Federal University of São Carlos — UFSCar
Center of Exact Sciences and Technology — CCET
Computing Department — DC
Graduate Program in Computer Science — PPGCC

Alaor Cervati Neto

Dimensionality reduction-based metric learning using information theoretic measures

São Carlos
2024

Alaor Cervati Neto

Dimensionality reduction-based metric learning using information theoretic measures

Ph.D. Thesis presented to the Graduate Program in Computer Science at the Center of Exact Sciences and Technology of Federal University of São Carlos, as part of the requirements for obtaining the title of Doctor in Computer Science.

Concentration Area: Computer Vision
Advisor: Prof. Dr. Alexandre Luis Magalhães Levada

São Carlos
2024

Cervati Neto, Alaor

Dimensionality reduction-based metric learning using information theoretic measures / Alaor Cervati Neto -- 2024.
61f.

Tese de Doutorado - Universidade Federal de São Carlos, campus São Carlos, São Carlos
Orientador (a): Alexandre Luis Magalhães Levada
Banca Examinadora: Alexandre Luis Magalhães Levada, Cesar Henrique Comin, André Ricardo Backes, Agma Juci Machado Traina, Fabricio Aparecido Breve
Bibliografia

1. Redução de dimensionalidade. 2. Aprendizado de métricas. 3. Teoria da informação. I. Cervati Neto, Alaor. II. Título.

Ficha catalográfica desenvolvida pela Secretaria Geral de Informática (SIn)

DADOS FORNECIDOS PELO AUTOR

Bibliotecário responsável: Ronildo Santos Prado - CRB/8 7325

UNIVERSIDADE FEDERAL DE SÃO CARLOS

Centro de Ciências Exatas e de Tecnologia
Programa de Pós-Graduação em Ciência da Computação

Folha de Aprovação

Defesa de Tese de Doutorado do candidato Alaor Cervati Neto, realizada em 30/04/2024.

Comissão Julgadora:

Prof. Dr. Alexandre Luis Magalhães Levada (UFSCar)

Prof. Dr. Cesar Henrique Comin (UFSCar)

Prof. Dr. André Ricardo Backes (UFSCar)

Profa. Dra. Agma Juci Machado Traina (USP)

Prof. Dr. Fabricio Aparecido Breve (UNESP)

O Relatório de Defesa assinado pelos membros da Comissão Julgadora encontra-se arquivado junto ao Programa de Pós-Graduação em Ciência da Computação.

Acknowledgements

To my advisor, Prof Alexandre, and his endless patience for my innumerable delays and miscommunications. None of this would be possible without his constant guidance.

To my family and friends, for providing all the support I never had to ask for.

To my teachers and colleagues, always helpful and available whenever needed.

To my students, who taught me more than I did them. For the best way to assess knowledge about a subject is to try and teach it, and I certainly tried (results varied).

To whoever is reading, for the interest in this little contribution to our knowledge. May it not be so small as to go unrecognised.

Resumo

O processamento de grandes quantidades de dados para extrair informações úteis é um dos principais problemas que podem ser abordados com o aprendizado de máquina. Uma das formas de obter essas informações é agrupando os dados de acordo com as características que tenham em comum. Em conjuntos de dados muito complexos, esta tarefa pode ser realizada encontrando formas mais simples de representar as relações entre esses dados, diminuindo o número de suas dimensões. Há vários métodos para descobrir os agrupamentos de dados em um conjunto de forma automática. Por outro lado, descobrir um meio de minimizar a complexidade desses dados sem perda de conteúdo relevante é um processo computacionalmente custoso. Uma alternativa para isto é o tratamento destes conjuntos de dados como distribuições de probabilidades de variáveis aleatórias e a utilização de conceitos e medidas de teoria de informação para descobrir suas relações de modo mais eficiente. Este trabalho descreve alguns métodos de redução de dimensionalidade e medidas de teoria de informação e propõe que ambos sejam unidos para a obtenção de resultados melhores, criando variantes mais resistentes a perturbações nos dados ou diferenças no tamanho dos conjuntos. A adaptação dos métodos existentes para incluir medidas baseadas em teoria da informação é testada em conjuntos de dados reais, e os resultados verificados formalmente quanto a sua adequação para a obtenção de métricas mais precisas. Na maior parte dos casos estudados, os resultados demonstraram um desempenho superior ao dos métodos tradicionais para classificação, enquanto em outros as alterações realizadas tornaram-nos mais eficazes para conjuntos de dados com um número reduzido de amostras.

Palavras-chave: Redução de dimensionalidade. Aprendizado de métricas. Teoria da informação.

Abstract

Processing large amounts of data to extract useful information is one of the main issues that may be approached using machine learning. One way to obtain this information is by grouping data according to their common features. In very complex data sets, this task may be accomplished by finding simpler ways of representing the relations between this data, lowering their dimensionality. There are many methods to find the data groups in a set automatically. However, finding a way to minimise the complexity of this data without losing relevant content is a computationally costly process. An alternative to that is treating these data sets as probability distributions of random variables and using concepts and measures from information theory to find their relations more efficiently. This work describes some dimensionality reduction methods and information theory measures and proposes that they be joined in order to obtain better results, by creating variants more resistant to disruption in data or differences in set sizes. The adaptation of existing methods to include information theory-based measures is tested on real datasets, and results formally verified as to their adequacy for obtaining more accurate metrics. In most of the tested cases, results show a better performance compared to traditional classification methods, while in others the modifications made those more effective for datasets with fewer samples.

Keywords: Dimensionality Reduction. Metric Learning. Information Theory.

List of Figures

List of Tables

List of Algorithms

List of Acronyms

CAN Clustering with Adaptive Neighbours

DR Dimensionality Reduction

DT Decision Trees

ES-ISOMAP Enhanced Supervised Isometric Feature Mapping

GPC Gaussian Process Classifier

iid independent and identically distributed

IMSE integrated mean square error

ISOMAP Isometric Feature Mapping

ISOMAP-KL Entropic Isometric Feature Mapping

KDE Kernel Density Estimation

KDE-ISOMAP Kernel Density Estimation-based Isometric Feature Mapping

K-ISO-F Kernel Density Estimation-based Isometric Feature Mapping with fixed bandwidth $h = 0.1$ for all probability density functions

K-ISO-SC Kernel Density Estimation-based Isometric Feature Mapping with Scott's rule for bandwidth estimation

K-ISO-SIL Kernel Density Estimation-based Isometric Feature Mapping with Silverman's rule for bandwidth estimation

KL-divergence Kullback-Leibler Divergence

KNN k Nearest Neighbours

KPCA Kernel Principal Component Analysis

LDA Linear Discriminant Analysis

LE Laplacian Eigenmaps

LLE Locally Linear Embedding

LPP Locality Preserving Projections

MDS Multidimensional Scaling

ML Manifold Learning

MLP Multilayer Perceptron

NB Naive Bayes

PCA Principal Component Analysis

pdf probability density function

PNN Probabilistic Nearest Neighbours

PNN-LPP Probabilistic Nearest Neighbours-Based Locality Preserving Projections

QDA Quadratic Discriminant Analysis

RFC Random Forest Classifier

SC Silhouette Coefficient

SEt-SNE Supervised Entropic t-Distributed Stochastic Neighbour Embedding

SGt-SNE Supervised Geodesic t-Distributed Stochastic Neighbour Embedding

S-LE Supervised Laplacian Eigenmaps

SLLE Supervised Locally Linear Embedding

SNE Stochastic Neighbour Embedding

SVM Support Vector Machines

t-SNE t-Distributed Stochastic Neighbour Embedding

UMAP Uniform Manifold Approximation and Projection

Contents

Chapter 1

Introduction

Many recent machine learning applications require synthesising a classification or function from a very large data set. Modern datasets consist of a large amount of examples, each made up of many features. Although having access to a large amount of examples is beneficial to an algorithm that attempts to generalise something about the data, managing a larger number of features, some of which may be irrelevant or misleading, is usually costly for the algorithm. In order to reduce this cost for machine learning algorithms, many techniques were developed to greatly reduce the number of features in a dataset, that is, the dimensionality of data (CAYTON, 2005).

A common approach for this sort of problem is based on the observation that high dimensionality data are usually much simpler than the number of dimensions suggests. Particularly, a dataset may contain many features which have the same cause and, therefore, much information in common. It is interesting to have a simplified representation that aligns with the parameters that generate such data. This notion is formalised in the concept of *manifold*: the data are in a low dimension set embedded in a high dimension space, where the low dimension space reflects the underlying parameters represented in the high dimension space. The attempt to find this manifold structure in a dataset is called Manifold Learning (ML) (CAYTON, 2005).

