Our goal is to find which are the principal components of the covariance matrix $\mathbf{\Sigma}_X$:

$$\mathbf{\Sigma}_X = \frac{1}{n}\mathbf{X}^\top\mathbf{X} \tag{3.1}$$

Using the **Singular Value Decomposition** method described in section 2.3.3, we known that

$$\mathbf{X} = \mathbf{U\Sigma V}^\top \tag{3.2}$$

Needless to say, $\mathbf{\Sigma}$ is the diagonal matrix of singular values, not to be mistaken by the covariance matrix $\mathbf{\Sigma}_X$.

From 3.1 and 3.2:

$$\begin{aligned}\mathbf{\Sigma}_X &= \frac{1}{n}\mathbf{X}^\top\mathbf{X} \\ &= \frac{1}{n}(\mathbf{U\Sigma V}^\top)^\top\mathbf{U\Sigma V}^\top \\ &= \frac{1}{n}\mathbf{V\Sigma^2 V}^\top\end{aligned}$$

Which entails that $\mathbf{V}$ is the orthonormal matrix with $\mathbf{\Sigma}_X$'s eigenvectors as columns, where $\mathbf{\Sigma}$ contains the correspondent eigenvalues σ_i^2 associated with $v_i \in \mathbf{V}$ in its diagonal. Now notice that the principal components also spawn $\mathbf{K}$ it $\mathbf{K}$ is centered on the origin. Under these conditions, $\sigma_i v_i$ surely matches the i-th principal component. As we are interested in the dimensions that give most variance, we keep only the $m \in \mathbb{R}$ most significant eigenvalues and their correspondent eigenvectors.

Finally, it also worth remarking once again that the principal components are linear combinations of the original features (the canonical base) and $\mathbf{V}$ is the change-of-basis matrix from the generated base to the canonical. Naturally, $\mathbf{V}^{-1}$ is a change-of-basis matrix from the original space to the one that is generated by the principal components. Formally, if x is a sample (row vector) from the $\mathbf{X}$ data set, its project y is defined as:

$$\begin{aligned}\mathbf{Vy}^\top &= \mathbf{x}^\top \\ \mathbf{y}^\top &= \mathbf{V^{-1}x}^\top\end{aligned}$$

In the other hand, $\mathbf{V}$ is orthogonal, hence $\mathbf{V}^{-1}$ exists and it is equal to $\mathbf{V}^\top$:

$$\begin{aligned}\mathbf{y}^\top &= \mathbf{V}^\top \mathbf{x}^\top \\ \mathbf{y} &= (\mathbf{V}^\top \mathbf{x}^\top)^\top \\ &= \mathbf{x}\mathbf{V}\end{aligned}$$

3.1.2 Formalization of the PCA Algorithm

Let $\mathbf{D}$ be a data set with n samples and f features and $m \in \mathbb{R}$ the number of dimensions desired for the reduced data set [36, 37].

1. Find $\mathbf{X} = \mathbf{HD}$, where $\mathbf{H}$ is the centering matrix.
2. Calculate the covariance matrix $\mathbf{\Sigma}_X$.
3. Use singular value decomposition to find the eigenvalues $\mathbf{\Sigma} = \{\sigma_i\}$ and eigenvectors $\mathbf{V} = \{v_i\}$ of $\mathbf{\Sigma}_X$.
4. Sort the eigenvalues by their absolute value in descending order and select the first m ones and their respective eigenvectors.

3.2 Multidimensional Scaling

Alternatively to PCA, Multidimensional Scaling (or simply MDS) can be used to reduce the dimensionality of a data set. The method has, however, an extensive application domain and often appears in the literature in different contexts. An example of this is the problem of, given a set of objects O and a dissimilarity measurement $\delta_{rs}, \forall (r, s) \in O \times O$, finding a suitable representation in the $\mathbb{R}^n$ for the objects in O [13].

For this project, we study the **classic MDS**. That is, when the dissimilarities considered are the Euclidean distances between coordinates in the $\mathbb{R}^n$, equivalating the method to PCA.

3.2.1 Study of the MDS

Let $\mathbf{\Delta} = [\delta_{rs}]_{n \times n}$ be the dissimilarity matrix, where δ_{rs} represents the Euclidean distances between two samples $\mathbf{x}_r, \mathbf{x}_s \in \mathbb{R}^m$ from the data set $[\mathbf{X}]_{n \times m}$ induced by

the $L2$-norm. In other words,

$$\begin{aligned}
\delta_{rs} &= \sqrt{\sum_i (x_{ri} - x_{si})^2} \\
&\iff \\
\delta_{rs}^2 &= \sum_i (x_{ri} - x_{si})^2 \\
&= (\mathbf{x_r} - \mathbf{x_s}) \cdot (\mathbf{x_r} - \mathbf{x_s}) \\
&= \mathbf{x_r} \cdot \mathbf{x_r} + \mathbf{x_s} \cdot \mathbf{x_s} - \mathbf{2x_r} \cdot \mathbf{x_s}
\end{aligned} \tag{3.3}$$

Now consider the matrix $\mathbf{B} = \mathbf{X}\mathbf{X}^\top$, where $b_{rs} = \mathbf{x}_{.r} \cdot \mathbf{x}_{.s}$. $\mathbf{B}$ can be decomposed as $\mathbf{U}\mathbf{\Sigma}\mathbf{U}^\top = \mathbf{U}\mathbf{\Sigma}^{\frac{1}{2}}\mathbf{\Sigma}^{\frac{1}{2}}\mathbf{U}^\top = \mathbf{U}\mathbf{\Sigma}^{\frac{1}{2}}(\mathbf{U}\mathbf{\Sigma}^{\frac{1}{2}})^\top = \mathbf{X}\mathbf{X}^\top \iff \mathbf{X} = \mathbf{U}\mathbf{\Sigma}^{\frac{1}{2}}$. If $\mathbf{B}$ can be derived from 3.3, the problem is reduced to simply decompose $\mathbf{B}$ and using its eigenvalues and eigenvectors (similarly to what was done in PCA) to construct the data set $\mathbf{Y}$ [13].

Firstly, we will assume that $\mathbf{Y}$ is centered in the origin (i.e., $\mathbf{Y}$ has its features' means equal to zero):

$$\sigma_f = \sum_i y_{if} = 0, \forall f \in [0, m) \tag{3.4}$$

Now, 3.3 $\implies$

$$\begin{aligned}
\frac{1}{n}\sum_r \delta_{rs}^2 &= \frac{1}{n}\sum_r (\mathbf{x}_r \cdot \mathbf{x}_r + \mathbf{x}_s \cdot \mathbf{x}_s - 2\mathbf{x}_r \cdot \mathbf{x}_s) \\
&= \frac{1}{n}\sum_r \mathbf{x}_r \cdot \mathbf{x}_r + \sum_r \mathbf{x}_s \cdot \mathbf{x}_s - 2\sum_r \mathbf{x}_r \cdot \mathbf{x}_s \\
&= \frac{1}{n}\sum_r \mathbf{x}_r \cdot \mathbf{x}_r + \sum_r \mathbf{x}_s \cdot \mathbf{x}_s - 2(\sum_r \mathbf{x}_r) \cdot \mathbf{x}_s \\
&= \frac{1}{n}\sum_r \mathbf{x}_r \cdot \mathbf{x}_r + n\mathbf{x}_s \cdot \mathbf{x}_s - 2(\mathbf{0}) \cdot \mathbf{x}_s \\
&= \frac{1}{n}\sum_r \mathbf{x}_r \cdot \mathbf{x}_r + \mathbf{x}_s \cdot \mathbf{x}_s \\
&\iff \\
\mathbf{x}_s \cdot \mathbf{x}_s &= \frac{1}{n}(\sum_r \delta_{rs}^2 - \sum_r \mathbf{x}_r \cdot \mathbf{x}_r)
\end{aligned} \tag{3.5}$$

Similarly, to 3.5:

$$\mathbf{x}_r \cdot \mathbf{x}_r = \frac{1}{n}(\sum_s \delta_{rs}^2 - \sum_s \mathbf{x}_s \cdot \mathbf{x}_s) \tag{3.6}$$

Putting 3.5 and 3.6 back in 3.3:

$$\begin{aligned}
\delta_{rs}^2 &= \frac{1}{n}(\sum_s \delta_{rs}^2 - \sum_s \mathbf{x}_s \cdot \mathbf{x}_s + \sum_r \delta_{rs}^2 - \sum_r \mathbf{x}_r \cdot \mathbf{x}_r) - 2\mathbf{x}_r \cdot \mathbf{x}_s \\
&\implies \\
\mathbf{x}_r \cdot \mathbf{x}_s &= -\frac{1}{2}(\delta_{rs}^2 - \frac{1}{n}[\sum_s \delta_{rs}^2 - \sum_s \mathbf{x}_s \cdot \mathbf{x}_s + \sum_r \delta_{rs}^2 - \sum_r \mathbf{x}_r \cdot \mathbf{x}_r]) \\
&= -\frac{1}{2}(\delta_{rs}^2 - \frac{1}{n}[\sum_s \delta_{rs}^2 + \sum_r \delta_{rs}^2 - 2\sum_r \mathbf{x}_r \cdot \mathbf{x}_r])
\end{aligned} \tag{3.7}$$

To eliminate the $\mathbf{x_r} \cdot \mathbf{x_r}$ term from 3.7:

$$\begin{aligned}
\frac{1}{n^2}\sum_s\sum_r \delta_{rs}^2 &= \frac{1}{n^2}\sum_s\sum_r(\mathbf{x}_r \cdot \mathbf{x}_r + \mathbf{x}_s \cdot \mathbf{x}_s - 2\mathbf{x}_r \cdot \mathbf{x}_s) \\
&= \frac{1}{n^2}\sum_s(\sum_r \mathbf{x}_r \cdot \mathbf{x}_r + \sum_r \mathbf{x}_s \cdot \mathbf{x}_s - 2\sum_r \mathbf{x}_r \cdot \mathbf{x}_s) \\
&= \frac{1}{n^2}\sum_s[\sum_r \mathbf{x}_r \cdot \mathbf{x}_r + \sum_r \mathbf{x}_s \cdot \mathbf{x}_s - 2(\mathbf{0} \cdot \mathbf{x}_s)] \\
&= \frac{1}{n^2}\sum_s(\sum_r \mathbf{x}_r \cdot \mathbf{x}_r + n\mathbf{x}_s \cdot \mathbf{x}_s) \\
&= \frac{1}{n^2}(n\sum_r \mathbf{x}_r \cdot \mathbf{x}_r + n\sum_s \mathbf{x}_s \cdot \mathbf{x}_s) \\
&= \frac{1}{n^2}2n\sum_r \mathbf{x}_r \cdot \mathbf{x}_r \\
&= \frac{2}{n}\sum_r \mathbf{x}_r \cdot \mathbf{x}_r
\end{aligned} \tag{3.8}$$

Finally, applying 3.8 on 3.5:

$$\mathbf{B}_{rs} = \mathbf{x}_r \cdot \mathbf{x}_s = -\frac{1}{2}(\delta_{rs}^2 - \frac{1}{n}[\sum_s \delta_{rs}^2 + \sum_r \delta_{rs}^2 - \frac{1}{n}\sum_s\sum_r \delta_{rs}^2]) \tag{3.9}$$

From 3.9, it becomes clear that $\mathbf{B}$ is, in fact, the double centering of the matrix $\mathbf{A} = -\frac{1}{2}\mathbf{\Delta}^2$. I.e., $\mathbf{B} = \mathbf{HAH}$. Spectral decomposition can now be performed onto $\mathbf{B}$, resulting in the matrices $\mathbf{\Sigma}$ and $\mathbf{U}$.

Finally, in order to reduce the dimensionality of the embedding, we can sort the eigenvalues (and their respective eigenvectors, the columns of $\mathbf{U}$) in decrease order and keep only the ones that offer greater variance.

Remark 3.2.1. *As euclidean distances were used to build the dissimilarity matrix* $\mathbf{\Delta}$, $\mathbf{B}$ *is indubitably positive semidefinite, hence* $\sigma_i \geq 0, \forall i \in [0, n)$. *However, negative eigenvalues might appear if other dissimilarity measurement were to be used. In these cases, one might consider to simply ignore such components.*

3.2.2 Formalization of the Multidimensional Scaling Method

Let $\mathbf{X}$ be a data set with n samples and f features and $m \in \mathbb{R}$ the number of dimensions desired for the reduced data set [13].

1. Calculate the dissimilarity matrix $[\delta]_{rs}$, where $\delta_{rs} = \sqrt{\sum_i (x_{ri} - x_{si})^2}$
2. Calculate the matrix $\mathbf{A} = -\frac{1}{2}\delta_{rs}^2$ and $\mathbf{B} = \mathbf{HAH}$, where $\mathbf{H}$ is the centering matrix.
3. Use spectral decomposition to find the matrices $\mathbf{\Sigma}$ and $\mathbf{U}$.
4. Select the m greatest eigenvalues in $\mathbf{\Sigma}$. From these, create the matrices $\mathbf{\Sigma}' = [\sigma'_{m \times m}]$ and $\mathbf{U}' = [u'_{n \times m}]$, where each column i contains the eigenvector associated with σ'_i.
5. Construct the m-dimensional embedding $\mathbf{Y} = \mathbf{U}'\mathbf{\Sigma}'^{\frac{1}{2}}$

3.3 Evaluating Reductions

Although the factors that determine if a reduction is acceptable or not are often influenced by particularities of the problem in hand, the researcher's past experience with MDS and his judgment [38], some measures were developed to attempt to somehow formalize it. One in particular, which recurrently appears in literature, is known as the **Kruskal's stress**.

Intuitively, Kruskal's stress [38] considers reductions that preserve dissimilarities between samples better that the ones which highly distort them. Formally, let $\mathbf{X}_{n \times f}$ be a data set with n samples and f dimensions, $\mathbf{Y}_{n \times p}$ its reduction to p

dimensions, and the dissimilarity measurements δ_{ij} and $\hat{\delta}_{ij}$ defined for all samples i and j in $\mathbf{X}$ and $\mathbf{Y}$, respectively:

$$Stress = [\frac{\sum_i \sum_j (\delta_{ij} - \hat{\delta}_{ij})^2}{\sum_i \sum_j \delta_{ij}^2}]^{\frac{1}{2}}$$

From the formula above, *Stress* is visibly contained in the interval $[0, 1]$, where 0 represents the best possible fit (all dissimilarities are the same), while 1 represents the worse.

Chapter 4

Nonlinear Dimensionality Reduction

Although PCA presents promising results in many cases, hence its great popularity in dimensionality reduction problems, there are many examples in which PCA will fail to understand the structure of the data. Consider the example bellow:

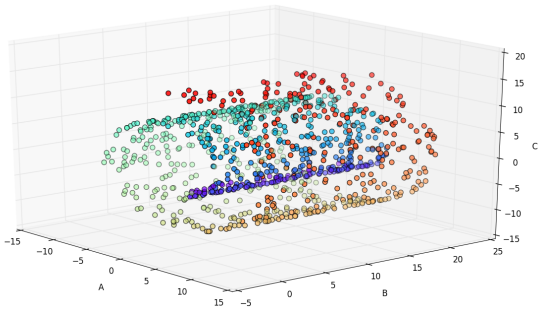


Figure 4.1: The Swiss roll data set.