ML is deeply connected to unsupervised metric learning as, aside from learning a more compact and meaningful representation of the observed datasets, they learn a distance function which, geometrically, is more adequate for representing a similarity measure between a pair of objects in the collection. As such, by learning a manifold structure, generally one consequently acquires a powerful metric (WANG; SUN, 2015).

1.1 Motivation

Though existing ML and Dimensionality Reduction (DR) methods achieve mostly acceptable results, their formulation generally depends on metrics that, despite being usually sufficient, can be heavily influenced by disruptions in data (FRENAY; VERLEYSEN, 2014). Having a method that is more resistant to disruptions such as noise and outliers can provide better classification than traditional methods and allow for reliable application on more diverse datasets.

1.2 Research Hypothesis

The present work proposes to investigate the possibility of extending existing DR methods in order to obtain more powerful metrics. To do so, the construction of the matrices that define the classification of these methods which, in most cases, use the Euclidean distance to measure dissimilarity between objects in a dataset, are to be modified into ones using information theory-based measures. Using these metrics may provide a more adequate dataset for the application of these methods than the starting ones, since they are based on the distribution of the probabilities of each datum occurring. Thus, the algorithms are changed to provide possibly more relevant results, considering the relationship between each data class and its impact over the full set.

1.3 Objectives

The main way this hypothesis is to be tested is statistical. After an initial reformulation of existing algorithms to work with more contextually relevant distance measures, rather than the original, often overly sensitive and inflexible, ones, tests using real datasets aim to assert their effectiveness in actual use cases. While mathematical representations of these modified algorithms lead to intuit their higher accuracy in classification compared to traditional ones, the unpredictable nature of real data, represented as random variables, is a much more reliable test of their potential benefits. Therefore, all proposed methods have been extensively tested and their performance compared to the exiting ones on several datasets to obtain a statistically significant sample that can be confidently treated as sufficient for an informed evaluation.

1.4 Structure of This Work

This work is organised as follows:

- Chapter 1 is this introduction, describing the motivation behind its existence, the research hypothesis, and how it is expected to be achieved.

- Chapter 2 briefly explains the concept of a manifold, the information theory measures, and the methods that are adapted to use them as part of their construction.
- Chapter 3 explores these modifications and formulations of the new methods to be applied to the data.
- Chapter 4 demonstrates the usage of these modified methods on data and discusses the results of such experiments, as well as the tests used to quantify their effectiveness.
- Chapter 5 contextualises the experimental findings, surmises the contributions these methods have provided to this research field, and suggests possible paths for future research.

Chapter 2

Theoretical Foundations

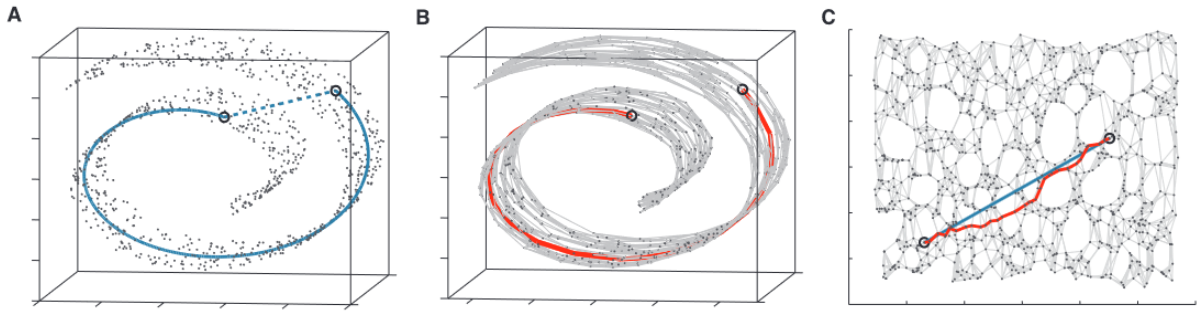
In this chapter, the bases for understanding the motivation behind the methods proposed in this work are presented. From an initial mathematical description of basic space definitions to the currently available algorithms for DR which are to be modified, in search of greater efficiency, are described next. To provide a clearer logical progression, the mathematical derivations leading from the initial ideas to the formulations used in the modified methods are detailed, as well as the tests employed to verify their effectiveness.

2.1 Riemannian Manifolds

A Riemannian manifold, so named for being defined by Riemann (1851), and often called simply *manifold*, is a topological space in which the region around each point resembles an Euclidean space. If a n-dimensional space can be divided into patches (or neighbourhoods) which are locally Euclidean spaces, that is, the distances between points in each patch of which can be adequately calculated using Euclidean measures, it is a *manifold*. This concept can be easily grasped considering the many examples of manifolds found in simple geometric features in $\mathbb{R}^3$, such as surfaces with zero width, representing a 2D object immersed in a 3D space.

For instance, an irregular surface in $\mathbb{R}^2$ can represent a plane from $\mathbb{R}^2$ the values of which in z are irrelevant to its original domain. These values of z, however, may affect the Euclidean distances of this object in $\mathbb{R}^3$ in a way that mischaracterises the relations between its points in the original feature space $\mathbb{R}^2$. Which is to say, in representing an object defined in $\mathbb{R}^2$ in a $\mathbb{R}^3$ space, one adds unnecessary data that only complicates its comprehension (noise), while providing no valuable information (CAYTON, 2005).

Figure 1 – Representation of Euclidean and geodesic distances in the Swiss roll data set.



Source: (TENENBAUM; SILVA; LANGFORD, 2000).

A classical example of one such manifold is known as the "Swiss roll", as shown in Figure 1, in which the Euclidean distances between the points in a high dimension space (Figure 1A) distort their proper representation and topology (Figure 1B) in a low dimension space (Figure 1C). Although this is a rather simple example, the issue it reveals, called the "Curse of Dimensionality", affects other, less easily visualised, high dimension spaces. This issue is caused by excess data that do not contribute to express relevant information about the subject in question.

ML, also known as *non-linear dimension reduction*, is a set of methods to find the low-dimensional structure of data. DR for large, high-dimensional data is not merely a way to reduce the data; the new representations and descriptors obtained by ML reveal the geometric shape of high-dimensional point clouds and allow one to visualise, de-noise, and interpret them (MEILĂ; ZHANG, 2024). The purpose of ML, then, is to find this lower dimension topology from a data set in a high dimension space, discovering the underlying manifold in a set of data points with a larger number of features than necessary to define it. Therefore, the main goal is to reduce the dimensionality of the input space, that is, to synthesise a representation of the data that expresses its relevant features while disregarding or combining those less important to the low dimension manifold (TENENBAUM; SILVA; LANGFORD, 2000).

2.2 Information Theory-based Measures

The amount of information that can be sent over a communication channel can be computed from the properties of both message and channel. Likewise, the amount of information contained in one such message can be known based on the probabilities of each element that makes it up being present. From this set of measures to quantify information transmitted originated the *Mathematical Theory of Communication*, also known as *Information Theory* (SHANNON; WEAVER, 1964).

2.2.1 Shannon Entropy

The main measure in the Mathematical Theory of Communication is *entropy*, that has different interpretations depending on the context in which it is calculated. Its general form is (SHANNON; WEAVER, 1964):

$$H = -\sum_{i}^{n} p_i \log p_i, \tag{1}$$

where p_i is the probability of the i-th element in a set of n occurring. For the case of a message generated from a random variable, it can also be written (COVER; THOMAS, 2006):

$$H(X) \equiv -\sum_{x} p(x) \log p(x), \tag{2}$$

where X is a discrete random variable shaped like a triple (x, A_X, P_X), being x the index of vectors $A_X = \{a_1, a_2, \ldots, a_i\}$, that lists the symbols in which the message is coded, $P_X = \{p_1, p_1, \ldots, p_i\}$, which gives the probability of each symbol appearing. Since these values come from statistics and probability, $p(x = a_i) \geq 0$ and $\sum_{x \in A_X} p(x = a_i) = 1$.

The simple case of a variable with two possible results, which defines a binary system, helps to understand entropy as a measure of the uncertainty over a system. Considering a random variable that can take values 0 and 1, if probability $p_0 = p$ is associated to state 0, the probability of state 1 occurring must be $p_1 = (1 - p)$. Let $x = \{0, 1\}$, $A_X = \{0, 1\}$, and $P_X = \{p, 1 - p\}$, equation (2) can be written as:

$$H(p_0, p_1) = -p_0 \log_2 p_0 - p_1 \log_2 p_1, \tag{3}$$

$$H(p_0, p_1) = H(p) = -p \log_2 p - (1 - p) \log_2 (1 - p). \tag{4}$$

It can be noted that entropy assumes maximum value ($H = 1$) when $p = 0.5$, that is, when both states occur with the same probability, which matches the interpretation of entropy as a measure of uncertainty over the system's results. For maximum entropy, there is the same likelihood of knowing both states and, therefore, the uncertainty over which result is obtained is maximum. Assuming a higher probability for one result, such as $p = 0.8$ for example, entropy lessens, as does uncertainty over the result, as experimental results tend to a value. In an extreme case where $p = 1$ (or $p = 0$) entropy is zero, for both information obtained with the measure as the uncertainty over the result are zero, since the value is known before measuring, being 0 (or 1) with certainty.

Introducing entropy of a random variable x as expected value of self-information:

$$H(p) = -\int p(x) \left[\log p(x)\right] \mathrm{d}x = -E\left[\log p(x)\right], \tag{5}$$

where $p(x)$ is the probability density function (pdf) of x. Assuming x has a normal distribution $N(\mu, \sigma^2)$, pdf $p(x)$ is given by:

$$p(x; \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(x - \mu)^2\right), \tag{6}$$

where μ denotes mean and σ^2 variance of x. Calculating the logarithm of pdf comes:

$$\log p(x) = -\frac{1}{2}\log\left(2\pi\sigma^2\right) - \frac{1}{2\pi\sigma^2}(x-\mu)^2. \tag{7}$$

Replacing equation (7) into (5), leads to:

$$H(p) = \frac{1}{2}\log\left(2\pi\sigma^2\right) + \frac{1}{2\sigma^2}E\left[(x-\mu)^2\right] = \frac{1}{2}\log\left(2\pi\sigma^2\right) + \frac{1}{2} = \frac{1}{2}\left(1+\log\left(2\pi\sigma^2\right)\right). \tag{8}$$

Assuming a random vector $\vec{x} \in \mathbb{R}^d$ the pdf of which is a multivariate Gaussian $N(\vec{\mu}, \Sigma)$, where $\vec{\mu}$ is the vector of means and Σ the covariance matrix:

$$p(\vec{x}; \vec{\mu}, \Sigma) = \frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma|^{\frac{1}{2}}}\exp\left(-\frac{1}{2}(\vec{x}-\vec{\mu})^T\Sigma^{-1}(\vec{x}-\vec{\mu})\right). \tag{9}$$

Thus, the logarithm of pdf is given by:

$$\log p(\vec{x}; \vec{\mu}, \Sigma) = -\frac{d}{2}\log(2\pi) - \frac{1}{2}\log|\Sigma| - \frac{1}{2}(\vec{x}-\vec{\mu})^T\Sigma^{-1}(\vec{x}-\vec{\mu}), \tag{10}$$

which leads to (LEVADA, 2019):

$$\begin{aligned}
H(p) &= \frac{d}{2}\log(2\pi) + \frac{1}{2}\log|\Sigma| + \frac{1}{2}E\left[(\vec{x}-\vec{\mu})^T\Sigma^{-1}(\vec{x}-\vec{\mu})\right] \\
&= \frac{d}{2}\log(2\pi) + \frac{1}{2}\log|\Sigma| + \frac{1}{2}E\left[\mathrm{Tr}\left((\vec{x}-\vec{\mu})^T\Sigma^{-1}(\vec{x}-\vec{\mu})\right)\right] \\
&= \frac{d}{2}\log(2\pi) + \frac{1}{2}\log|\Sigma| + \frac{1}{2}E\left[\mathrm{Tr}\left(\Sigma^{-1}(\vec{x}-\vec{\mu})(\vec{x}-\vec{\mu})^T\right)\right] \\
&= \frac{d}{2}\log(2\pi) + \frac{1}{2}\log|\Sigma| + \frac{1}{2}\mathrm{Tr}\left(\Sigma^{-1}E\left[(\vec{x}-\vec{\mu})(\vec{x}-\vec{\mu})^T\right]\right) \\
= \frac{d}{2}\log(2\pi) + \frac{1}{2}\log|\Sigma| + \frac{1}{2}\mathrm{Tr}\left(\Sigma^{-1}\Sigma\right) &= \frac{d}{2}\log(2\pi) + \frac{1}{2}\log|\Sigma| + \frac{1}{2}\mathrm{Tr}(\mathcal{I}) \\
&= \frac{1}{2}\log|\Sigma| + \frac{d}{2}(1+\log(2\pi)).
\end{aligned} \tag{11}$$

It can be noted that, as in the univariate case, the entropy of a random Gaussian vector does not depend on the mean.

2.2.1.1 Joint Entropy

The contents of a message can be described by more than one random variable. Thus, it might be interesting to measure the uncertainty over message elements when it is composed of more than one variable. To do that, a measure called joint entropy is used, written as (SHANNON; WEAVER, 1964):

$$H(x,y) = -\sum_{i,j} p(i,j)\log p(i,j), \tag{12}$$

where i and j are indices of variables x and y, respectively, and $p(i,j)$ the probability of the combination of elements i in x and j in y occurring, and:

$$H(x) = -\sum_{i,j} p(i,j)\log\sum_{j} p(i,j), \tag{13}$$

$$H(y) = -\sum_{i,j} p(i,j)\log\sum_{i} p(i,j). \tag{14}$$

Moreover:

$$H(x,y) \leq H(x) + H(y), \tag{15}$$

with equality only if the variables are independent, that is, $p(i,j) = p(i)\,p(j)$. The uncertainty of a joint event is lesser or equal than the sum of individual uncertainties.

2.2.1.2 Conditional Entropy

Assuming two random variables x and y, not necessarily independent, uncertainty over the occurrence of an element of y, for example, when an element of x occurs can be measured, that is, the probability of occurring that an element of y in bound to one in x. This measure is named conditional entropy and is calculated as:

$$H_x(y) = -\sum_{i,j} p(i,j) \log p_i(j). \tag{16}$$

It quantifies the mean uncertainty over y when x is known. Substituting $p_i(j)$, it becomes:

$$H_x(y) = -\sum_{i,j} p(i,j) \log p(i,j) + \sum_{i,j} p(i,j) \log \sum_i p(i,j) = H(x,y) - H_x(y), \tag{17}$$

or:

$$H(x,y) = H(x) + H_x(y) = H(y) + H_y(x). \tag{18}$$

The uncertainty over the set x, y is the uncertainty over x plus the uncertainty over y when x is known (SHANNON; WEAVER, 1964).

2.2.2 Kullback-Leibler Divergence

To measure the proximity between two probability distribution functions of a random variable, one uses relative entropy, also known as Kullback-Leibler Divergence (KL-divergence) (KULLBACK; LEIBLER, 1951), which is a measure of similarity between two probability distribution functions, $p(x)$ and $q(x)$, relative to the same indices x. Relative entropy is defined by:

$$H(p(x) \,||\, q(x)) \equiv \sum_x p(x) \log \frac{p(x)}{q(x)} = -H(X) - \sum_x p(x) \log q(x), \tag{19}$$

where, by definition, $q(x) \to 0 \Rightarrow -p(x) \log \frac{p(x)}{q(x)} \to +\infty$, if $p(x) > 0$, that is, the event with probability $p(x)$ happens and the event with probability $q(x)$ does not.

From relative entropy comes an important theorem used to prove many results in information theory, named *Information Inequality*. According to this theorem, relative entropy is non-negative, that is:

$$H(p(x) \,||\, q(x)) \geq 0, \tag{20}$$

with equality if, and only if, $p(x) = q(x), \forall x$. Proof comes from inequality $\log x = \frac{\ln x}{\ln 2} \leq x - 1$, for with $x = \frac{1}{t}$:

$$\frac{1}{\ln 2}(\ln 1 - \ln t) \leq \frac{1}{t} - 1 \Rightarrow \frac{\ln t}{\ln 2} \geq 1 - \frac{1}{t}. \tag{21}$$

Using Equation (21), the conclusion is (COVER; THOMAS, 2006):

$$\begin{aligned} H(p(x) \mid\mid q(x)) = \sum_x p(x) \log \frac{p(x)}{q(x)} = \frac{1}{\ln 2} \sum_x p(x) \ln \frac{p(x)}{q(x)} \geq \sum_x p(x) \left(1 - \frac{q(x)}{p(x)}\right) \\ = \sum_x (p(x) - q(x)) = (1 - 1) = 0. \end{aligned} \tag{22}$$

This inequality happens if, and only if, $q(x) = p(x), \forall x$.