The application of the PCA algorithm over the Swiss Roll data set results in the following reductions:

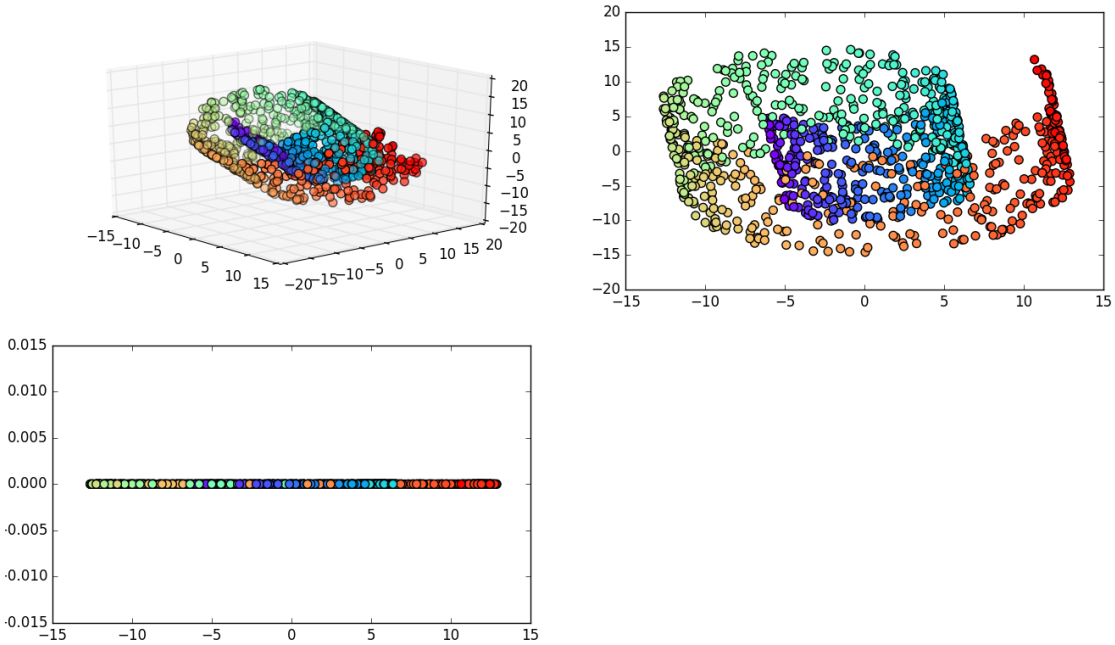


Figure 4.2: The Swiss Roll data set reductions to 3, 2 and 1 dimensions, respectively, using the PCA algorithm.

Looking at figure 4.2 (specially in the last reduction, to a single dimension), it becomes clear that PCA has a big draw back: it assumes that the data lies on a liner subspace [10] and, therefore, applying linear transformations to it will rotate and scale it, without distort the original data structure. When this assumption does not hold, PCA will incorrectly extract the underlying structure, possibly mixing very dissimilar samples.

Clearly, linear dimensionality reduction techniques are not adequate to reduce the Swiss-roll. In fact, it is not adequate to reduce any data set lying on a nonlinear manifold.

4.1 The ISOMAP Algorithm

Firstly suggested by Tenenbaum, de Silva and Langfor, **Isometric Feature Mapping** (or ISOMAP) assumes that the data lies near a smooth manifold. If the assumption is reasonable, it is possible to explore concepts such as neighborhood and local linearity to map the manifold to a linear structure before reducing it

with a linear algorithm.

4.1.1 Study of the ISOMAP Algorithm

As the original data set might be folded, twisted or curved [39], we must first find a suitable linear representation for it.

Let **S** be our original data set, as illustrated in figure 4.3.

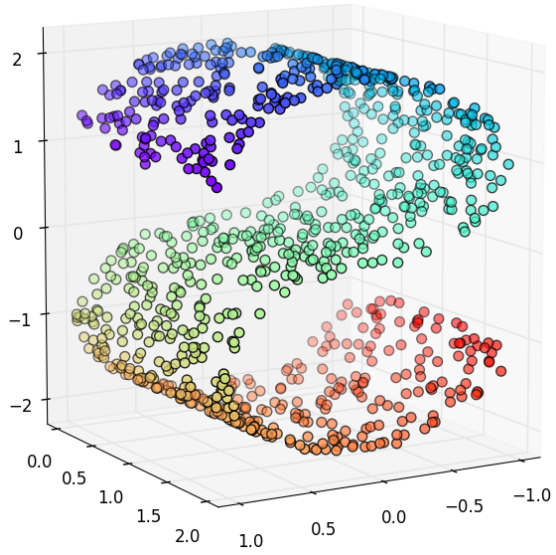


Figure 4.3: The data set **S**, consisting of 1000 samples and 3 features.

Additionally, consider the symmetric undirected weighted graph $G = (V, E)$ and $w\colon E \to \mathbb{R} \mid w(x, y) = \delta_{xy}$, where δ_{xy} is the euclidean distance between the samples $\mathbf{x}$ and $\mathbf{y}$ in $\mathbf{S}$. That is,

$$\delta_{xy} = \sqrt{\sum_i (x_i - y_i)^2}, \forall(\mathbf{x}, \mathbf{y}) \in \mathbf{S} \times \mathbf{S} \mid \mathbf{x} \neq \mathbf{y}$$

Now that only distances were kept, an infinite number of n-dimensional embeddings can be found with **MDS**, as every reduction can be transposed, rotated or reflected while maintaining the original dissimilarities. This does not handle the non-linearity of the data, though, as the original distances strictly constraint the samples to their original pattern. In order to "unfold" the data set, **Nearest**

neighbor search can be performed over G, resulting in the graph G'. Nearest-neighbor search will preserve edges connecting closer samples, hence preserving local (linear) distances, but erase edges connecting samples which are far from each other (non necessarily linear). Obviously, the search parameters (k or ϵ) must be carefully chosen to limit the connectivity of the vertices to a small (linear) neighborhood while maintaining the graph completely connected.

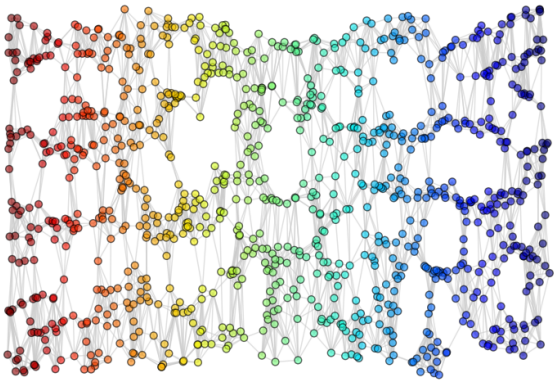
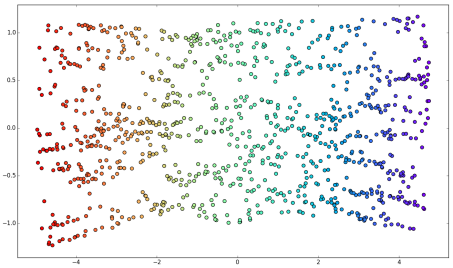


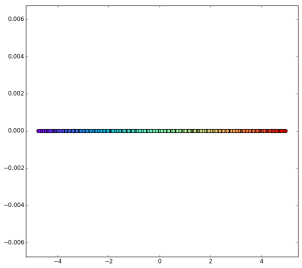
Figure 4.4: The graph G'.

At this moment, not all distances in G' are defined. This can be easily handled, though, by performing the **Floyd-Warshall** algorithm over G', resulting in the shortest-paths graph G''. Alternatively, G'' can be achieved by performing **Dijkstra's algorithm** for all nodes and joining all shortest-path trees found.

Finally, G'' is a euclidean graph which roughly lies on a linear subspace. Furthermore, the adjacency matrix associated to G'' contains not the distance induced by the L_2 norm, but the geodesic pairwise distances [11]. The **MDS** method can now be used to construct a representation in sub-spaces of the $\mathbb{R}^n$. **S**, specifically, can be reduced to the $\mathbb{R}^2$ or $\mathbb{R}$:



(a) Reduction to 2D.



(b) Reduction to 1D.

Figure 4.5: **S**' reductions.

4.1.2 Formalization of the ISOMAP Algorithm

Let $\mathbf{X}$ be the original data set and $p \in \mathbb{R}$ the number of dimensions desired for the reduced data set [39],

1. Construct the weighted graph G from the distances pairwise $\delta_{xy}, \forall(\mathbf{x}, \mathbf{y}) \in \mathbf{X} \times \mathbf{X}, \mathbf{x} \neq \mathbf{y}$ and find the graph G' by applying the **nearest-neighbor algorithm** on the graph G.

2. Compute the shortest path graph G'' between all pairs of nodes from graph G'. This might be done by the **all-pairs Dijkstra's** or by the **Floyd-Warshall** algorithm.

3. Use G'' to construct the p-dimensional embedding using the **MDS** algorithm.

4.1.3 Computational Complexity

If n is the number of training samples, $f \in \mathbb{R}$ the number of features and $k \in \mathbb{R}$ the number of nearest neighbors:

1. The time complexity associated with building the neighborhood graph is $O(n^2)$, whereas the space necessary to represent the distances kept is n^2.

2. Algorithms for finding shortest path graph can be executed in-place and do not increase space complexity. Regarding time complexity, however:

   (a) Dijkstra's algorithm implementation using Fibonacci Heaps have time complexity $O(nk + n \log n)$. As the algorithm must be calculated for each node, this step has time complexity $O[n^2(k + \log n)]$.

   (b) Alternatively, using the Floyd-Warshall algorithm, this step has time complexity equals to $O(n^3)$.

3. Taking $O(n^3)$ time steps to calculate the spectral decomposition, MDS clearly is the bottleneck of the entire algorithm [10, 40]. As for space complexity, $O(f^2 + nf)$ is required for storing the matrices $\mathbf{\Sigma}$ and $\mathbf{U}$.

The time complexity of the ISOMAP algorithm (when using Dijkstra's) is, therefore, $O[n^2 + n^2(k + \log n) + n^3]$, and the space complexity is $O(n^2 + f^2 + nf)$.

Experiment 4.1.1 (Timing the ISOMAP Algorithm)**.** *Consider the data set* ***Digits*** *with 1797 samples and 64 features. The following table indicates the time duration for each step of the ISOMAP algorithm when reducing Digits to 3 dimensions:*

Pairwise distances from data set	.44 s
K Nearest Neighbors Search	1.3 s
All Pairs Dijkstra's	51.44 s
MDS	118.84 s
Total Time	172.69 s

Table 4.1: Listing of time spent on each step of the ISOMAP algorithm.

As expected, most of the time (68.81%) was spent executing MDS.

In practice, ISOMAP's time complexity of $O[n^2 + n^2(k + \log n) + n^3]$ makes it unsuitable for data sets with high number of samples. This is illustrated in Shi and Gu's experiments: ISOMAP could not reduce data sets with more than 6000 samples in reasonable time [41].

Experiment 4.1.2. *Let* ***Spam*** *be a data set with 4601 samples and 57 features. The table bellow describes the time necessary for the implemented ISOMAP and sk-ISOMAP (the algorithm from the scikit-learn library) to reduce Spam to 3 dimensions:*

Algorithm	Time
ISOMAP	29.9 m
sk-ISOMAP	10.76 s

Table 4.2: Timing the implemented ISOMAP and scikit-learn's implementation.

The original ISOMAP implemented required almost 30 minutes to reduce Spam. Surprisingly, scikit-learn's implementation executed incredibly fast. A set of factors contribute for this result:

- ***Ball Tree*** *is used for efficient neighbor search (requiring only* $O(fn \log k \log n)$ *time steps).*

- *Many components in scikit-learn have python signatures, but are actually implemented in C, considerably increasing performance.*
- *ISOMAP is implemented as a kernel for the* ***KernelPCA*** *method. Further investigation is done in section 4.1.4.*

4.1.4 Extensions

ISOMAP as a variation of Kernel PCA

While "classic" PCA aims to find the principal components of the covariance matrix $\mathbf{\Sigma} = \frac{1}{n}\mathbf{X}^\top\mathbf{X} = \frac{1}{n}\sum_k x_{ik} \cdot x_{kj}, \forall(i,j) \in [0, |\mathbf{S}|]^2$ of a centered data set $\mathbf{X}$, it can be easily modified to attempt to do the same, but over the matrix $\mathbf{\Sigma}' = \frac{1}{n}\sum_i \phi(x)_i \cdot \phi(y)_i = \frac{1}{n}\sum_i K(x,y)$, where $\phi\colon \mathbb{R}^f \to \mathbb{R}^p$ is a function that projects vectors in the original f-space to a different p-space and $K\colon \mathbb{R}^f \times \mathbb{R}^f \to \mathbb{R}$ is a kernel function. Such modification is named Kernel PCA.

In his paper, Ghodsi describes how MDS (and therefore, ISOMAP) is equivalent to Kernel PCA, given it is using the following kernel function [11]:

$$\mathbf{K}_{ISOMAP} = -\frac{1}{2}\mathbf{H}\delta^{\mathbf{2}}\mathbf{H}$$

Where $\mathbf{H}\delta^{\mathbf{2}}\mathbf{H}$ is the double centered squared dissimilarity matrix, i.e, the double centered squared geodesic distances between samples found from the shortest-path graph.

An advantage of substituting the MDS algorithm by Kernel PCA is that the latter provides an embedding, i.e., a transformation matrix from the original space to the reduced one; whereas MDS only embeds the data set [42]. However, Silva and Tenenbaum have shown that MDS's result can be used to reduce new samples (L-MDS) [40]. More will be discussed ahead, when L-ISOMAP is presented.

Figure 4.6 displays the reduction of the Swiss-roll data set done by the original implementation of ISOMAP (using MDS) and scikit-learn's implementation (using Kernel PCA). Numerically, they are identical (i.e., save from difference which tends to 0).

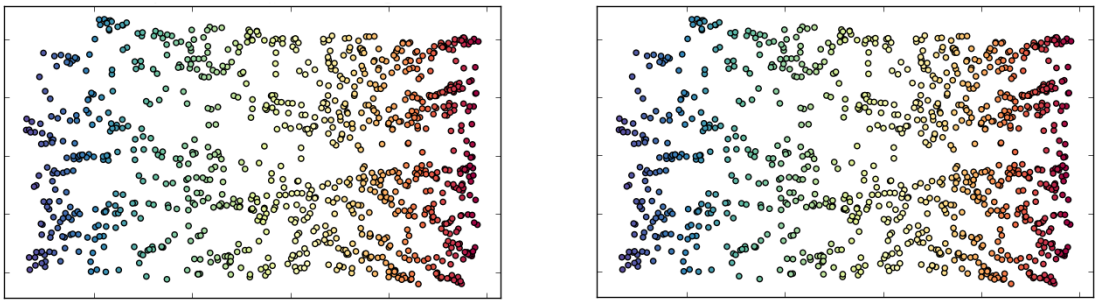


Figure 4.6: The embedded swiss-roll data set by the original and scikit-learn's implementation.