Similarly, the cross entropy between two pdfs can be defined as:

$$H(p, q) = -\int p(x) [\log q(x)] \, \mathrm{d}x. \tag{23}$$

KL-divergence, then, is the difference between the cross entropy of $p(x)$ and $q(x)$ and the entropy of $p(x)$, that is:

$$\begin{aligned} D_{KL}(p, q) = H(p, q) - H(p) = -\int p(x) [\log q(x)] \, \mathrm{d}x + \int p(x) [\log p(x)] \, \mathrm{d}x \\ = \int p(x) \log \left(\frac{p(x)}{q(x)}\right) \mathrm{d}x = E_p \left[\log \left(\frac{p(x)}{q(x)}\right)\right]. \end{aligned} \tag{24}$$

It must be noted that relative entropy is always non-negative, that is, $D_{KL}(p, q) \geq 0$, being $D_{KL}(p, q) = 0 \iff p(x) = q(x)$. First, there is $\log(a) \leq a - 1$ for $a > 0$, therefore:

$$\begin{aligned} -D_{KL}(p, q) = -\int p(x) \log \left(\frac{p(x)}{q(x)}\right) \mathrm{d}x = \int p(x) \log \left(\frac{q(x)}{p(x)}\right) \mathrm{d}x \\ \leq \int p(x) \left(\frac{p(x)}{q(x)} - 1\right) \mathrm{d}x = \int p(x) \, \mathrm{d}x - \int q(x) \, \mathrm{d}x = 1 - 1 = 0. \end{aligned} \tag{25}$$

Let $p(x)$ and $q(x)$ be Gaussian univariate densities, $N(\mu_1, \sigma_1^2)$ and $N(\mu_2, \sigma_2^2)$, then the KL-divergence is given by:

$$\begin{aligned} D_{KL}(p, q) = E_p \left[-\log \sigma_1 - \frac{1}{2\sigma_1^2}(x - \mu_1)^2 + \log \sigma_2 \frac{1}{2\sigma_2^2}(x - \mu_2)^2\right] \\ = \log \left(\frac{\sigma_2}{\sigma_1}\right) + \frac{1}{2\sigma_2^2} E_p \left[(x - \mu_2)^2\right] - \frac{1}{2\sigma_1^2} E_p \left[(x - \mu_1)^2\right]. \end{aligned} \tag{26}$$

One can easily notice that:

$$E_p \left[(x - \mu_1)^2\right] = \sigma_1^2, \tag{27}$$

$$E_p \left[(x - \mu_2)^2\right] = E\left[x^2\right] - 2E[x] \mu_2 + \mu_2^2, \tag{28}$$

$$E\left[x^2\right] = \mathrm{Var}[x] + E^2[x] = \sigma_1^2 + \mu_1^2, \tag{29}$$

which, finally, leads to:

$$D_{KL}(p,q) = \log\left(\frac{\sigma_2}{\sigma_1}\right) + \frac{1}{2\sigma_2^2}\left(\sigma_1^2 + \mu_1^2 - 2\mu_1\mu_2 + \mu_2^2\right) - \frac{1}{2}$$
$$= \log\left(\frac{\sigma_2}{\sigma_1}\right) + \frac{\sigma_1^2 + (\mu_1 - \mu_2)^2}{2\sigma_2^2} - \frac{1}{2}. \quad (30)$$

Assuming one wishes to compute KL-divergence between two multivariate Gaussian densities: $N(\vec{\mu}_1, \Sigma_1)$ and $N(\vec{\mu}_2, \Sigma_2)$. Considering parameters vector $\vec{\theta} = \{\vec{\mu}, \Sigma\}$, comes (DUCHI, 2007):

$$D_{KL}(p,q) = E\left[\log p\left(\vec{x};\vec{\theta}\right) - \log q\left(\vec{x};\vec{\theta}\right)\right]$$
$$= E\left[-\frac{1}{2}\log|\Sigma_1| - \frac{1}{2}(\vec{x}-\vec{\mu}_1)^T\Sigma_1^{-1}(\vec{x}-\vec{\mu}_1) + \frac{1}{2}\log|\Sigma_2| + \frac{1}{2}(\vec{x}-\vec{\mu}_2)^T\Sigma_2^{-1}(\vec{x}-\vec{\mu}_2)\right]$$
$$= \frac{1}{2}\log\left(\frac{|\Sigma_2|}{|\Sigma_1|}\right) - \frac{1}{2}E_p\left[(\vec{x}-\vec{\mu}_1)^T\Sigma_1^{-1}(\vec{x}-\vec{\mu}_1)\right] + \frac{1}{2}E_p\left[(\vec{x}-\vec{\mu}_2)^T\Sigma_2^{-1}(\vec{x}-\vec{\mu}_2)\right]$$
$$= \frac{1}{2}\log\left(\frac{|\Sigma_2|}{|\Sigma_1|}\right) - \frac{1}{2}E_p\left[\mathrm{Tr}\left[\Sigma_1^{-1}(\vec{x}-\vec{\mu}_1)(\vec{x}-\vec{\mu}_1)^T\right]\right] + \frac{1}{2}E_p\left[\mathrm{Tr}\left[\Sigma_2^{-1}(\vec{x}-\vec{\mu}_2)(\vec{x}-\vec{\mu}_2)^T\right]\right]$$
$$= \frac{1}{2}\log\left(\frac{|\Sigma_2|}{|\Sigma_1|}\right) - \frac{1}{2}\mathrm{Tr}\left[\Sigma_1^{-1}\Sigma_1\right] + \frac{1}{2}E_p\left[\mathrm{Tr}\left[\Sigma_2^{-1}\left(\vec{x}\vec{x}^T - 2\vec{x}\vec{\mu}_2^T + \vec{\mu}_2\vec{\mu}_2^T\right)\right]\right]$$
$$= \frac{1}{2}\log\left(\frac{|\Sigma_2|}{|\Sigma_1|}\right) - \frac{d}{2} + \frac{1}{2}\mathrm{Tr}\left[\Sigma_2^{-1}E_p\left[\left(\vec{x}\vec{x}^T - 2\vec{x}\vec{\mu}_2^T + \vec{\mu}_2\vec{\mu}_2^T\right)\right]\right]$$
$$= \frac{1}{2}\log\left(\frac{|\Sigma_2|}{|\Sigma_1|}\right) - \frac{d}{2} + \frac{1}{2}\mathrm{Tr}\left[\Sigma_2^{-1}\left(\Sigma_1 + \vec{\mu}_1\vec{\mu}_1^T - 2\vec{\mu}_2\vec{\mu}_1^T + \vec{\mu}_2\vec{\mu}_2^T\right)\right]$$
$$= \frac{1}{2}\log\left(\frac{|\Sigma_2|}{|\Sigma_1|}\right) - \frac{d}{2} + \frac{1}{2}\mathrm{Tr}\left[\Sigma_2^{-1}\Sigma_1\right] + \frac{1}{2}\left(\vec{\mu}_1^T\Sigma_2^{-1}\vec{\mu}_1 - 2\vec{\mu}_1^T\Sigma_2^{-1}\vec{\mu}_2 + \vec{\mu}_2^T\Sigma_2^{-1}\vec{\mu}_2\right)$$
$$= \frac{1}{2}\left[\log\left(\frac{|\Sigma_2|}{|\Sigma_1|}\right) - d + \mathrm{Tr}\left[\Sigma_2^{-1}\Sigma_1\right] + (\vec{\mu}_2-\vec{\mu}_1)^T\Sigma_2^{-1}(\vec{\mu}_2-\vec{\mu}_1)\right]. \quad (31)$$

It can be noted that, if covariance matrices are equal, KL-divergence is simplified to the Mahalanobis distance between means.

2.2.3 Bhattacharyya Distance

Another similarity measure between two probability distributions that can be applied on random variables is Bhattacharyya distance, given by (BHATTACHARYYA, 1943 apud CROOKS, 2017):

$$D_B(p,q) = -\log\sum_x \sqrt{p(x)\,q(x)}, \quad (32)$$

where p and q are probabilities that element x of a random variable occurs. Like KL-divergence, Bhattacharyya distance also depends solely on pdfs. Therefore, one can define a relationship between them using these functions as parameters.

Bhattacharyya distance is a similarity measure between pdfs derived in terms of the Bhattacharyya coefficient, which is a classic statistic measure of overlap between two

samples. It is a generalisation of the Mahalanobis distance in the sense that, when distributions have close means but different variances, the Mahalanobis distance tends to zero, while Bhattacharyya increases as variances grow apart. Bhattacharyya coefficient is defined by:

$$C_B(p,q) = \int \sqrt{p(x)\,q(x)}\,\mathrm{d}x = \int p(x)\sqrt{\frac{q(x)}{p(x)}}\,\mathrm{d}x = E_p\left[\sqrt{\frac{q(x)}{p(x)}}\right]. \tag{33}$$

Bhattacharyya distance is the negative of the logarithm of Bhattacharyya coefficient, that is:

$$D_B(p,q) = -\log C_B(p,q). \tag{34}$$

Since $\log(x)$ is a convex function, by Jensen's inequality, comes (LEVADA, 2019):

$$D_B(p,q) = -\log E_p\left[\sqrt{\frac{q(x)}{p(x)}}\right] \leq E_p\left[-\log\sqrt{\frac{q(x)}{p(x)}}\right] = \frac{1}{2}E_p\left[\log\left(\frac{p(x)}{q(x)}\right)\right], \tag{35}$$

which leads to:

$$D_{KL}(p,q) \geq 2D_B(p,q). \tag{36}$$

It can be shown that the Bhattacharyya coefficient for two univariate Gaussian densities is given by:

$$C_B(p,q) = \sqrt{\frac{2\sigma_1\sigma_2}{\sigma_1^2+\sigma_2^2}}\exp\left(-\frac{1}{4}\frac{(\mu_1-\mu_2)^2}{\sigma_1^2+\sigma_2^2}\right). \tag{37}$$

The Bhattacharyya coefficient for two multivariate Gaussian densities can be calculated as:

$$C_B(p,q) = \frac{|\Sigma_1|^{\frac{1}{4}}|\Sigma_2|^{\frac{1}{4}}}{\left|\frac{\Sigma_1+\Sigma_2}{2}\right|^{\frac{1}{2}}}\exp\left(-\frac{1}{8}(\vec{\mu}_1-\vec{\mu}_2)^T\left(\frac{\Sigma_1+\Sigma_2}{2}\right)^{-1}(\vec{\mu}_1-\vec{\mu}_2)\right). \tag{38}$$

Finally, the expression of Bhattacharyya distance for two multivariate Gaussian densities is:

$$D_B(p,q) = \frac{1}{8}(\vec{\mu}_1-\vec{\mu}_2)^T\Sigma^{-1}(\vec{\mu}_1-\vec{\mu}_2) + \frac{1}{2}\log\left(\frac{\Sigma}{\sqrt{|\Sigma_1|\,|\Sigma_2|}}\right), \tag{39}$$

where $\Sigma = \frac{\Sigma_1+\Sigma_2}{2}$.

Expressing Bhattacharyya distance in terms of the so-called Bhattacharyya coefficient defined by equation (33), which can be used in:

$$D_H(p,q) = \sqrt{1-C_B(p,q)}, \tag{40}$$

which is called *Hellinger distance*, the relation between which and KL-divergence is given by:

$$D_{KL}(p\,||\,q) \geq 2D_H^2(p,q) = 2\sqrt{1-C_B(p,q)}. \tag{41}$$

Therefore (DIFFERENCES..., 2014):

$$D_B(p,q) = -\log C_B(p,q) = -\log \int \sqrt{p(x)\,q(x)}\,\mathrm{d}x \overset{\text{def}}{=} -\log \int h(x)\,\mathrm{d}x$$
$$= -\log \int \frac{h(x)}{p(x)} p(x)\ \mathrm{d}x \leq \int -\log \frac{h(x)}{p(x)} p(x)\ \mathrm{d}x = \int -\frac{1}{2}\log \frac{h^2(x)}{p^2(x)} p(x)\ \mathrm{d}x$$
$$= \int -\frac{1}{2}\log \frac{q(x)}{p(x)} p(x)\ \mathrm{d}x = \frac{1}{2} D_{KL}(p \mid\mid q). \quad (42)$$

Thus, the inequality between these two distances is:

$$D_{KL}(p \mid\mid q) \geq 2 D_B(p,q). \quad (43)$$

since $-\log x \geq 1 - x$ for $0 \leq x \leq 1$:

$$D_{KL}(p \mid\mid q) \geq 2 D_B(p,q) \geq 2 D_H^2(p,q). \quad (44)$$

2.2.4 Hellinger Distance

A possible limitation of Bhattacharyya distance is that it does not obey triangular inequality. As an alternative, Hellinger distance was proposed to overcome this problem. Squared Hellinger distance is given by (LEVADA, 2019):

$$D_H^2(p,q) = \frac{1}{2}\int \left(\sqrt{p(x)} - \sqrt{q(x)}\right)^2 \mathrm{d}x$$
$$= \frac{1}{2}\left(\int p(x)\ \mathrm{d}x - 2\int \sqrt{p(x)\,q(x)}\,\mathrm{d}x + \int q(x)\ \mathrm{d}x\right)$$
$$= \frac{1}{2}\left(2 - 2\int \sqrt{p(x)\,q(x)}\,\mathrm{d}x\right) = 1 - C_B(p,q), \quad (45)$$

which leads to equation (40). It can be noted that, by definition, squared Hellinger distance is the integral of a non-negative function, which implies $D_H^2(p,q) \geq 0$. Since the Bhattacharyya coefficient is the integral of the square root of two densities, it has a lower bound in zero and a higher in one, that is $D_H^2(p,q) \leq 1$.

Another important statistic divergence is total variance distance, defined by:

$$D_{TV}(p,q) = \frac{1}{2}\int |p(x) - q(x)|\ \mathrm{d}x. \quad (46)$$

Since $D_H^2(p,q)$ is bound by $D_{TV}(p,q)$, comes:

$$D_H^2(p,q) = \frac{1}{2}\int \left|\sqrt{p(x)} - \sqrt{q(x)}\right| \left|\sqrt{p(x)} - \sqrt{q(x)}\right| \mathrm{d}x$$
$$\leq \frac{1}{2}\int \left|\sqrt{p(x)} - \sqrt{q(x)}\right| \left|\sqrt{p(x)} + \sqrt{q(x)}\right| \mathrm{d}x = \frac{1}{2}\int |p(x) - q(x)|\ \mathrm{d}x = D_{TV}(p,q). \quad (47)$$

Conversely, $D_{TV}(p,q)$ is bound by $\sqrt{2}D_H(p,q)$:

$$\begin{aligned} D_{TV}^2(p,q) &= \frac{1}{4}\left(\int |p(x)-q(x)|\,\mathrm{d}x\right)^2 \\ &= \frac{1}{4}\left(\int \left(\sqrt{p(x)}-\sqrt{q(x)}\right)\left(\sqrt{p(x)}+\sqrt{q(x)}\right)\mathrm{d}x\right)^2 \\ &\leq \frac{1}{4}\left(\int \left(\sqrt{p(x)}-\sqrt{q(x)}\right)^2\mathrm{d}x\right)\left(\int \left(\sqrt{p(x)}+\sqrt{q(x)}\right)^2\mathrm{d}x\right), \end{aligned} \tag{48}$$

in which the Cauchy-Schwarz inequality is applied to the second row. The number in the first parentheses is exactly twice that of squared Hellinger distance, that is, $2D_H^2(p,q)$, and the second can be simplified to:

$$\int \left(\sqrt{p(x)}+\sqrt{q(x)}\right)^2\mathrm{d}x = 2+2\int\sqrt{p(x)\,q(x)}\,\mathrm{d}x = 2+2C_B(p,q). \tag{49}$$

From previous calculations, it is known that:

$$2D_H^2(p,q) = 2-2C_B(p,q), \tag{50}$$

which implies:

$$2C_B(p,q) = 2-2D_H^2(p,q), \tag{51}$$

that leads to:

$$D_{TV}^2(p,q) \leq \frac{1}{2}D_H^2(p,q)\left(2-2D_H^2(p,q)\right) \leq \sqrt{2}D_H(p,q). \tag{52}$$

For the case of two multivariate Gaussian densities, it can be easily noted that equation (38) can be replaced in equation (45), leading to:

$$D_H^2(p,q) = 1-\frac{|\Sigma_1|^{\frac{1}{4}}|\Sigma_2|^{\frac{1}{4}}}{|\Sigma|^{\frac{1}{2}}}\exp\left(-\frac{1}{8}(\vec{\mu}_1-\vec{\mu}_2)^T\Sigma^{-1}(\vec{\mu}_1-\vec{\mu}_2)\right). \tag{53}$$

2.2.5 Cauchy-Schwarz Divergence

Shannon Entropy has an important part in Information Theory, however there are other definitions of entropy such as the one given by the so-called Rényi or Quadratic Entropy:

$$H_{R_\alpha}(p) = \frac{1}{1-\alpha}\log\left(\int p^\alpha(x)\,\mathrm{d}x\right). \tag{54}$$

When $\alpha = 2$, it becomes:

$$H_R(p) = -\log\left(\int p^2(x)\,\mathrm{d}x\right). \tag{55}$$

The equivalent to KL-divergence for Quadratic Entropy is Cauchy-Schwarz divergence (SPUREK; PAŁKA, 2016):

$$D_{CS}(p,q) = \log\left(\int p^2(x)\,\mathrm{d}x\right)+\log\left(\int q^2(x)\,\mathrm{d}x\right)-2\log\left(\int p(x)\,q(x)\,\mathrm{d}x\right). \tag{56}$$

Calculations for multivariate Gaussian distributions follow simply:

$$D_{CS}(p,q) = -\frac{1}{2}\log\left(|4\Sigma_1\Sigma_2|\right)+\log\left(|\Sigma_1+\Sigma_2|\right)+(\vec{\mu}_1-\vec{\mu}_2)^T\Sigma^{-1}(\vec{\mu}_1-\vec{\mu}_2). \tag{57}$$

2.3 Dimensionality Reduction for Metrics Learning

Linear DR methods have been developed for many areas of science, such as statistics, optimisation, machine learning, and other applied fields for over a century, and became powerful mathematical tools for analysing noisy and high dimensional data. Part of linear methods' success is due to the fact they have simple geometric interpretations and usually attractive computational properties. Basically, linear methods search for a T matrix which maps the original feature space samples ($\mathbb{R}^m$) in a linear space $\mathbb{R}^d$, where $d < m$. There are many ways to define linear DR methods (CUNNINGHAM; GHAHRAMANI, 2015).