Incremental ISOMAP

The original ISOMAP algorithm reduces the data set in a single execution. Even its implementation with Kernel PCA will not update the reduction model if new data points are given, only reduce these points using the eigenvectors previously found. Therefore, updating ISOMAP reduction model requires recalculating it at every new data "batch", which is very costly.

In order to allow ISOMAP to continuously learn from a stream of data, Incremental ISOMAP was proposed [43]. Relying on the fact that new data points will unlikely afect a large subset of vertices, it selectively updates the structures kept by ISOMAP. The update is performed as follows.

Let $[\mathbf{X}]_{n\times f}$ be a data set with n samples and f features, $\delta_{\mathbf{ab}}$ be the dissimilarity measure between two samples $\mathbf{a}$ and $\mathbf{b}$, $\mathbf{x}_i$ be a sample in the nearest neighbor graph, $\mathbf{x}_j$ a neighbor of $\mathbf{x}_i$ such that their dissimilarity is higher among all $\mathbf{x}_i$'s neighbors. Additionally, let $[\mathbf{Y}]_{n\times d}$ be the embedding of $\mathbf{X}$ in the d-dimensional space. Finally, consider $\mathbf{x}_{n+1}$ a new sample to increment in the reduction model.

1. The new sample $\mathbf{x}_{n+1}$ will replace $\mathbf{x}_j$ as a neighbor of $\mathbf{x}_i$ if $\delta_{\mathbf{x}_i\mathbf{x}_{n+1}} < \delta_{\mathbf{x}_i\mathbf{x}_j}$. A list of added or removed edges should be stored.

2. The update of the shorest-path graph:

   (a) For every edge $(\mathbf{x}_i, \mathbf{x}_j)$ removed, the shortest-path between any two vertices $(\mathbf{x}_a, \mathbf{x}_b)$ that contains $(\mathbf{x}_i, \mathbf{x}_j)$ must be updated. That involves

applying a slightly modified Dijsktra's algorithm, which will focus on finding the paths from a given source $\mathbf{u}$ and its unreached destination vertices, i.e., vertices $C(\mathbf{u}) = \{\mathbf{c}_i\}$ whose shortest-paths $(\mathbf{u}, \mathbf{c}_i)$ contained a removed edge.

(b) The geodesic distances between all vertices and $\mathbf{x}_{n+1}$ must then be calculated. Let A be the set of added edges:

$$g[n+1, i] = g[i, n+1] = \min_{j|(n+1,j)\in A}(g[i, j] + w(j, n+1)), \forall i.$$

All the added edges are incident on vertex $\mathbf{x}_{n+1}$, naturally. Calculating the shorest-path consists on considering two whichever edges $(\mathbf{a}, \mathbf{x}_{n+1}), (\mathbf{x}_{n+1}, \mathbf{b})$ and relaxing the edge $(\mathbf{a}, \mathbf{b})$ by considering the path $\mathbf{a} \to \mathbf{x}_{n+1} \to \mathbf{b}$.

3. Finally, the eigenvectors/eigenvalues of the inner product matrix $\mathbf{B}_{new}$ (which contains the recently added sample $\mathbf{x}_{n+1}$) can be found through an iterative scheme, where "a good initial guess for the subspace of dominant eigenvectors is the column space of $\mathbf{Y}$" [43] (the embedding before the $\mathbf{x}_{n+1}$ increment). A better eigen-space for $\mathbf{B}_{new}$, and thus $\mathbf{Y}_{new}$ can be found by Subspace iteration together with Rayleigh-Ritz acceleration [43]:

   (a) Let $\mathbf{Z} = \mathbf{B}_{new}\mathbf{Y}$. Perform QR-decomposition on $\mathbf{Z}$: $\mathbf{Z} = \mathbf{B_{new}}\mathbf{Y} = \mathbf{QR}$.

   (b) Take $\mathbf{V} = \mathbf{Q}$ (remember that $\mathbf{Q}$ is intrinsically an orthogonal matrix) and $\mathbf{Z}^* = \mathbf{V}^\top \mathbf{B}_{new}\mathbf{V}$. Perform spectral-decomposition of $[\mathbf{Z}^*]_{d\times d}$ to find $\mathbf{\Lambda} = diag(\lambda_i)$ and $\mathbf{V} = [\mathbf{v}_i]_{d\times d}$. This is significantly faster than directly decomposing $\mathbf{B}_{new}$, considering the dimensions of both matrices.

   (c) $\mathbf{Y}_{new} = [\mathbf{Y}, \mathbf{y}_{n+1}]^\top$, where $\mathbf{y}_{n+1}$ is the d-dimensional embedding of $\mathbf{x}_{n+1}$:

$$\mathbf{y}_{n+1} = (\frac{1}{\sqrt{\lambda_1}}v_1 \cdot f, \ldots, \frac{1}{\sqrt{\lambda_d}}v_d \cdot f)$$
$$2f_i \approx \frac{1}{n}[\sum_j g_{ij}^2 - \sum_{lj} g_{lj}^2 + \sum_l g_{l,n+1}^2] - g_{i,n+1}^2$$

In regard to time complexity, considering q the maximum degree of the vertices in the graph, F is the set of pairs whose shortest path contained a removed edge and H is the set of vertex pairs whose geodesic distances were modified, Incremental

ISOMAP will require $O[q(|F| + |H|)]$ time steps in phase 1; $O[n^2(\log n + q)]$ in 2; and $O(n^2)$ in 3. The time complexity of the algorithm is, therefore, $O[q(|F| + |H|) + n^2(\log n + q + 1)]$.

L-ISOMAP

ISOMAP's time complexity dependents mainly on the number of samples. Remember: for N samples, ISOMAP will require $O[n^2(\log n + k)]$ and $O[n^3]$ time operations to compute the shortest-paths and eigenvectors/values, respectively.

Landmark ISOMAP aims to reduce the complexity of the algorithm by finding the dissimilarities between n landmarks (a small subset of samples) and all the original samples. Its complexity can then be reduced to $O[n^2(\log N + k) + n^2 N]$. If $n \ll N$, this represents a expressive performance improvement [40].

Let $\mathbf{X}$ be the initial data set of interest, l be the dimensionality to which $\mathbf{X}$ should be reduced and $\epsilon > 0 \in \mathbb{N}$ a number to "ensure stability" [40]. L-ISOMAP is defined as it follows.

1. Select n random samples from $\mathbf{X}$, such that $n > l + 1 + \epsilon$.
2. Calculate the nearest-neighbors of each selected samples (all the original samples should be considered as neighbors).
3. Calculate the shortest-paths from the neighborhood graph obtained in the previous step and the dissimilarity matrix $[\delta]_{n \times N}$.
4. Perform Landmark MDS, a slightly modified MDS:
   (a) Apply classical MDS to the submatrix of landmarks $[\delta]_{n \times n}$.
   (b) Embbed the remaining points through:

$$\mathbf{y} = -\frac{1}{2} L^{-1}(\mathbf{\Delta}_x - \bar{\mathbf{\Delta}}_n)$$

$$\mathbf{y} = -\frac{1}{2} \begin{pmatrix} \frac{v_1^T}{\sqrt{\lambda_1}} \\ \frac{v_2^T}{\sqrt{\lambda_2}} \\ \cdots \\ \frac{v_n^T}{\sqrt{\lambda_n}} \end{pmatrix} (\mathbf{\Delta}_x - \bar{\mathbf{\Delta}}_n)$$

Where $\boldsymbol{\Delta}_x$ is the square dissimilarity column vector from sample $\mathbf{x}$ to all landmarks and $\bar{\boldsymbol{\Delta}}_n$ the vector containing the column mean of $\boldsymbol{\Delta}_n$ (the square dissimilarity matrix between landmarks).

p-ISOMAP

The embedding found by ISOMAP is strongly depentend of the parameters k or ϵ. When these parameters are too small, the neighborhood graph extracted might be disconnected, causing the dissimilarity between some vertices to be infinite. On the other hand, when the parameters are too big, vertices from different neighborhoods are prone to stay connected, preventing the manifold to be "unfolded". Hence the importance of choosing appropriate parameters. However, in many cases, these "appropriate" paramaters might not be evident. Hence the interest of developing a parameterless ISOMAP.

p-ISOMAP is an extension that automatically selects the parameters k or ϵ [44]. Assuming an result of ISOMAP for a particular parameter value is available, a new parameter will be considered and p-ISOMAP will iteratively update the soluton by:

1. Adding/removing neighbors for every vertex in the neighborhood graph regarding the new parameter. This operation has time complexity $O(n \max_i |\boldsymbol{\Delta} e_i|)$, where $|\boldsymbol{\Delta} e_i|$ is the number of inserted/removed edges associated with sample $\mathbf{x}_i$.

2. Update the shortest-paths described by the pairs of vertices in $F \subset Q$, where Q is the set of all pairs of vertices and F is the set of pairs associated with the updated edges in the previous step.

   (a) If the parameter value increased in comparison to the previous one, edges might have been added. Updating the shortest-path tree consists in simply relaxing edges associated with any of the vertices in the pairs of F. The time complexity is loosely bounded by $O(|A|q|F|)$, where $|A|$ is the number of inserted edges and q is the maximum degree of vertices in G.

   (b) If the parameter was decreased, then edges might have been removed and all pairs of vertices whose shortest-paths contain those edges are

added to F. Identifying F can be done in $O(n^2)$ times steps to execute, given some optimzation (see [44]). Now, Dijkstra's can be selectively performed, requiring $O(n \max_i(|E'_i| \log |V_d(i)| + |E''_i|)$, where, given a vertex $\mathbf{x}_i$, $V_d(i) = \{\mathbf{x}_j | (\mathbf{x}_i, \mathbf{x}_j) \in F\}$, E'_i is the set of edges connecting whichever pair of vertices in $V_d(i)$ and E''_i is the set of edges connecting the vertices $\mathbf{x}_a$ and $\mathbf{x}_b$ such that $\mathbf{x}_a$ is in $V_d(i)$ and $\mathbf{x}_b$ is not.

3. Finally, the last step is to update the eigenvectors and eigenvalues associated with the matrix $\mathbf{B}$. They can be approximated using the Lanczos algorithm, an iterative refining method that takes an initial solution candiadate. Officially, the time complexity of Lanczoz algorithm is $O(n^2)$ [45]. Yet, its timeframe for convergence also strongly depends on the solution candidate given. Assuming the previous inner matrix and the current one are similar, one might use the previous eigenvalues/vectors found as candidate. In this scenario, Lanczoz's will likely finish much faster than other algorithms.

The experiments presented in [44] have shown p-ISOMAP consistently maintaining a better performance than regular ISOMAP for data sets with a samples count varying between 500 and 3500. Additionally, the experiments have also demonstrated how p-ISOMAP performance is sensible to high variances of k and ϵ values.

4.1.5 Evaluating Reductions

ISOMAP's aim is to "unfold" the data set and, of course, it might distort dissimilarities between samples in the process. Hence drastically increasing Kruskal's stress. Clearly, this measure is unsuitable for evaluating ISOMAP. As proposed by L. Shi and J. Gu, Kruskal's stress can be easily updated to match ISOMAP nature. That is, to preserve dissimilarities within neighborhoods [41]:

Let $\mathbf{X}_{n \times f}$ be a data set with n samples and f dimensions, $\mathbf{Y}_{n \times p}$ its reduction to p dimensions, and δ_{ij} and $\hat{\delta}_{ij}$ the dissimilarity measurements defined for all samples $\mathbf{i}$ and $\mathbf{j}$ in both $\mathbf{X}$ and $\mathbf{Y}$. Furthermore, given a sample $\mathbf{i}$ and $\beta \in \mathbb{N} \mid \beta \leq n$, take $\Omega = K \bigcup \Theta$, where K is the set containing the nearest neighbors of i and Θ is a

set of random samples such that $|\Theta| \leq \beta \leq n$:

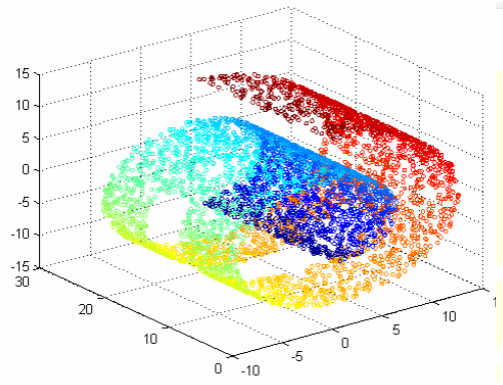
$$Stress_{ISOMAP} = [\frac{\sum_i \sum_{j \in \Omega} (\delta_{ij} - \hat{\delta}_{ij})^2}{\sum_i \sum_{j \in \Omega} \delta_{ij}^2}]^{\frac{1}{2}}$$

4.1.6 Applicability and Limitations

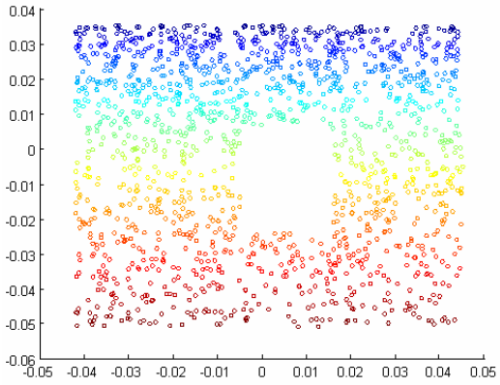
From figure 4.5 and the experiment shown in section 5, it is quite clear that ISOMAP outperforms PCA on the data set **S**. Unfortunately, this cannot be projected for a generic case considering the strict and artificial nature of **S**. In real-world data sets, the application of ISOMAP may lead to poor low-dimensional embeddings [46]. Bellow are listed some of the issues that have great influence over ISOMAP's results:

Manifold Assumption It refers to the initial assumption that the data lies on a low-dimensional manifold. Although wildly exploited by many authors, it is difficult to assert whether such assumption holds or not in real-world data sets [47]. Furthermore, even if the data roughly lies on a manifold, discontinuities in the data pattern can characterize the manifold as non-smooth. In these situations, graphs with edges that disrespect locality would be extracted and, hence, poor low-dimensional representations are likely to be produced.