Definition 1. (Linear DR). Given n m-dimensional data points, $X = [\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n] \in \mathbb{R}^{m \times n}$, and a dimensionality choice $d < m$, optimise a goal function $fx(\cdot)$ to create a linear transform $T \in \mathbb{R}^{d \times m}$, and name $Y = TX \in \mathbb{R}^{d \times n}$ the low dimensional transformed data.

2.4 Kernel Density Estimation

Kernel Density Estimation (KDE) is a non-parametric statistical technique to estimate the pdf of a random variable (ROSENBLATT, 1956; PARZEN, 1962). Let $\{x_1, x_2, \ldots, x_n\}$ be an independent and identically distributed (iid) sample from a 1D random variable x with unknown density function $f(x)$. The KDE of $f(x)$ is given by:

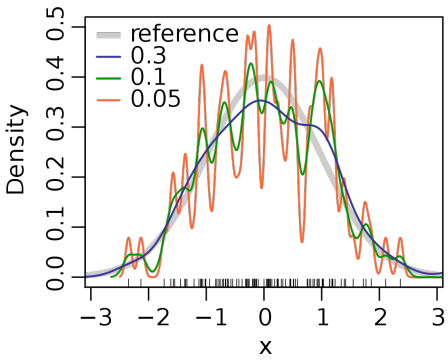
$$\hat{f}_h(x) = \frac{1}{n}\sum_{i=1}^{n} K_h(x - x_i) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right), \tag{58}$$

where $K(x)$ is the kernel function and h is the bandwidth, a parameter that controls the degree of smoothing of the density estimate. Several kernel functions have been proposed and applied with success in many problems: uniform, triangular, Gaussian, and Epanechnikov are among the most important ones. In this work, Gaussian kernels are used, as they can provide a reasonable approximation for many distributions in the studied datasets while maintaining properties that allow for some simplification of otherwise costly calculations (HONARKHAH; CAERS, 2010).

2.4.1 Bandwidth estimation methods

The choice of bandwidth h is crucial for the correct estimation of the unknown density function. Large values of h cause over-smoothing, making $\hat{f}_h(x)$ unimodal and with large variance. Small values of h often cause the emergence of noise, causing large variations in $\hat{f}_h(x)$ for nearby points in the neighbourhood of x. Often, the optimal bandwidth value is a trade-off between data fidelity and a smooth constraint. Figure 2 shows an illustration of different values of h in the KDE of a Gaussian pdf.

Figure 2 – The effect of bandwidth selection in Kernel Density Estimation of a Gaussian probability density function.



Source: (CERVATI NETO; LEVADA; HADDAD, 2024).

2.4.1.1 Silverman's rule

It has been shown that if both the kernel function and the unknown density are Gaussian, then the optimal bandwidth in terms of minimum integrated mean square error (IMSE) can be computed by (SILVERMAN, 1986):

$$h_{SIL} = 0.9 \min\left(\hat{\sigma}, \frac{IQR}{1.34}\right) n^{-\frac{1}{5}}, \tag{59}$$

where $\hat{\sigma}$ is the standard deviation of the samples, $IQR = Q_3 - Q_1$ is the interquartile range and n is the sample size.

2.4.1.2 Scott's rule

Under the Gaussian assumption, the following bandwidth estimation rule is also optimal in terms of IMSE (SCOTT, 1979):

$$h_{SC} = 3.49\hat{\sigma}n^{-\frac{1}{3}}, \tag{60}$$

where $\hat{\sigma}$ is the standard deviation of the samples, and n is the sample size.

2.5 Principal Component Analysis

Principal Component Analysis (PCA) is a computational method which implements the Karhunen-Loève transform, also known as Hotelling transform, a classic multivariate that expands a given vector $\vec{x} \in \mathbb{R}^m$ into the eigenvalues of its covariance matrix (JOLLIFFE, 2002), being the most known method for data compression and feature extraction. PCA is a second order statistic method, as it depends only on the covariance matrix, and is optimal for maximising the variance of new compact representation Y and minimise the square mean error between it and the original data X. The purpose of PCA is, from a

statistic point of view, to reduce redundancy between the random variables which make up vector $\vec{x} \in \mathbb{R}^m$, measured by the correlations between them. In that regard, PCA first decorrelates existing features and then reduces dimensionality by finding new ones which are linear combinations of the originals.

Let $Z = \left[T^T, S^T\right]$ be an orthonormal basis for $\mathbb{R}^m$ in which $T^T = [\vec{w}_1, \vec{w}_2, \ldots, \vec{w}_d]$ denotes the $d < m$ components one wishes to keep during the DR process, and $S^T = [\vec{w}_{d+1}, \vec{w}_{d+2}, \ldots, \vec{w}_m]$ the remainder that must be discarded. That is, T defines the linear subspace of PCA and S the one eliminated by the reduction process (YOUNG; CALVERT, 1974).

The problem can be summed up as: given an input feature space, find d $\vec{w}_j$ directions, for $j = 1, 2, \ldots, d$, such that, when the data are projected, variance is maximised, that is, directions that maximise data spread. The goal is, therefore, to obtain directions $\vec{w}_j$. It is assumed that, without loss of generality, sample $X = [\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n]$ has mean zero, that is, data points are centred around the origin.

$\vec{x} \in \mathbb{R}^m$ can be written as an expansion of orthonormal basis Z as:

$$\vec{x} = \sum_{j=1}^{m} \left(\vec{x}^T \vec{w}_j\right) \vec{w}_j = \sum_{j=1}^{m} \vec{c}_j \vec{w}_j, \tag{61}$$

where $\vec{c}_j$ are expansion coefficients. Thus, new vector $\vec{y} \in \mathbb{R}^d$ can be found using transform $\vec{y} = T\vec{x}$, as:

$$\vec{y}^T = \vec{x}^T T^T = \sum_{j=1}^{m} \vec{c}_j \vec{w}_j^T \left[\vec{w}_1, \vec{w}_2, \ldots, \vec{w}_d\right]. \tag{62}$$

Having an orthonormal basis, $\vec{w}_i^T \vec{w}_j = 1$ for $i = j$ and $\vec{w}_i^T \vec{w}_j = 0$ for $i \neq j$, resulting in:

$$\vec{y}^T = [c_1, c_2, \ldots, c_d]. \tag{63}$$

Hence, a linear transform T that maximises the variance retained in data is sought, that is, to maximise functional (HYVARINEN; KARHUNEN; OJA, 2001):

$$J_1^{PCA}(T) = E\left[\|\vec{y}\|^2\right] = E\left[\vec{y}^T \vec{y}\right] = \sum_{j=1}^{d} E\left[c_j^2\right]. \tag{64}$$

As c_j is the projection of $\vec{x}$ in $\vec{w}_j$, leads to:

$$J_1^{PCA}(T) = \sum_{j=1}^{d} E\left[\vec{w}_j^T \vec{x} \vec{x}^T \vec{w}_j\right] = \sum_{j=1}^{d} \vec{w}_j^T E\left[\vec{x}\vec{x}^T\right] \vec{w}_j = \sum_{j=1}^{d} \vec{w}_j^T \Sigma_x \vec{w}_j, \tag{65}$$

where Σ_x denotes the covariance matrix in data points X. Therefore, the constrained optimisation problem is given by:

$$\arg\max_{\vec{w}_j} \sum_{j=1}^{d} \vec{w}_j^T \Sigma_x \vec{w}_j \text{ subject to } \|\vec{w}_j\| = 1 \text{ for } j = 1, 2, \ldots, d, \tag{66}$$

which is solved using Lagrange multipliers. The Lagrangian function is given by:

$$J_1^{PCA}(T, \lambda_1, \lambda_2, \ldots, \lambda_d) = \sum_{j=1}^{d} \vec{w}_j^T \Sigma_x \vec{w}_j - \sum_{j=1}^{d} \lambda_j \left(\vec{w}_j^T \vec{w}_j - 1 \right). \tag{67}$$

Differentiating with respect to $\vec{w}_j$ and equating to zero, the necessary condition for optimality is found:

$$\frac{\partial}{\partial \vec{w}_j} J_1^{PCA}(T, \lambda_1, \lambda_2, \ldots, \lambda_d) = \Sigma_x \vec{w}_j - \lambda_j \vec{w}_j = 0, \tag{68}$$

which leads to the eigenvectors equation:

$$\Sigma_x \vec{w}_j = \lambda_j \vec{w}_j. \tag{69}$$

The optimisation problem can, therefore, be rewritten as:

$$\arg\max_{\vec{w}_j} \sum_{j=1}^{d} \vec{w}_j^T \Sigma_x \vec{w}_j = \arg\max_{\vec{w}_j} \sum_{j=1}^{d} \vec{w}_j^T \lambda_j \vec{w}_j = \arg\max_{\vec{w}_j} \sum_{j=1}^{d} \lambda_j, \tag{70}$$

which means that the k eigenvectors assigned to the k largest eigenvalues must be chosen to compose the basis of a linear subspace of PCA.

Another optimal property of the PCA subspace is the minimisation of mean square error between X and Y, being PCA approximation the best representation in terms of data compression. Let mean square error between random vectors $\vec{x} \in \mathbb{R}^m$ and $\vec{y} \in \mathbb{R}^d$:

$$J_2^{PCA}(T) = E\left[\|\vec{x} - \vec{y}\|^2 \right] = E\left[\left\| \vec{x} - \sum_{j=1}^{d} \left(\vec{w}_j^T \vec{x} \right) \vec{w}_j \right\|^2 \right]. \tag{71}$$

By expanding the norm, a second expression for mean square error is:

$$J_2^{PCA}(T) = E\left[\left(\vec{x} - \sum_{j=1}^{d} \left(\vec{w}_j^T \vec{x} \right) \vec{w}_j \right)^T \left(\vec{x} - \sum_{j=1}^{d} \left(\vec{w}_j^T \vec{x} \right) \vec{w}_j \right) \right]. \tag{72}$$

Applying the distributive property:

$$J_2^{PCA}(T) = E\left[\vec{x}^T \vec{x} - \vec{x}^T \left(\sum_{j=1}^{d} \left(\vec{w}_j^T \vec{x} \right) \vec{w}_j \right) - \left(\sum_{j=1}^{d} \left(\vec{w}_j^T \vec{x} \right) \vec{w}_j \right)^T \vec{x} + \left(\sum_{j=1}^{d} \left(\vec{w}_j^T \vec{x} \right) \vec{w}_j \right) \left(\sum_{j=1}^{d} \left(\vec{w}_j^T \vec{x} \right) \vec{w}_j \right)^T \right]. \tag{73}$$

Using the linearity of expected value and rearranging terms:

$$J_2^{PCA}(T) = E\left[\|\vec{x}\|^2 \right] - E\left[\sum_{j=1}^{d} \left(\vec{w}_j^T \vec{x} \right) \left(\vec{w}_j^T \vec{x} \right) \right] - E\left[\sum_{j=1}^{d} \vec{w}_j^T \left(\vec{x}^T \vec{w}_j \right) \vec{x} \right] + E\left[\left(\sum_{j=1}^{d} \vec{w}_j^T \left(\vec{x}^T \vec{w}_j \right) \right) \left(\sum_{j=1}^{d} \left(\vec{w}_j^T \vec{x} \right) \vec{w}_j \right) \right]. \tag{74}$$

Simplifying inner products:

$$J_2^{PCA}(T) = E\left[\|\vec{x}\|^2\right] - \sum_{j=1}^{d} E\left[\left(\vec{w}_j^T\vec{x}\right)^2\right] - \sum_{j=1}^{d} E\left[\left(\vec{w}_j^T\vec{x}\right)^2\right] + \sum_{j=1}^{d}\sum_{k=1}^{d} E\left[\left(\vec{w}_j^T\vec{x}\right)\left(\vec{x}^T\vec{w}_k\right)\vec{w}_j^T\vec{w}_k\right]. \quad (75)$$

Since $\vec{w}_j$ for $j = 1, 2, \ldots, d$ defines a set of orthonormal vectors:

$$J_2^{PCA}(T) = E\left[\|\vec{x}\|^2\right] - \sum_{j=1}^{d} E\left[\left(\vec{w}_j^T\vec{x}\right)^2\right] - \sum_{j=1}^{d} E\left[\left(\vec{w}_j^T\vec{x}\right)^2\right] + \sum_{j=1}^{d} E\left[\left(\vec{w}_j^T\vec{x}\right)^2\right] = E\left[\|\vec{x}\|^2\right] - \sum_{j=1}^{d} E\left[\left(\vec{w}_j^T\vec{x}\right)^2\right] = E\left[\|\vec{x}\|^2\right] - \sum_{j=1}^{d} \vec{w}_j^T\Sigma_x\vec{w}_j. \quad (76)$$

Being that the first term is constant, as it does not depend on $\vec{w}_j$, the optimisation problem is given by:

$$\arg\min_{\vec{w}_j} \; -\sum_{j=1}^{d} \vec{w}_j^T\Sigma_x\vec{w}_j \text{ subject to } \|\vec{w}_j\| = 1 \text{ for } j = 1, 2, \ldots, d, \quad (77)$$

which equates to maximising variance.

As previously mentioned, an interesting property of PCA is data decorrelation. This can be shown by using the spectral decomposition of Σ_x in $Q\Lambda Q^T$, where Q is the matrix of m eigenvalues of Σ_x and $\Lambda = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_m)$ is the diagonal matrix of eigenvalues of Σ_x. It is known that, before DR, $\vec{y} = Z\vec{x}$, where $Z = \left[T^T, S^T\right]$ and the covariance matrix of the transformed vector is given by $\Sigma_y = Z^T\Sigma_x Z$. In PCA, however, Z is made up of the eigenvalues of covariance matrix, therefore, $Z = Q$, which leads to $\Sigma_y = Z^TQ\Lambda Q^TZ = Q^TQ\Lambda Q^TQ = \Lambda$, where the orthonormality of eigenvectors implies $Q^TQ = \mathcal{I}$.

2.6 Kernel Principal Component Analysis

PCA only provides linear DR. However, if the data are structured as non-linear functions of the original features, it does not obtain relevant information. Kernel Principal Component Analysis (KPCA) allows generalising PCA for non-linear DR. (SCHÖLKOPF; SMOLA; MÜLLER, 1999).

The Vapnik-Chervonenkis theory shows that, under certain circumstances, mappings that lead to a higher dimensionality space than the input's provide more classification power (VAPNIK, 1993). Nevertheless, mapping to a higher dimensionality space can significantly increase computational cost. This can be mitigated using the so-called *kernel trick*: given an algorithm that can be expressed only in terms of its inner products, it can be constructed as different linear versions of itself (THEODORIDIS; KOUTROUMBAS,

2008). The main idea behind KPCA is using this *kernel trick* to compute the inner product of a higher dimensionality space without the need to project the data, allowing for the extraction of up to n (number of samples) non-linear principal components foregoing costly computations (SCHÖLKOPF; SMOLA; MÜLLER, 1998).

Let $\phi(\vec{x})$ be a non-linear mapping of the original m-dimensional input space for the M-dimensional feature space, where $M > m$. It is assumed, initially, that the mean of data after mapping to the higher dimensional space is zero:

$$\frac{1}{n}\sum_{i=1}^{n}\phi(\vec{x}_i) = 0. \tag{78}$$

Thus, the covariance matrix of the samples of projected data $M \times M$ is given by:

$$C = \frac{1}{n}\sum_{i=1}^{n}\phi(\vec{x}_i)\,\phi(\vec{x}_i)^T, \tag{79}$$

and the eigenvalues of C are:

$$C\vec{v}_k = \lambda_k\vec{v}_k \text{ for } k = 1, 2, \ldots, M. \tag{80}$$

The following result demonstrates that the eigenvalues of the covariance matrix can be written in terms of $\phi(\vec{x}_i)$.