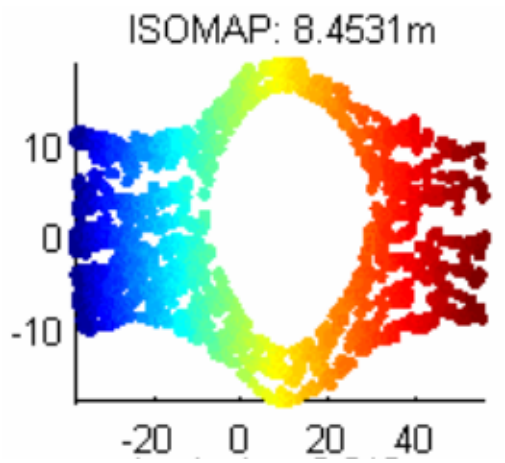
Manifold Convexity ISOMAP assumes that the data is geodesically convex [48]. This issue becomes clear when dealing with data sets with "holes" in it that occupy sufficiently large areas, resulting in a neighborhood graph with great disconnected areas that require paths with great curvature to get around. Lerman demonstrated how reducing non-convex data sets with ISOMAP can easily yield distorted results [49]:



(a) The non-convex Swiss-roll [6].



(b) The expected reduction [6].

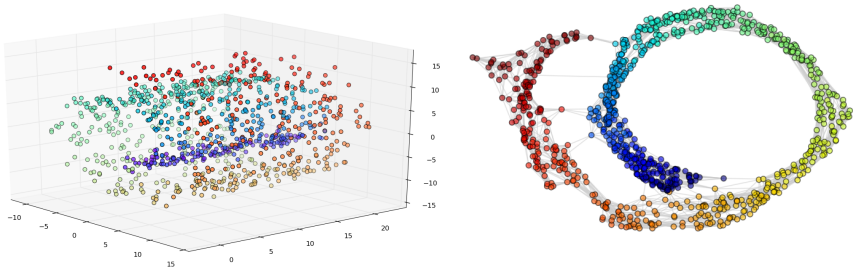


(c) The actual reduction [6].

[6]From "Manifold Learning Techniques: so which is the best?" by G. Lerman, *University of California, Los Angeles*. Available at: `http://math.ucla.edu/~wittman/mani/mani_presentation.pdf`.

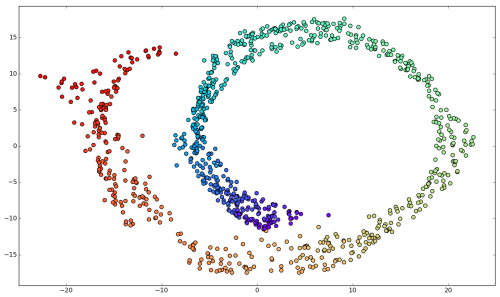
Noise Noise in data is expressed by samples that somehow diverge from their neighborhood (outliers). As these samples become farthest from its original neighborhood, the chances of being linked to samples from other neighborhood increase, possibly decreasing the quality of the solution. A possible solution is to remove these samples during the pre-processing stage [46].

The image bellow illustrates an attempt to use ISOMAP to reduce the $\mathbf{S}wiss_{n=.4}$, which is the Swiss-roll data set subjected to a noise factor of .8:



(a) The data set $\mathbf{Swiss}_{n=.8}$.

(b) The neighborhood graph extracted from $\mathbf{Swiss}_{n=.8}$.



(c) $\mathbf{Swiss}_{n=.8}$ reduced.

Figure 4.8: The ISOMAP applied on a noisy data set.

Observe how some outliers were sufficiently far from their original neighborhood in the graph 4.8b, to the point that Nearest-Neighbor Search maintained the edge between them and samples of completely different colors. MDS then attempted to maintain this dissimilarity, resulting in a deformed reduction (figure 4.8c).

Chapter 5

Experiments

This chapter will cover the experiments implemented throughout this project, which focused on reducing data sets with the presented algorithms and evaluating these reductions. Once again, it is quite difficult to evaluate reductions, but a set of metrics were adopted to attempt to understand them as much as possible.

First, let $\mathbf{X}_{n \times f}$ be a data set with n samples and f features, and $\mathbf{D} \subset \mathbb{N} \mid d \in \mathbf{D} \implies d \leq f$ a set of dimensions to which $\mathbf{X}$ should be reduced. Additionally, consider the score to be the class prediction accuracy or the coefficient of determination R^2, depending on the nature (classification or regression) of the problem of interest. Then, the sequence bellow represents a default format followed by all experiments:

1. The data set $\mathbf{X}$ is loaded and its first three features are plotted, given an initial idea of how the data is distributed for those features.

2. $\mathbf{X}$ is reduced to 3, 2 and 1 dimensions. Each reduction is plotted for visualization.

3. Kruskal's stress is calculated for $\mathbf{X}$.

4. In an attempt to evaluate local dissimilarity preservation, a 1-NN classifier or regressor is trained with 80% of $\mathbf{X}$'s samples and tested with the other 20%. The score is kept for future reference.

5. Grid search is executed over $\mathbf{X}$ using a SVM classifier or regressor. This step evaluates how a "real world" algorithm would perform over the original data

set.

6. for $d \in \mathbf{D}$:

   (a) $\mathbf{X}$ is embedded into the $\mathbb{R}^d$, creating the data set $\mathbf{Y}_{n \times d}$.

   (b) Kruskal's stress is calculated for $\mathbf{Y}$.

   (c) A 1-NN classifier or regressor is trained with the same samples used in step 4, and tested with all the others. The score is compared to the one retrieved in step 4.

   (d) Grid Search is executed over $\mathbf{Y}$ and the score is compared to the one presented in step 5.

5.1 K Data Set

The synthetic data set $\mathbf{K}$, with 1000 samples and 2 features.

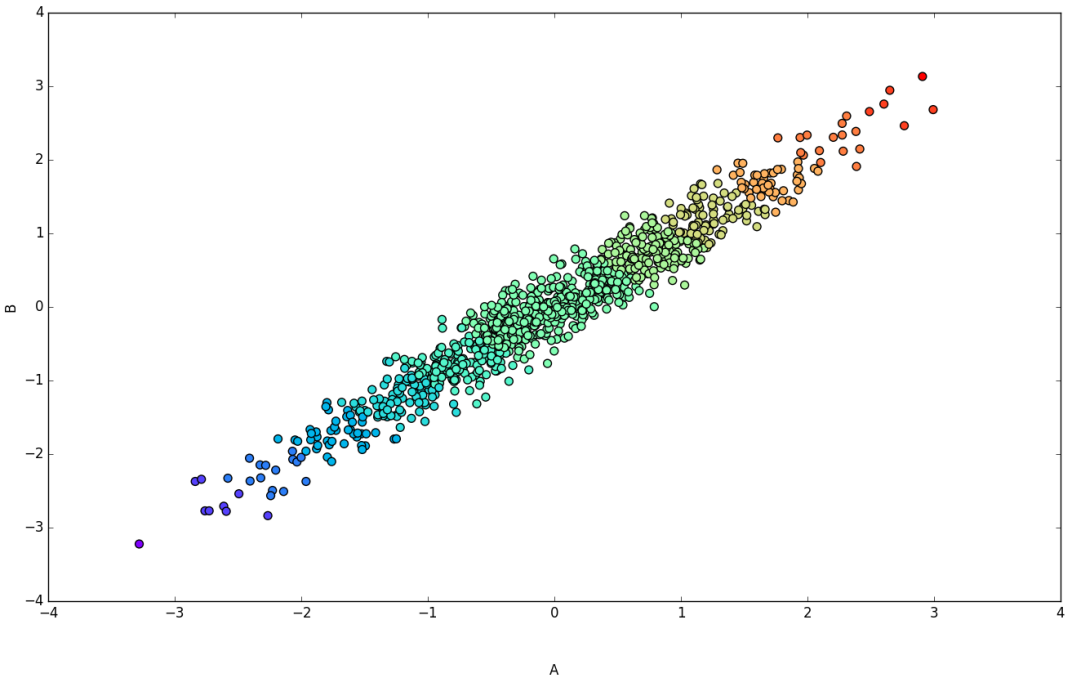


Figure 5.1: The $\mathbf{K}$ data set.

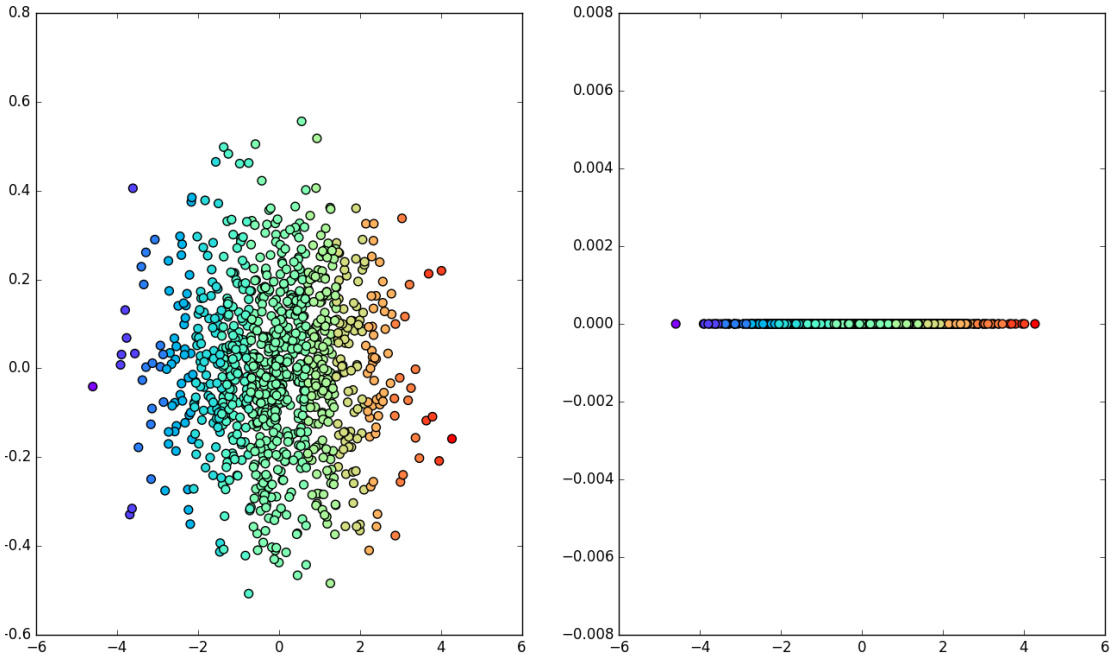


Figure 5.2: The reductions of **K** to 2 and 1 dimension, respectively.

Figure 5.2 illustrates the results of PCA algorithm application over **K** data set. Notice that, for the second application, it correctly chose to discard the vertical dimension, as the samples offer less variability in this component.

	Original $\mathbb{R}^2$	$\mathbb{R}^2$	$\mathbb{R}$
1-NN	.99	.99	1
Score	.982	.982	.994
Best parameters	'C': 1000, 'k': 'linear'	'C': 1000, 'k': 'linear'	'C': 1000, 'g': 10, 'k': rbf
GridSearch time	2.55 s s	2.93 s	2.94 s
Reduction time	-	1.97 s	1.93 s
Kruskal's stress	-	0	.0399
Data size	15.62 KB	15.62 KB	7.81 KB

Table 5.1: Description of predictions and reduction performance for **K**.

From the table above, one can see that, given the linear nature of **K**, PCA was an adequate method to reduce it, as both 1-NN regressor and SVM created

very accurate models. It is also possible to observe how Kruskal's stress might not be an appropriate quality measurement, given a specific domain. Indeed, **K**'s stress increase resulted from the removal of one of the components did not imply on prediction accuracy decrease.

5.2 The Iris Flower Data Set

The Iris Flower data set, as presented in section 2.1, containing 150 samples and 4 features.

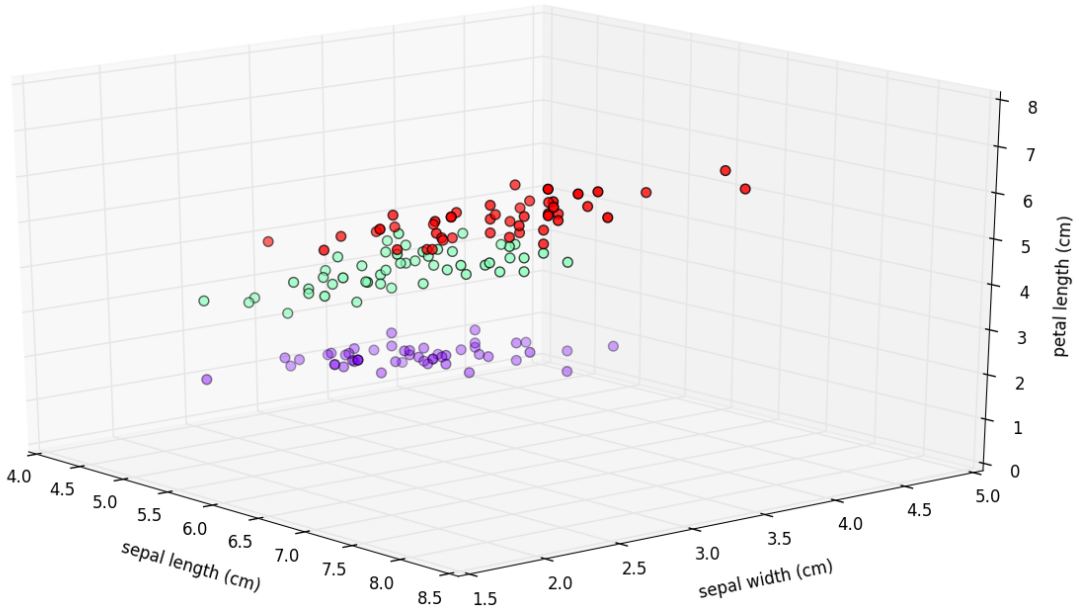


Figure 5.3: The Iris Flower data set.

Reducing the Iris Flower Data Set With the PCA Algorithm

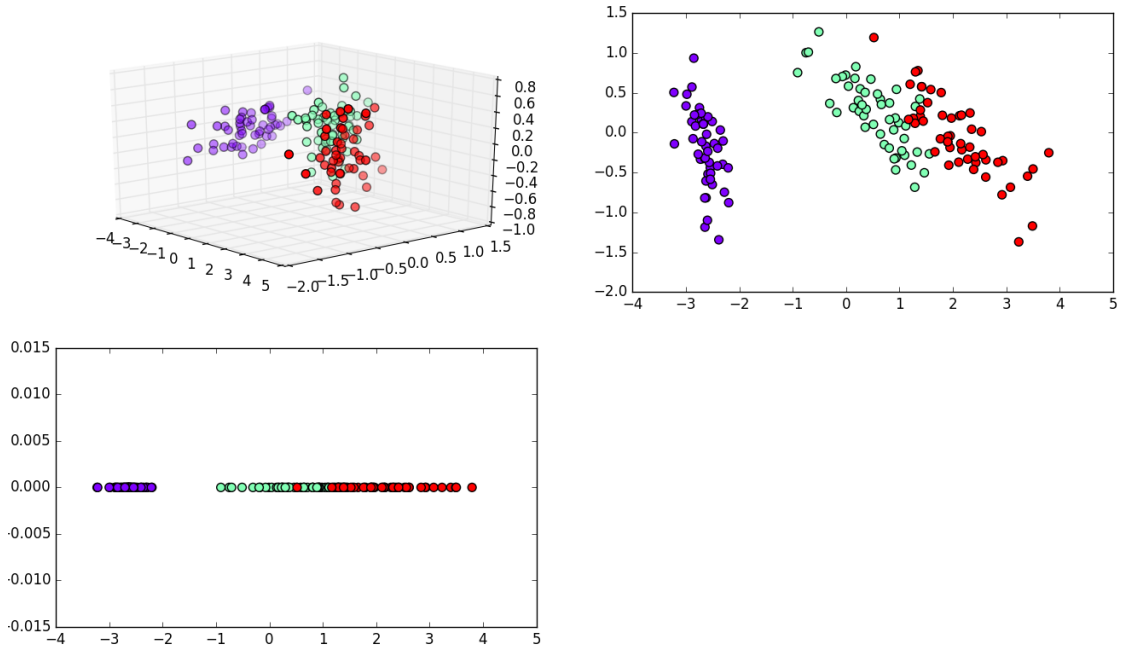


Figure 5.4: The reductions of Iris Flower to 3, 2, and 1 dimensions, respectively, using the PCA algorithm.

One can infer from figure 5.4 that classes are still somewhat organized in different clusters, even in the reductions for 2 and 1 dimensions. The Application of PCA therefore results in fairly good visualizations.

	Original	$\mathbb{R}^3$	$\mathbb{R}^2$	$\mathbb{R}$
1-NN	1	1	1	.87
Score	.99	.98	.97	.95
Best parameters	g: .01, k: rbf, C: 100	k: linear, C: 10	g: 10, k: rbf, C: 1	g: .1, k: sigmoid, C: 1
GridSearch time	2.14 s	2.64 s	2.42 s	2.64 s
Reduction time	-	.17 s	.17 s	.16 s
Kruskal's stress	-	.01	.04	.11
Data size	4.69 KB	3.52 KB	2.34 KB	1.17 KB

Table 5.2: Description of predictions and reduction performance for the Iris flower and PCA algorithm.

Kruskal's stress kept bellow 11% for all reductions. Indeed, variations in the learning phase performance were small: 1-NN and SVM score were high along all reductions (except for the last one, perhaps, where 1-NN decreased to .87).

Reducing the Iris Flower Data Set With the ISOMAP Algorithm

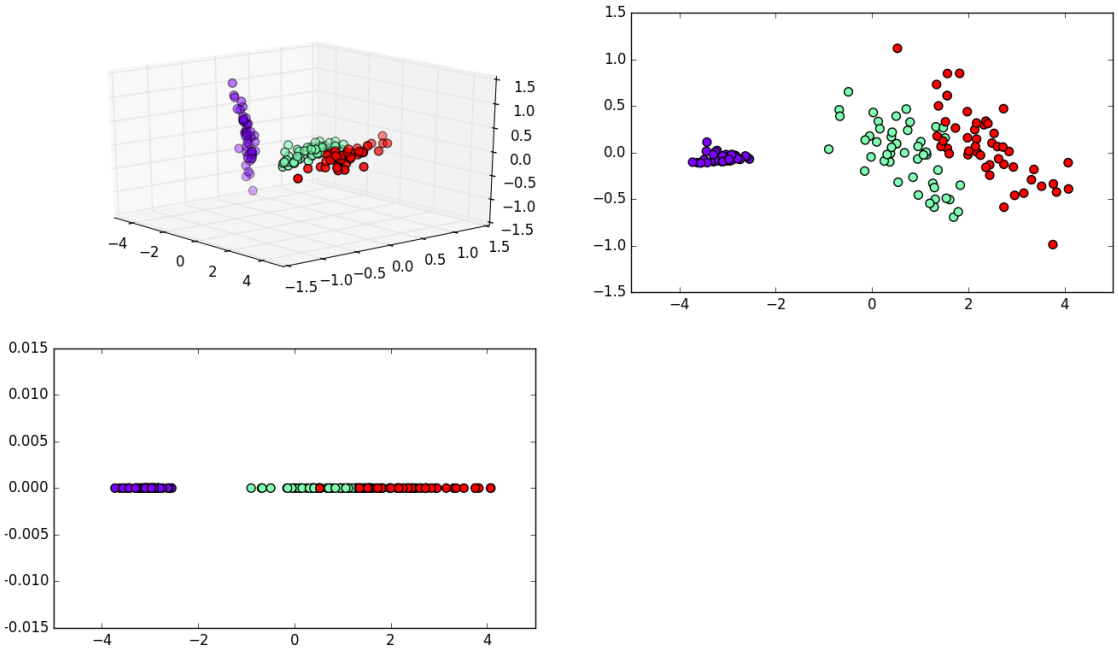


Figure 5.5: The reductions of Iris Flower to 3, 2, and 1 dimensions, respectively, using the ISOMAP algorithm.

It is clear, from figure 5.5, that the ISOMAP algorithm has also resulted in acceptable reductions of the Iris Flower data set, where classes kept a similar organization compared to the reductions made by the PCA algorithm.

	Original	$\mathbb{R}^3$	$\mathbb{R}^2$	$\mathbb{R}$
1-NN	1	1	1	.87
Pred. accuracy	.99	.98	.98	.93
Best parameters	k: rbf, g: 0.01, C: 100	k: linear, C: 10	k: linear, C: 10	k: sigmoid, g: 0.1, C: 1
GridSearch time	2 s	2.55 s	2.42 s	2.37 s
Reduction time	-	1.28 s	1.3 s	1.27 s
Kruskal's stress	-	.15	.15	.16
Data size	4.69 KB	3.52 KB	2.34 KB	1.17 KB

Table 5.3: Description of predictions and reduction performance for the Iris flower and ISOMAP algorithm.

The results were very similar to the ones obtained when applying the PCA algorithm. The differences consist on the Kruskal's stress, now limited to 16% and the reduction time, which has consistently increased due to the higher computational cost of the ISOMAP algorithm.

5.3 The Digits Data Set

Digits data set is composed by 1797 samples, 64 features and 10 classes. Each sample is a 8x8 image of a hand-written digit from 0 to 9.

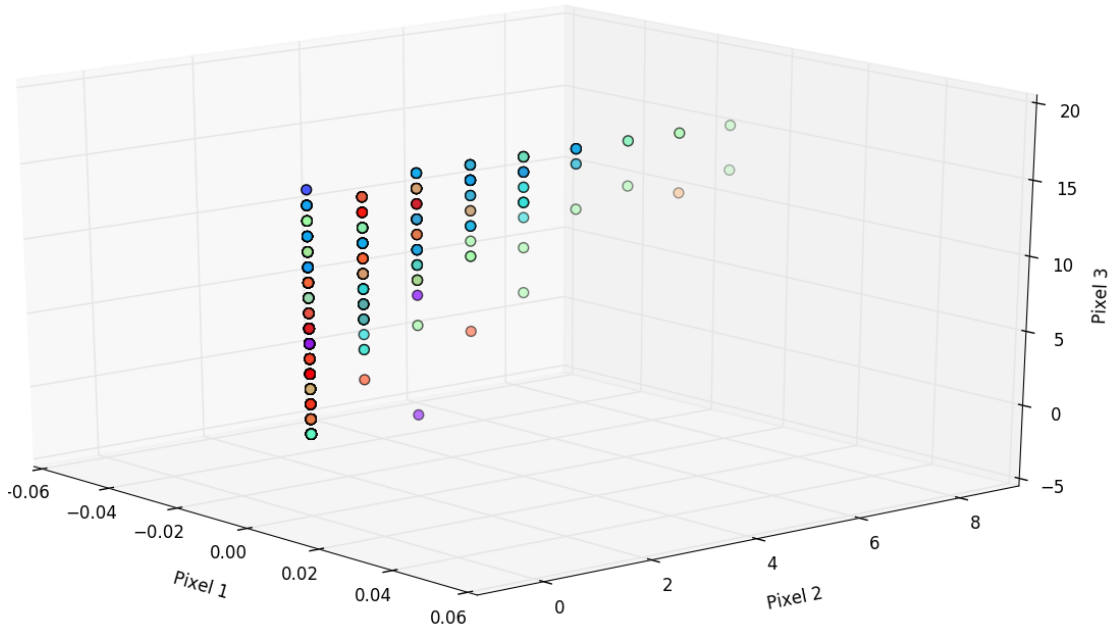


Figure 5.6: The Digits data set.

Reducing the Digits Data Set With the PCA Algorithm

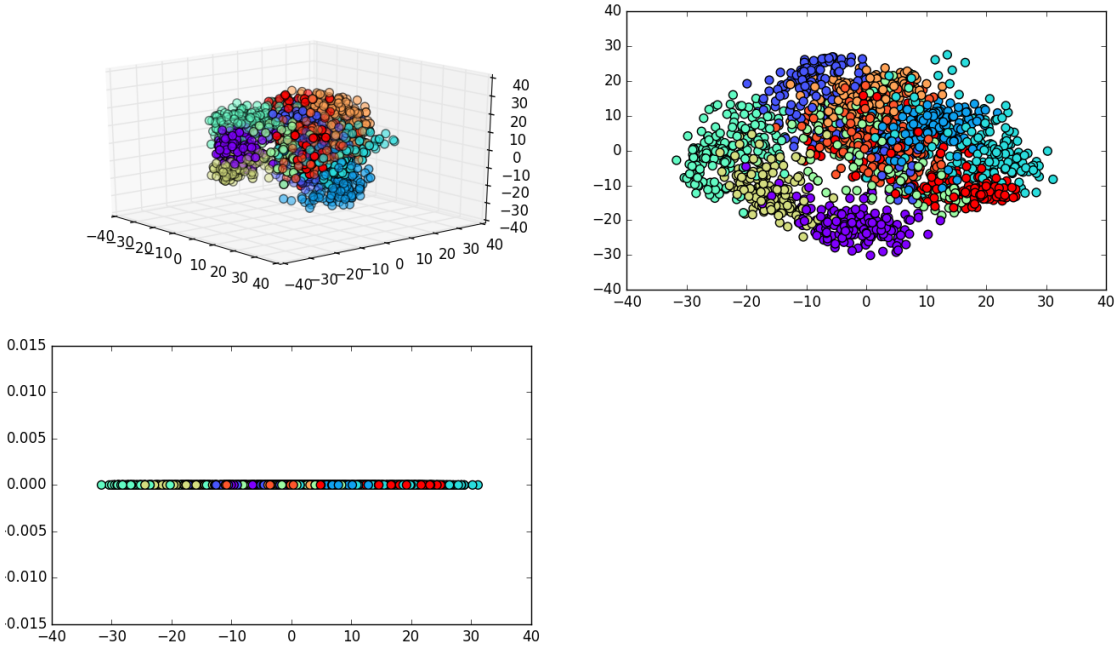


Figure 5.7: The reductions of Digits to 3, 2 and 1 dimensions, respectively, with the PCA algorithm.

From figure 5.7, we have glimpse of Digits's samples structure. The many classes' clusters are visible in the 3D, although they become highly mixed in the 2D and 1D, yielding a confusing representation.

	Original	$\mathbb{R}^{10}$	$\mathbb{R}^3$	$\mathbb{R}^2$	$\mathbb{R}$
1-NN	.99	.97	.68	.56	.28
Score	.98	.95	.74	.64	.39
Best parameters	k: rbf, C: 10, g: .001	k: rbf, C: 10, g: .001	k: rbf, C: 10, g: .01	k: rbf, C: 1, g: .01	k: rbf, C: 10, g: .001
GridSearch time	11.75 s	26.97 s	150.91 s	125.64 s	104.41 s
Reduction time	-	6.02 s	6.54 s	6.05 s	6.1 s
Kruskal's stress	-	.16	.42	.54	.71
Data size	898.5 KB	140.4 KB	42.1 KB	28.08 KB	14 KB

Table 5.4: Description of predictions and reduction performance for Digits and ISOMAP algorithm.

Notice that it was possible to eliminate 54 dimensions, consistently reducing the data set size, and only suffering 3% of prediction accuracy loss. The score drastically decreased, however, when more dimensions were removed. Furthermore, reductions to less than 10 dimensions would generate highly mixed samples, requiring more time for grid searching. Finally, it is also clear that score was inversely proportional to stress.

Reducing the Digits Data Set With the ISOMAP Algorithm

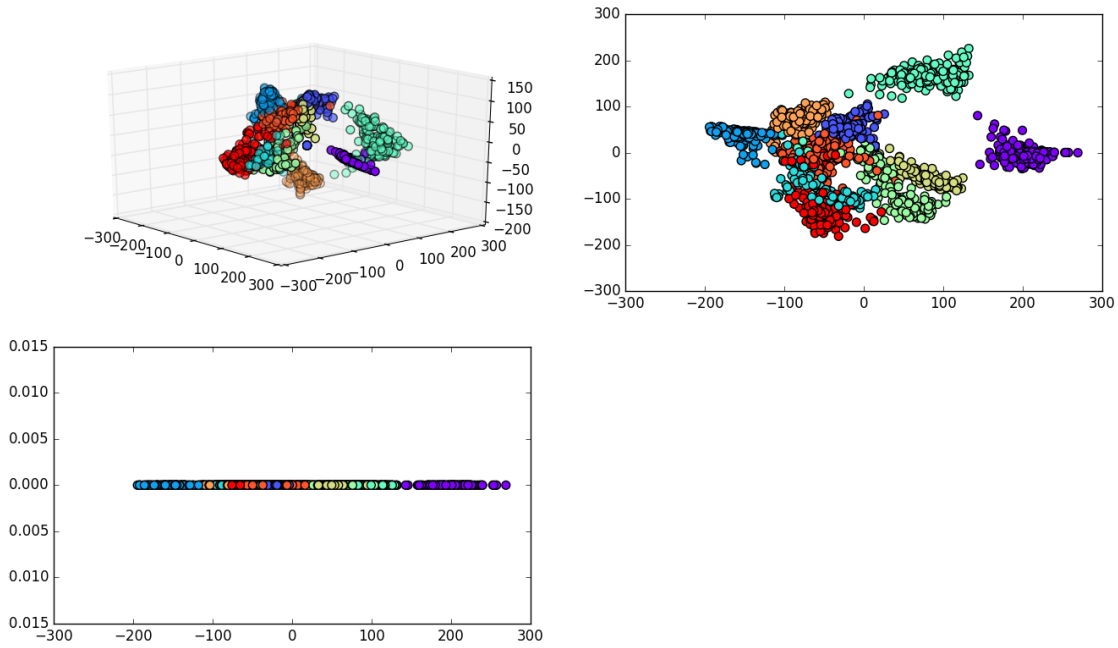


Figure 5.8: The reductions of Digits to 3, 2 and 1 dimensions, respectively, with the ISOMAP algorithm.

Digits reductions using ISOMAP, as illustrated in figure 5.8, have proven to be much more organized, when compared to the reductions made by the PCA algorithm. Indeed, the representation in the 2D shows much less mixed samples, which translates into better scores:

	Original	$\mathbb{R}^{10}$	$\mathbb{R}^3$	$\mathbb{R}^2$	$\mathbb{R}$
1-NN	.99	.99	.97	.92	.46
Score					
Best parameters					
GridSearch time	7.68 s	5.1 s	4.59 s	4.47 s	5.44 s
Reduction time	-	47.93 s	47.99 s	48.31 s	48.82 s
Kruskal's stress	-	.16	.42	.54	.71
Data size	898.5 KB	140.4 KB	42.1 KB	28.08 KB	14 KB

Table 5.5: Description of predictions and reduction performance for Digits and ISOMAP algorithm.

5.4 The Swiss Roll Data Set

The Swiss Roll data set, with 1000 samples and 3 features.

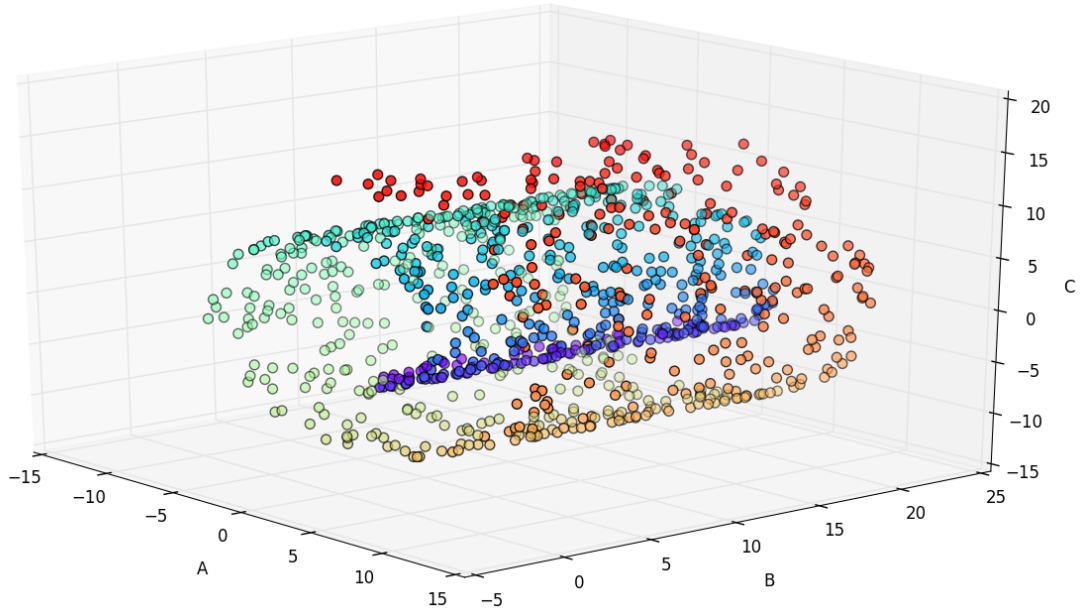


Figure 5.9: The Swiss Roll data set.

Reducing the Swiss Roll Data Set With the PCA Algorithm

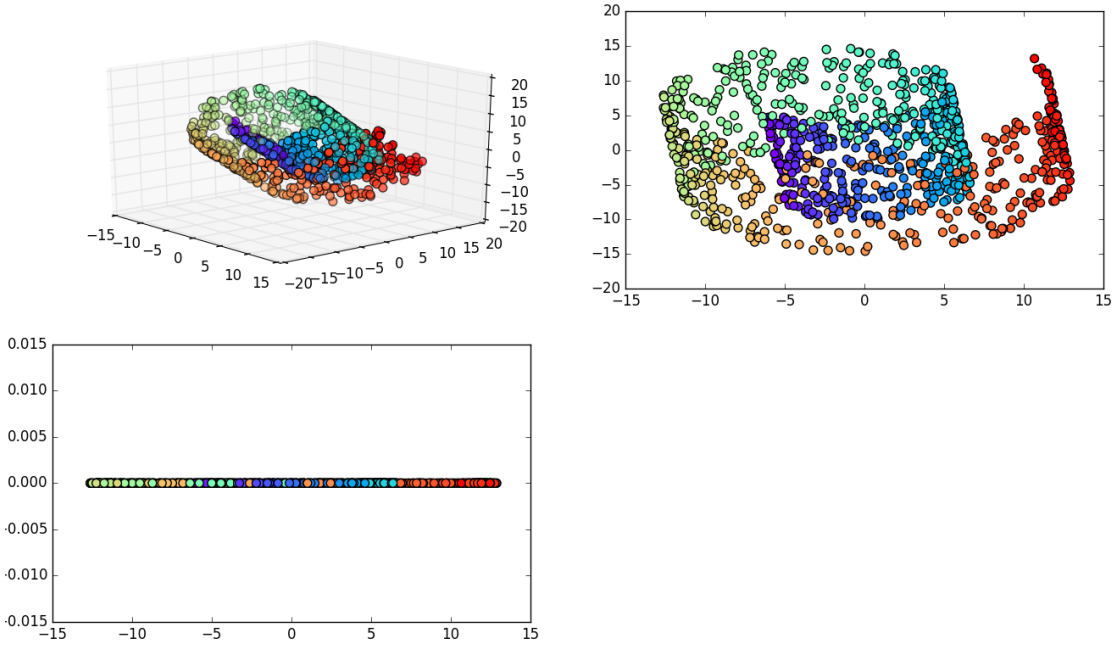


Figure 5.10: The reductions of the Swiss Roll to 3, 2 and 1 dimensions, respectively, with the PCA algorithm.

Figure 5.10 illustrates how the PCA algorithm is not suitable for reducing the Swiss-roll, given its non-linear distribution. Reductions to 2 and 1 dimensions have made very dissimilar samples to become mixed, damaging the original structure of the data set and, of course, confusing learners and reducing their score in the learning process:

	Original	$\mathbb{R}^3$	$\mathbb{R}^2$	$\mathbb{R}$
1-NN	1	1	.26	.2
Score	1	1	.68	.54
Best parameters	k: rbf, C: 100, g: 0.01	C: 100, k: rbf, g: 0.01	C: 100, k: rbf, g: 0.1	C: 1, k: rbf, g: 0.1
GridSearch time	12.72 s	15.13 s	9.07 s	7.11 s
Reduction time	-	2.15 s	2.02 s	2.08 s
Kruskal's stress	-	0	.28	.53
Data size	23.44 KB	23.44 KB	15.63 KB	7.81 KB

Table 5.6: Description of predictions and reduction performance for the Swiss Roll and PCA algorithm.

Notice that the learners trained during the grid search can achieve high scores in the original space $\mathbb{R}^3$. When the same search is applied to the data set reductions, learners only manage to retrieve significantly lower scores. Clearly, reducing non-linear data sets with linear methods has a highly negative impact on the learning process.

Reducing the Swiss Roll Data Set With the ISOMAP Algorithm

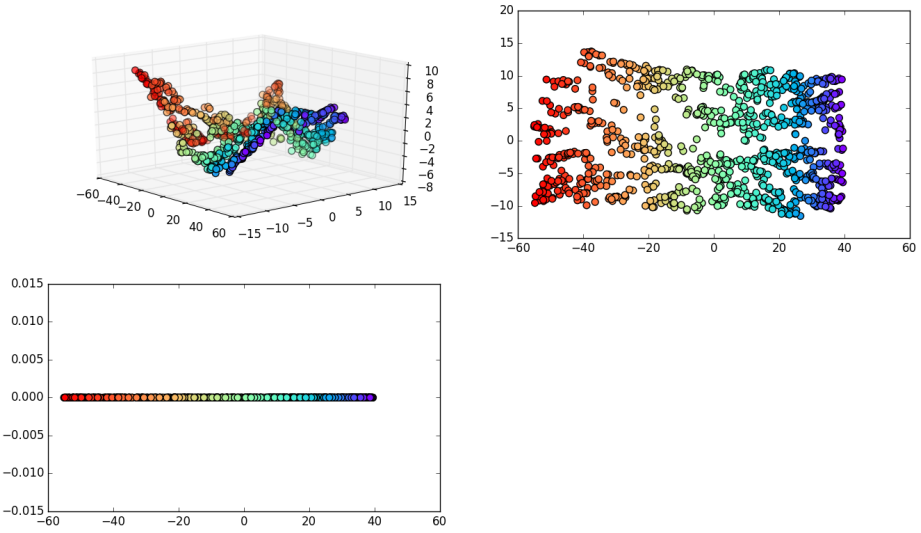


Figure 5.11: The reductions of the Swiss Roll to 3, 2 and 1 dimensions, respectively, with the ISOMAP algorithm.

Figure 5.11 illustrates the efficacy of the ISOMAP algorithm in reducing the Swiss Roll while still preserving the original structure of the data set (similar samples were maintained close to each other). Of course, this has a positive impact on learning:

	Original	$\mathbb{R}^3$	$\mathbb{R}^2$	$\mathbb{R}$
1-NN	1	1	1	1
Score	1	1	1	1
Best parameters	k: rbf, C: 100, g: 0.01	C: 10, k: rbf, g: 0.01	C: 100, k: rbf, g: 0.01	C: 10, k: rbf, g: 0.01
GridSearch time	2.41 s	2.21 s	2.13 s	2.35 s
Reduction time	-	16.32 s	16.75 s	14.72 s
Kruskal's stress	-	.49	.37	.46
Data size	23.44 KB	23.44 KB	15.63 KB	7.81 KB

Table 5.7: Description of predictions and reduction performance for the Swiss Roll and ISOMAP algorithm.

Differently from the reductions with the PCA algorithm, learners trained over reductions retrieved from ISOMAP maintained a perfect score throughout the entire experiment, confirming that ISOMAP is a more adequate method for nonlinear data set reduction.

5.5 The Glass Data Set

Motivated by criminological investigations, the classification of different types of glasses based on the material composition can be demonstrated by the Glass data set, which contains 214 samples and 10 features, where each sample belongs to a specific class (e.g., building windows glass, tableware and headlamps). Lastly, it must be remarked that the first feature is the identification number of a given sample, and was therefore removed during our experiment.

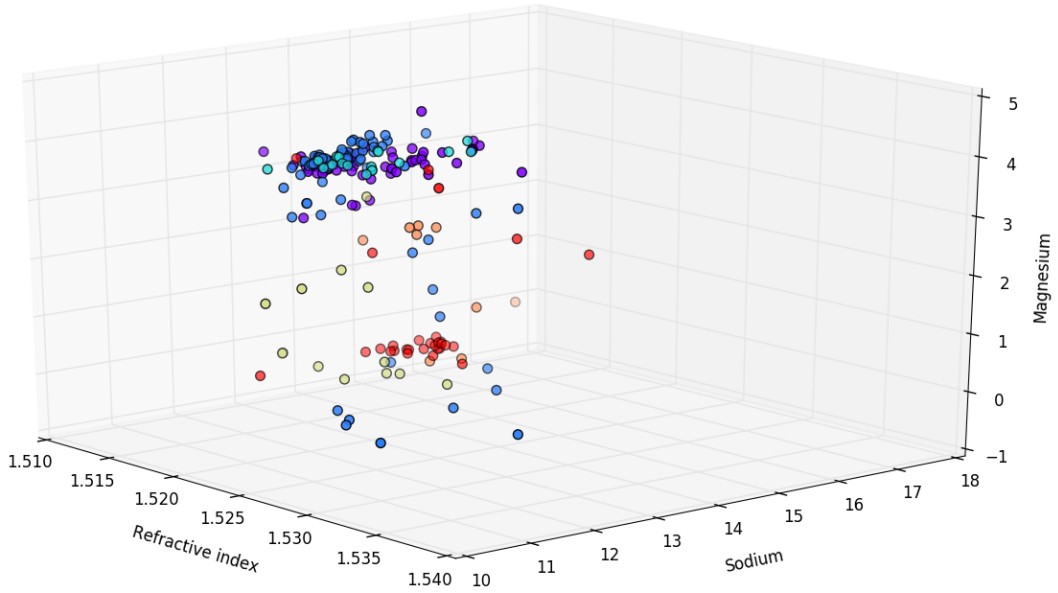


Figure 5.12: The Glass data set.

Reducing the Glass Data Set With the ISOMAP Algorithm

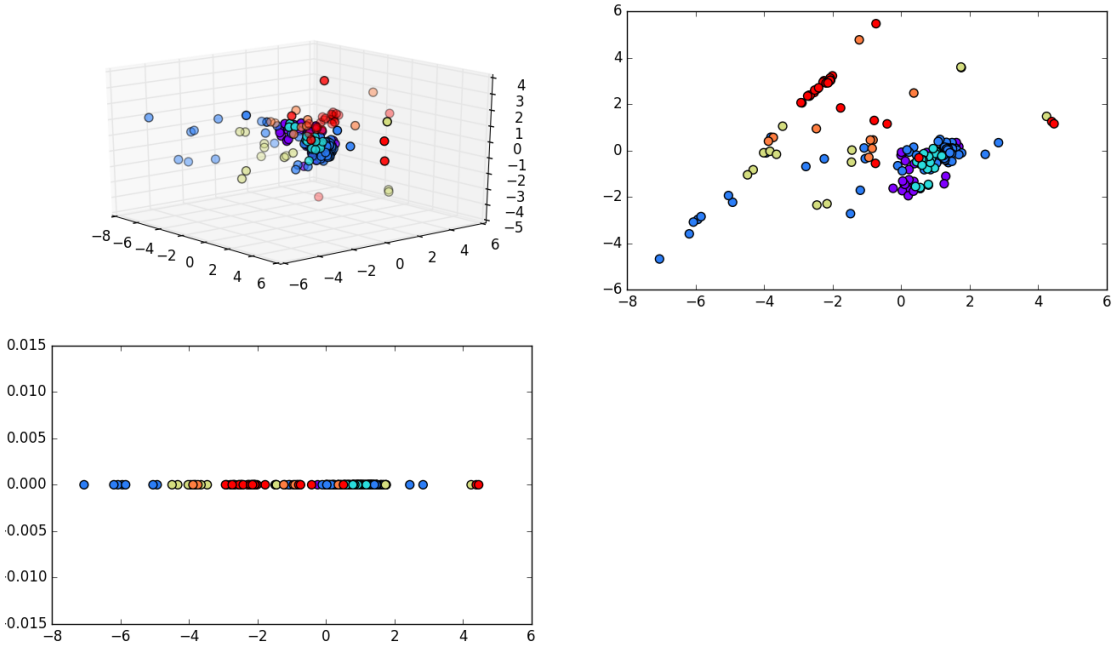


Figure 5.13: The reductions of Glass to 3, 2 and 1 dimensions, respectively, with the ISOMAP algorithm.

We can observe from the representation of Glass in the $\mathbb{R}^3$ (5.13) that samples from different classes present little separation from each other. Indeed, learners will have more difficulty when processing such reductions.

	Original	$\mathbb{R}^8$	$\mathbb{R}^6$	$\mathbb{R}^4$	$\mathbb{R}^3$
1-NN	.63	.63	.6	.63	.65
Score	.64	.65	.63	.65	.65
Best parameters	k: rbf, g: 0.001, C: 1000	k: rbf, g: 0.01, C: 100	k: rbf, g: 0.1, C: 100	k: rbf, g: 0.1, C: 10	k: rbf, g: 0.1, C: 10
GridSearch time	1.96 s	2.85 s	2.89 s	3.21 s	3.40 s
Reduction time	-	2.8 s	2.96 s	2.87 s	2.90 s
Kruskal's stress	-	.2	.16	.15	.21
Data size	15.05 KB	13.38 KB	10.03 KB	6.69 KB	5.02 KB

Table 5.8: Description of predictions and reduction performance for Glass and ISOMAP algorithm.

5.6 The Dermatology Data Set

The Dermatology data set, composed by 366 samples and 35 features. Here, the target is to correctly diagnose the erythemato-squamous diseases.

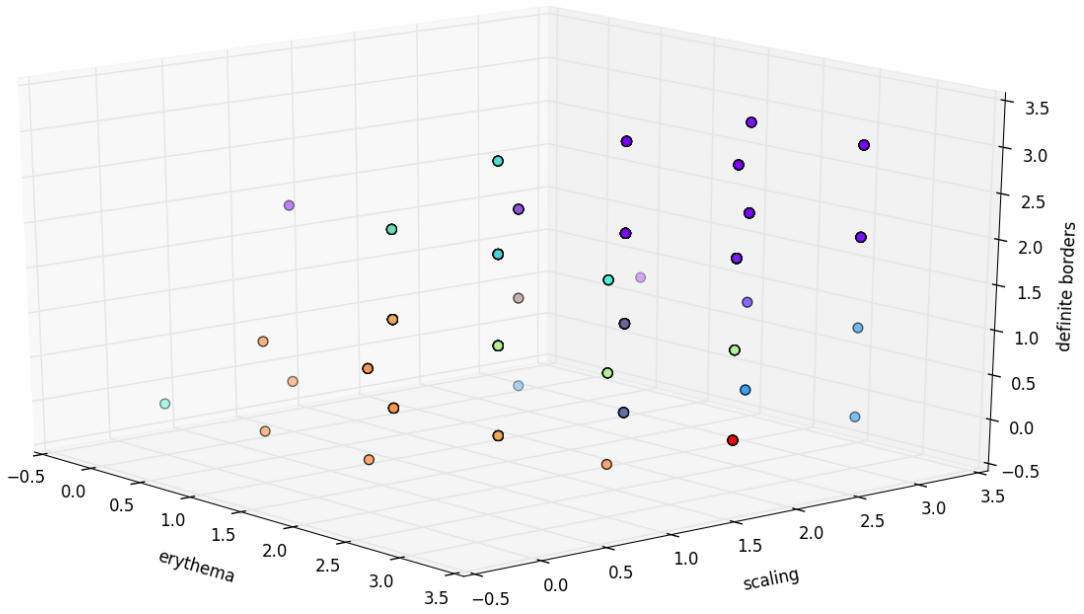


Figure 5.14: The Dermatology data set.

Reducing the Dermatology Data Set With the ISOMAP Algorithm

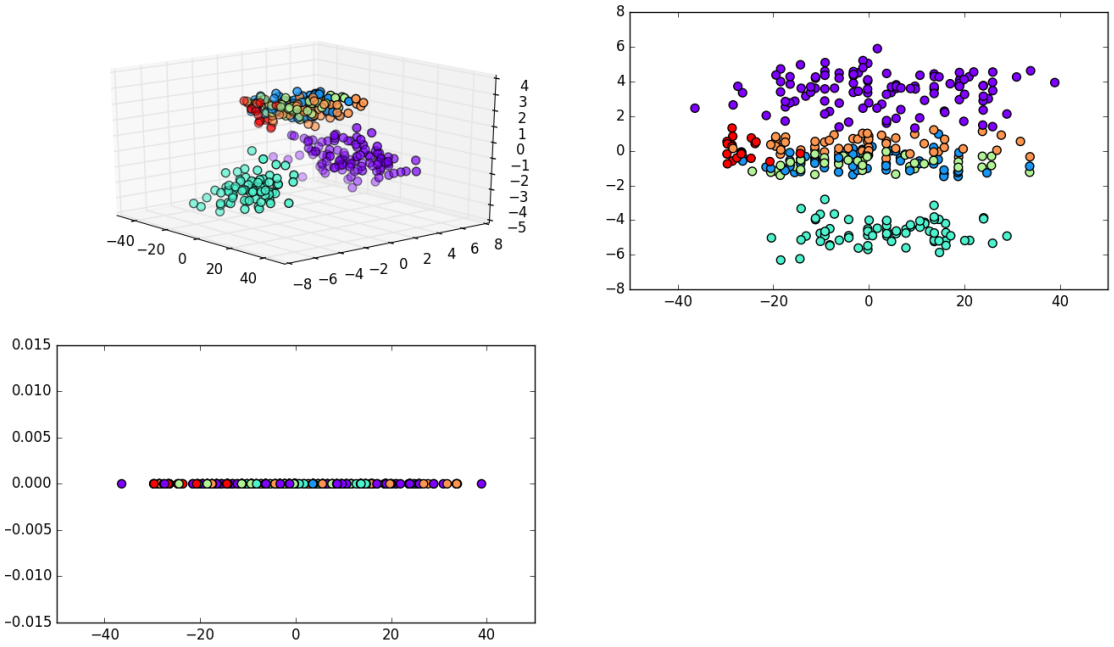


Figure 5.15: The reductions of Dermatology to 3, 2 and 1 dimensions, respectively, with the ISOMAP algorithm.

From figure 5.15, we easily infer that two classes are trivially separable. The rest, however, is highly convoluted in the origin of the space, hence higher dimensions might be necessary to achieve high accuracy scores.

	Original	$\mathbb{R}^{20}$	$\mathbb{R}^{10}$	$\mathbb{R}^{3}$	$\mathbb{R}^{2}$
1-NN	.91	.89	.91	.72	.74
Score	.95	.95	.96	.8	.78
Best parameters	k: linear, C: 10	k: linear, C: 1	k: linear, C: 1	k: linear, C: 100	k: linear, C: 1
GridSearch time	2.76 s	2.61 s	2.57 s	14.47 s	25.66 s
Reduction time	-	66.04 s	65.89 s	65.85 s	66.18 s
Kruskal's stress	-	.02	.03	.07	.1
Data size	97.22 KB	57.19 KB	28.59 KB	8.58 KB	5.72 KB

Table 5.9: Description of predictions and reduction performance for Glass and ISOMAP algorithm.

5.7 The Leukemia Data Set

The Leukemia data set contains 72 samples and 7129 features, which express levels of the genes in a given patient. Each sample belongs to a class $t \in \{-1, 1\}$, tagging which of two variants of leukemia is present in the sample (AML, 25 samples, or ALL, 47 samples) [50].

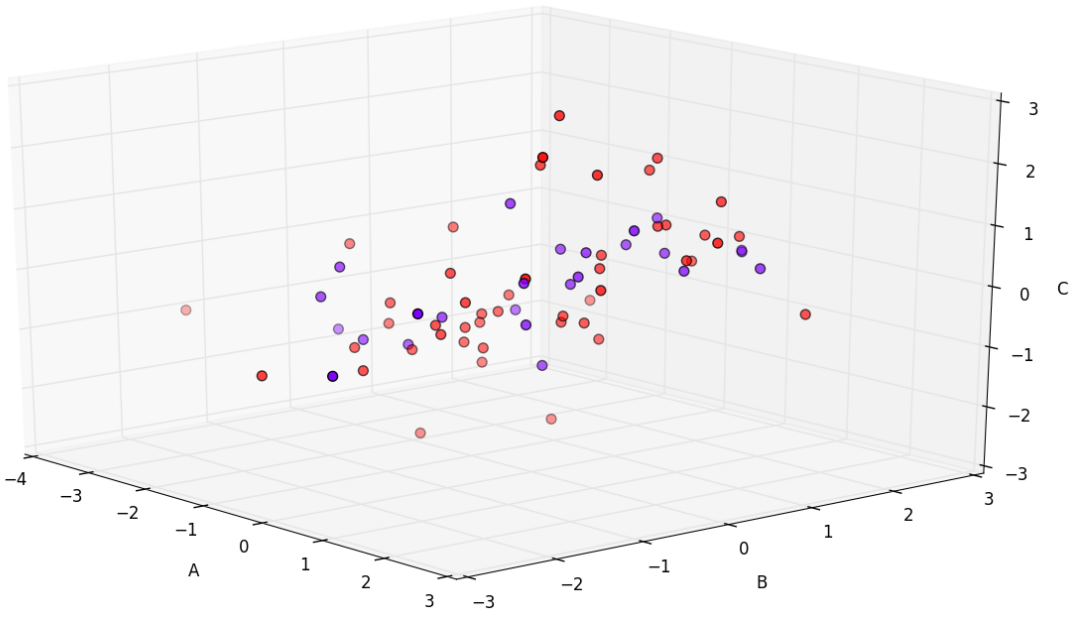


Figure 5.16: The Leukemia data set.

Reducing the Leukemia Data Set With the ISOMAP Algorithm

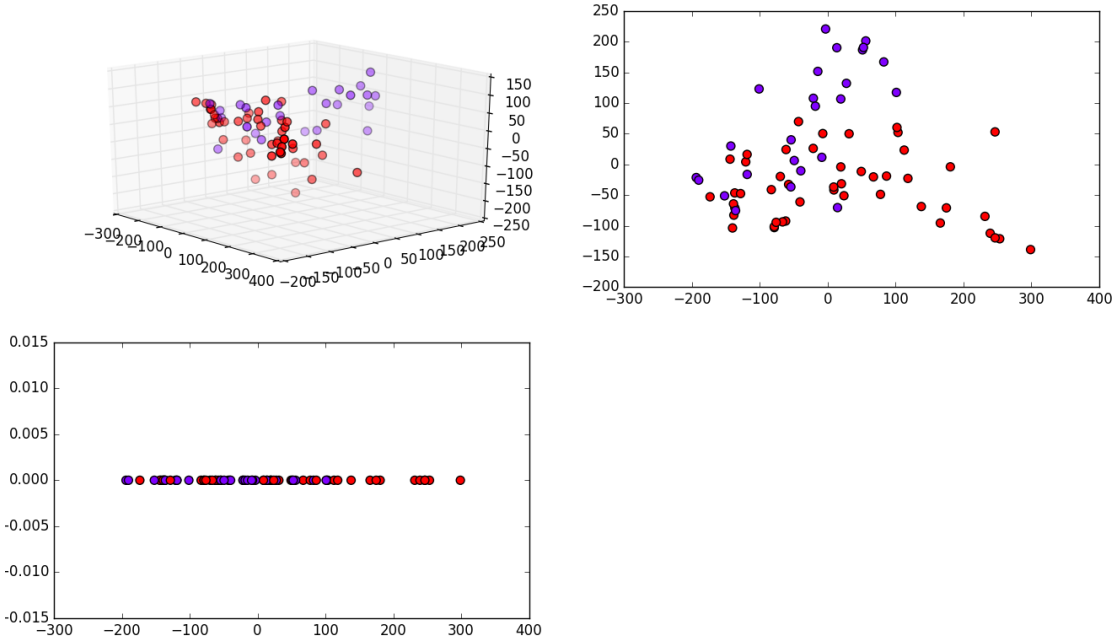


Figure 5.17: The reductions of Leukemia to 3, 2 and 1 dimensions, respectively, with the ISOMAP algorithm.

The Results in this experiment are quite impressive: although visualization has barely changed from 7129 to 3 dimensions, a 7129-dimensional space was reduced to a 10-dimensional one with only 10% of accuracy loss.

5.7.1 The WDBC Data Set

The Wisconsin Diagnostic Breast Cancer (WDBC) data set, containing 596 samples and 32 features computed from a breast mass. WDBC has 357 benign samples and 212 malignant. During this experiment, the first feature was disregarded, as it represents the identification numbers of the samples.

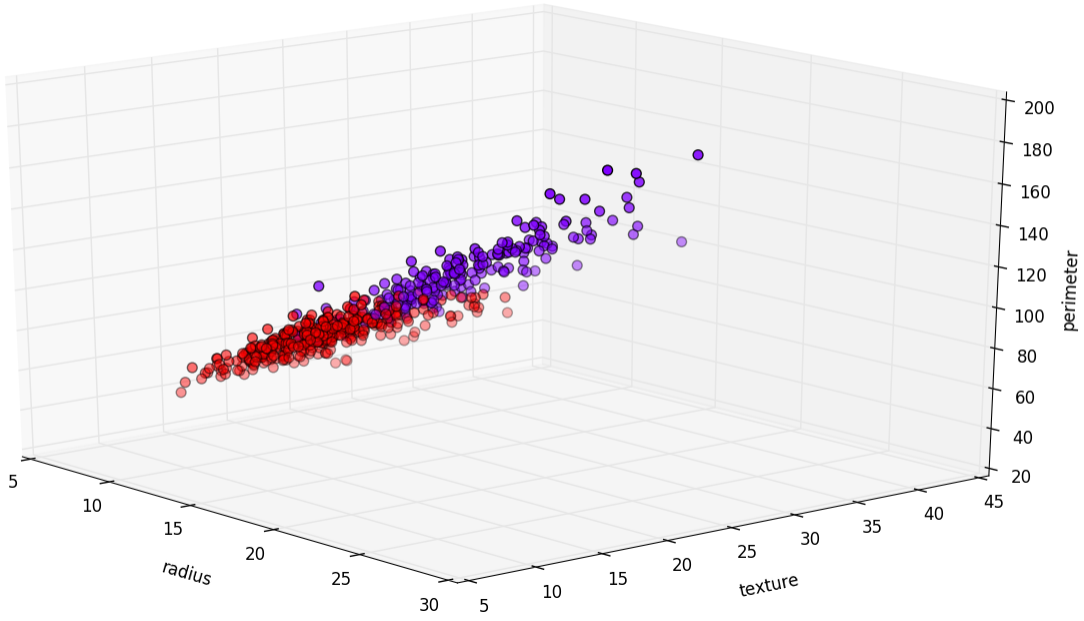


Figure 5.18: The WDBC data set.

Reducing the WDBC Data Set With the ISOMAP Algorithm

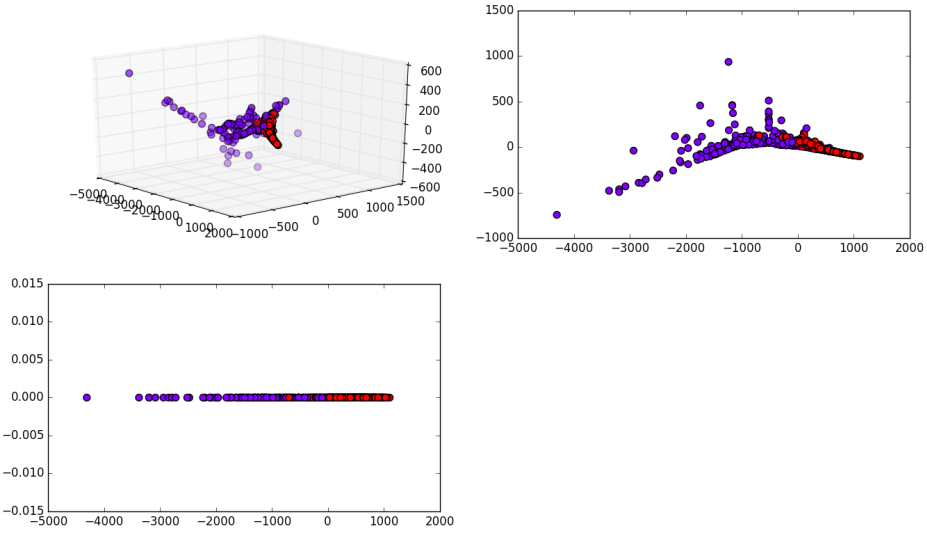


Figure 5.19: The reductions of WDBC to 3, 2 and 1 dimensions, respectively, with the ISOMAP algorithm.

Here, prediction accuracy consistently increased when the data set was reduced.

5.7.2 Diabetes Data Set

With 9 features observed from 768 samples collected from residents of Arizona, USA, the Diabetes data set seeks to diagnose whether a patient (sample) shows signs of diabetes (class 1) according to World Health Organization criteria or not (class 0). Is is also important to mention that only 268 samples belong, in fact, to class 1.

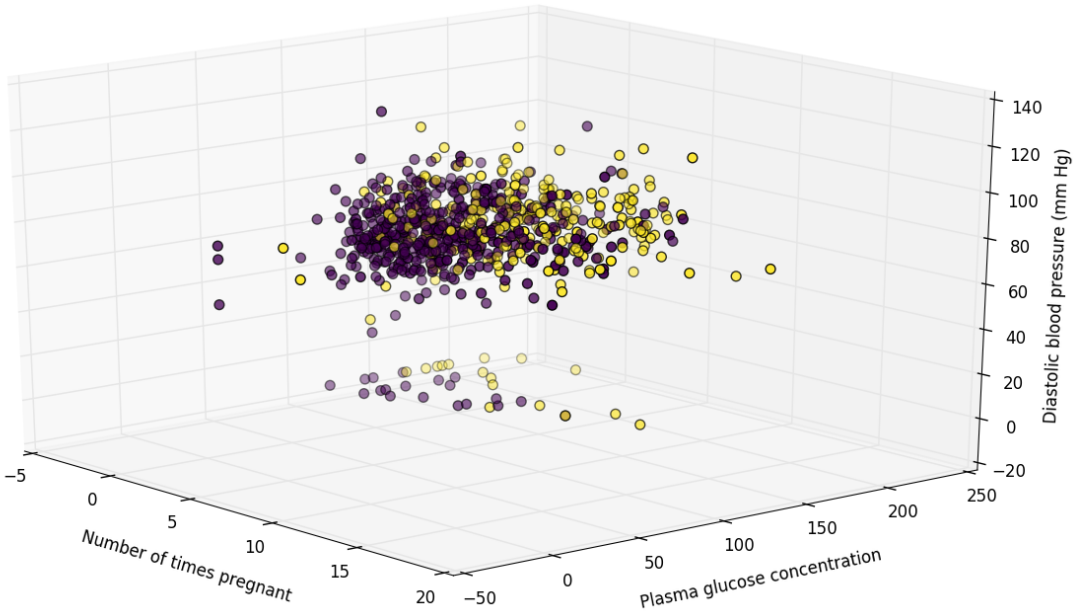


Figure 5.20: The Diabetes data set.

Reducing the Diabetes Data Set With the ISOMAP Algorithm

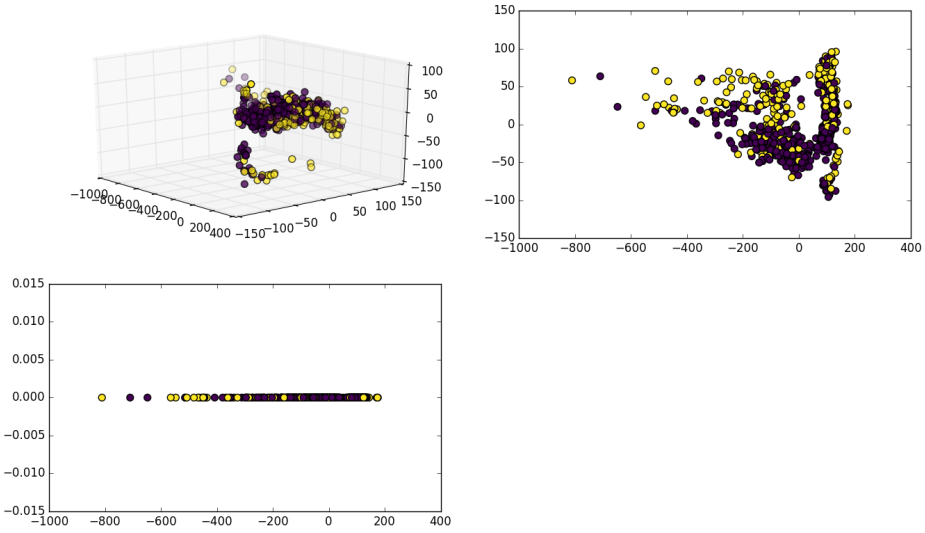


Figure 5.21: The reductions of Diabetes to 3, 2 and 1 dimensions, respectively, with the ISOMAP algorithm.

In this experiment, Diabetes was compressed to one forth of its original size (48 KB to 12 KB), and no significant prediction accuracy was lost.

Final Considerations

In this work, the subject of dimensionality reduction was approached. First, focusing on linear methods and their capacity of extracting features that maximize variance in a data set, we successfully reduced, visualized and learned from well known data sets. Furthermore, we demonstrated how these methods would fail to reduce data sets that followed a nonlinear distribution.

In order to reduce nonlinear data sets, we introduced the ISOMAP algorithm, which takes advantage of properties commonly present in manifolds (e.g., linear locality, neighborhood) to map the data set of interest to an intermediate representation that only preserves dissimilarities of restrictive neighborhoods, "unfolding" the data set before applying a linear reduction method. We then proceeded to formally define ISOMAP's implementation, to analyze its complexity and present some of its limitations, variations and applications. Finally, we demonstrated that ISOMAP can successfully reduce data sets that roughly lie on nonlinear manifolds, but it also strongly dependents on many conditions, such as the manifold assumption, manifold convexity and controlled data noise, severely affecting the number of problems to which it might be applied.

In conclusion, we have observed that although its limitations, ISOMAP represented an advance in manifold learning, being a highly regarded method in dimensionality reduction until today. In practice, its importance is clearly observed when considering the great number of machine learning libraries, languages and computational environments which implement it, as well as the great load of study done by many authors trying to apply, improve or extend it.

Bibliography

[1] E. W. Weisstein, "Stereographic projection." [Online]. Available: http://mathworld.wolfram.com/StereographicProjection.html.

[2] scikit learn, "Svm: Maximum margin separating hyperplane." [Online]. Available: http://scikit-learn.org/stable/auto_examples/svm/plot_separating_hyperplane.html.

[3] P. R. C. D. Manning and H. Schutze, *Introduction to information retrieval*, vol. 1. Cambridge university press Cambridge, 2008.

[4] B. Rohrer, "How to choose algorithms for microsoft azure machine learning." [Online]. Available: https://azure.microsoft.com/en-us/documentation/articles/machine-learning-algorithm, 2015.

[5] M. Lichman, "UCI machine learning repository," 2013.

[6] J. Brownlee, "Practical machine learning problems." [Online]. Available: http://machinelearningmastery.com/practical-machine-learning-problems/, 2013.

[7] P. Baldi and S. Brunak, *Bioinformatics: The Machine Learning Approach*. A Bradford book, A Bradford Book, 2001.

[8] J. Wang, *Geometric Structure of High-Dimensional Data and Dimensionality Reduction*. Springer Berlin Heidelberg, 2012.

[9] S. S. A. H. Renear and K. M. Wickett, "Definitions of dataset in the scientific and technical literature," *ASIS&T 2010*, 2010.

[10] L. Cayton, "Algorithms for manifold learning," *Univ. of California at San Diego Tech. Rep*, pp. 1–17, 2005.

[11] A. Ghodsi, "Dimensionality reduction a short tutorial," *Department of Statistics and Actuarial Science, Univ. of Waterloo, Ontario, Canada*, 2006.

[12] S. Ross, *Introductory Statistics*. Elsevier Science, 2010.

[13] T. Cox and M. A. A. Cox, *Multidimensional scaling*. Boca Raton, FL, USA: CRC Press, 2000.

[14] W. Gander, *The Singular Value Decomposition*, December 2008.

[15] J. M. Lee, *Manifolds and differential geometry*, vol. 107. American Mathematical Society Providence, 2009.

[16] J. Lee, *Introduction to Smooth Manifolds*. Graduate Texts in Mathematics, New York, NY, USA: Springer, 2002.

[17] S. Burris and H. Sankappanavar, *A Course in Universal Algebra*. Graduate Texts in Mathematics, Springer New York, 2011.

[18] C. Berge and E. Minieka, *Graphs and hypergraphs*, vol. 7. North-Holland publishing company Amsterdam, 1973.

[19] W. Mayeda, *Graph Theory*. John Wiley & Sons, 1972.

[20] T. Cormen, *Introduction to Algorithms*. Introduction to Algorithms, MIT Press, 2001.

[21] M. J.Golin, "Lecture 15: The floyd-warshall algorithm." [Online]. Available: `http://www.cse.ust.hk/faculty/golin/COMP271Sp03/Notes/MyL15.pdf`, 2003.

[22] P. Langley, *Elements of Machine Learning*. Machine Learning Series, Morgan Kaufmann, 1996.

[23] L. W. Hosch, "Machine learning." [Online]. Available: `http://global.britannica.com/technology/machine-learning`.

[24] M. Awad and R. Khanna, *Efficient Learning Machines: Theories, Concepts, and Applications for Engineers and System Designers*. New York, NY, USA: Apress, 2015.

[25] X. Zhu and A. B. Goldberg, "Introduction to semi-supervised learning," *Synthesis lectures on artificial intelligence and machine learning*, vol. 3, no. 1, pp. 1–130, 2009.

[26] O. Kramer, *Dimensionality Reduction with Unsupervised Nearest Neighbors*. Intelligent Systems Reference Library, Springer Berlin Heidelberg, 2013.

[27] C. Elkan, "Nearest neighbor classification," *elkan@ cs. ucsd. edu‖,‖ January*, vol. 11, p. 3, 2011.

[28] J. Weston, "Support vector machine (and statistical learning theory) tutorial." [Online]. Available: `http://www.cs.columbia.edu/~kathy/cs4701/documents/jason_svm_tutorial.pdf`.

[29] P. Winston, "Learning: Support vector machine." [Online]. Available: `https://www.youtube.com/watch?v=_PwhiWxHK8o`, 2010.

[30] R. Rifkin, "Multiclass classification." [Online]. Available: `http://www.mit.edu/~9.520/spring09/Classes/multiclass.pdf`, 2008.

[31] "Svm multi-class classification." [Online]. Available: `https://www.sec.in.tum.de/assets/lehre/ws0910/ml/slideslecture9.pdf`, 2008.

[32] scikit learn, "Cross-validation: evaluating estimator performance." [Online]. Available: `http://scikit-learn.org/stable/modules/cross_validation.html`.

[33] scikit learn, "Grid search: Searching for estimator parameterse." [Online]. Available: `http://scikit-learn.org/stable/modules/grid_search.html`.

[34] J. Shlens, "A tutorial on principal component analysis," *arXiv preprint arXiv:1404.1100*, 2014.

[35] G. Dunteman, *Principal components analysis*. No. 69, Newbury Road, CA, USA: Sage, 1989.

[36] L. I. Smith, "A tutorial on principal components analysis," *Cornell University, USA*, vol. 51, p. 52, 2002.

[37] S. Raschka, "Implementing a principal component analysis (pca) in python step by step." [Online]. Available: `http://sebastianraschka.com/Articles/2014_pca_step_by_step.html`, April 2014.

[38] T. Naes and E. Risvik, *Multivariate Analysis of Data in Sensory Science*. Data Handling in Science and Technology, Elsevier Science, 1996.

[39] V. S. T. B. Joshua and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

[40] V. D. Silva and J. B. Tenenbaum, "Global versus local methods in nonlinear dimensionality reduction," in *Advances in neural information processing systems*, pp. 705–712, 2002.

[41] L. Shi and J. Gu, "A fast manifold learning algorithm," *Information Technology Journal*, vol. 11, no. 3, pp. 380–383, 2012.

[42] S. M. J. Ham, D. D. Lee and B. Schölkopf, "A kernel view of the dimensionality reduction of manifolds," in *Proceedings of the twenty-first international conference on Machine learning*, p. 47, ACM, 2004.

[43] M. H. C. Law and A. K. Jain, "Incremental nonlinear dimensionality reduction by manifold learning," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 28, no. 3, pp. 377–391, 2006.

[44] H. L. J. Choo, C. K. Reddy and H. Park, "p-isomap: An efficient parametric update for isomap for visual analytics.," in *SDM*, pp. 502–513, SIAM, 2010.

[45] J. L. Y. Chen and J. Wang, *Machine Learning and Statistical Modeling Approaches to Image Retrieval*. The Information Retrieval Series, Springer US, 2004.

[46] E. P. L. Maaten and J. Herik, "Dimensionality reduction: A comparative review," *Journal of Machine Learning Research*, vol. 10, no. 1-41, pp. 66–71, 2009.

[47] T. Lin and H. Zha, "Riemannian manifold learning," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 30, no. 5, pp. 796–809, 2008.

[48] D. Donoho and C. Grimes, *When Does ISOMAP Recover the Natural Parameterization of Families of Articulated Images?* Technical report (Stanford University. Dept. of Statistics), Department of Statistics, Stanford University, 2002.

[49] G. Lerman, "Manifold learning techniques: so which is the best?." [Online]. Available: `http://www.math.ucla.edu/~wittman/mani/mani_presentation.pdf`, 2005.

[50] F. Ducatelle, "Datasets for data mining." [Online]. Available: `http://www.inf.ed.ac.uk/teaching/courses/dme/html/datasets0405.html`.