Theorem 1. The eigenvalues of C can be expressed as a linear combination of its features, that is:

$$\vec{v}_k = \sum_{i=1}^{n}\alpha_{ki}\phi(\vec{x}_i). \tag{81}$$

From equations (79) and (80), comes:

$$C\vec{v}_k = \frac{1}{n}\sum_{i=1}^{n}\phi(\vec{x}_i)\,\phi(\vec{x}_i)^T\vec{v}_k = \lambda\vec{v}_k, \tag{82}$$

which implies:

$$\vec{v}_k = \frac{1}{n\lambda_k}\sum_{i=1}^{n}\left(\phi(\vec{x}_i)^T\vec{v}_k\right)\phi(\vec{x}_i) = \sum_{i=1}^{n}\alpha_{ki}\phi(\vec{x}_i), \tag{83}$$

where $\alpha_{ki} = \frac{1}{n\lambda_k}\phi(\vec{x}_i)^T\vec{v}_k$. Then, finding the eigenvalues equals finding coefficients α_{ki}. Replacing (83) in (82), gives:

$$\frac{1}{n}\sum_{i=1}^{n}\phi(\vec{x}_i)\,\phi(\vec{x}_i)^T\left(\sum_{j=1}^{n}\alpha_{kj}\phi(\vec{x}_j)\right) = \lambda_k\sum_{j=1}^{n}\alpha_{kj}\phi(\vec{x}_j), \tag{84}$$

which can be rewritten as:

$$\frac{1}{n}\sum_{i=1}^{n}\phi(\vec{x}_i)\left(\sum_{j=1}^{n}\alpha_{kj}\phi(\vec{x}_i)^T\phi(\vec{x}_j)\right) = \lambda_k\sum_{j=1}^{n}\alpha_{kj}\phi(\vec{x}_j). \tag{85}$$

Using the *kernel trick*, that is, $K(\vec{x}_i, \vec{x}_j) = \phi(\vec{x}_i)^T \phi(\vec{x}_j)$, yields:

$$\frac{1}{n}\sum_{i=1}^{n}\phi(\vec{x}_i)\left(\sum_{j=1}^{n}\alpha_{kj}K(\vec{x}_i,\vec{x}_j)\right) = \lambda_k\sum_{j=1}^{n}\alpha_{kj}\phi(\vec{x}_j). \tag{86}$$

Multiplying both sides by $\phi(\vec{x}_l)^T$ results in:

$$\frac{1}{n}\sum_{i=1}^{n}\phi(\vec{x}_l)^T\phi(\vec{x}_i)\left(\sum_{j=1}^{n}\alpha_{kj}K(\vec{x}_i,\vec{x}_j)\right) = \lambda_k\sum_{j=1}^{n}\alpha_{kj}\phi(\vec{x}_l)^T\phi(\vec{x}_j). \tag{87}$$

Using the *kernel trick* again:

$$\frac{1}{n}\sum_{i=1}^{n}K(\vec{x}_l,\vec{x}_i)\left(\sum_{j=1}^{n}\alpha_{kj}K(\vec{x}_i,\vec{x}_j)\right) = \lambda_k\sum_{j=1}^{n}\alpha_{kj}K(\vec{x}_l,\vec{x}_j). \tag{88}$$

Using the matrix-vector notation, the equation can be written as (SCHÖLKOPF; SMOLA; MÜLLER, 1999):

$$K^2\vec{\alpha}_k = (\lambda_k n)K\vec{\alpha}_k, \tag{89}$$

where $K_{i,j} = K(\vec{x}_i, \vec{x}_j)$ e $\vec{\alpha}_k$ is the n-dimensional column vector of α_{ki}, that is, $\alpha_{ki} = [\alpha_{k1}, \alpha_{k2}, \ldots, \alpha_{kn}]^T$. Simplifying it, comes to:

$$K\vec{\alpha}_k = (\lambda_k n)\vec{\alpha}_k, \tag{90}$$

showing that $\vec{\alpha}_k$ are the eigenvectors of the kernel matrix. There is a condition for normalising eigenvectors $\vec{\alpha}_k$. Firstly, it is known that $\vec{v}_k^T\vec{v}_k = 1$, which implies:

$$\sum_{r=1}^{n}\sum_{s=1}^{n}\alpha_{kr}\alpha_{ks}\phi(\vec{x}_r)^T\phi(\vec{x}_s) = 1 \implies \vec{\alpha}_k^T K\vec{\alpha}_k = 1. \tag{91}$$

Multiplying equation (90) by $\vec{\alpha}_k^T$:

$$\vec{\alpha}_k^T K\vec{\alpha}_k = (\lambda_k n)\vec{\alpha}_k^T\vec{\alpha}_k \implies (\lambda_k n)\vec{\alpha}_k^T\vec{\alpha}_k = 1 \implies \vec{\alpha}_k^T\vec{\alpha}_k = \frac{1}{n\lambda_k}. \tag{92}$$

For a new point $\vec{x}$, its projection over the k-th principal component is given by:

$$y_k(\vec{x}) = \phi(\vec{x})^T\vec{v}_k = \sum_{i=1}^{n}\alpha_{ki}\phi(\vec{x})^T\phi(\vec{x}_i) = \sum_{i=1}^{n}\alpha_{ki}K(\vec{x},\vec{x}_i). \tag{93}$$

The advantage of using the kernel trick is not needing to compute $\phi(\vec{x}_i)$ explicitly for $i = 1, 2, \ldots, n$, such that the kernel matrix can be constructed from training data. Two widely used non-linear kernels are the polynomial:

$$K(\vec{x},\vec{y}) = \left(\vec{x}^T\vec{y} + c\right)^d, \tag{94}$$

where $c \geq 0$ is a constant, and the Gaussian kernel:

$$K(\vec{x},\vec{y}) = \exp\left(-\frac{\|\vec{x}-\vec{y}\|^2}{2\sigma^2}\right), \tag{95}$$

with parameter σ^2. If the projected data have zero mean, they need to be centred:

$$\tilde{\phi}\left(\vec{x}_i\right) = \phi\left(\vec{x}_i\right) - \frac{1}{n}\sum_{k=1}^{n}\phi\left(\vec{x}_k\right). \tag{96}$$

Hence, the corresponding kernel matrix is given by:

$$\begin{aligned}\tilde{K}\left(\vec{x}_i,\vec{x}_j\right) &= \tilde{\phi}\left(\vec{x}_i\right)^T\tilde{\phi}\left(\vec{x}_j\right) = \left(\phi\left(\vec{x}_i\right) - \frac{1}{n}\sum_{k=1}^{n}\phi\left(\vec{x}_k\right)\right)^T\left(\phi\left(\vec{x}_j\right) - \frac{1}{n}\sum_{k=1}^{n}\phi\left(\vec{x}_k\right)\right)\\ &= \phi\left(\vec{x}_i\right)^T\phi\left(\vec{x}_j\right) - \frac{1}{n}\sum_{k=1}^{n}\phi\left(\vec{x}_i\right)^T\phi\left(\vec{x}_k\right) - \frac{1}{n}\sum_{k=1}^{n}\phi\left(\vec{x}_k\right)^T\phi\left(\vec{x}_j\right) + \frac{1}{n^2}\sum_{k=1}^{n}\sum_{l=1}^{n}\phi\left(\vec{x}_k\right)^T\phi\left(\vec{x}_l\right)\\ &= K\left(\vec{x}_i,\vec{x}_j\right) - \frac{1}{n}\sum_{k=1}^{n}K\left(\vec{x}_i,\vec{x}_k\right) - \frac{1}{n}\sum_{k=1}^{n}K\left(\vec{x}_k,\vec{x}_j\right) + \frac{1}{n^2}\sum_{k=1}^{n}\sum_{l=1}^{n}K\left(\vec{x}_k,\vec{x}_l\right).\end{aligned} \tag{97}$$

In matrix form, kernel matrix K is replaced by Gram matrix $\tilde{K}$:

$$\tilde{K} = K - 1_n K - K 1_n + 1_n K 1_n, \tag{98}$$

where 1_n is the $n \times n$ matrix with all elements equalling $\frac{1}{n}$.

2.7 Isometric Feature Mapping

Isometric Feature Mapping (ISOMAP) was one of the first algorithms for non-linear DR. The proposed approach combines the main features of the PCA and Multidimensional Scaling (MDS) algorithms (COX; COX, 2001; BORG; GROENEN, 2005) — computational efficiency, discovery of global optimality, and guarantee of asymptotic convergence — with the flexibility to learn a wide class of non-linear manifolds (TENENBAUM; SILVA; LANGFORD, 2000). The main idea behind the ISOMAP algorithm is to initially construct a graph connecting the k Nearest Neighbours (KNN) in the input space, obtain the shortest paths between each pair of vertices in this graph, and then, knowing the approximated geodesic distances between points, find a mapping in the Euclidean subspace $\mathbb{R}^d$ which preserves these distances.

The hypothesis of the ISOMAP algorithm is that the shortest paths in a KNN graph are good approximations for the actual geodesic distances in the manifold. It has been shown that, both for graphs based on the ϵ-neighbourhood rule as for on KNN, under certain regularity conditions, the following result is valid (BERNSTEIN et al., 2000).

Theorem 2. (Asymptotic Convergence Theorem) Given $\lambda_1, \lambda_2, \mu > 0$, for a large enough number of samples $\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n \in \mathbb{R}^m$, inequality

$$\left(1-\lambda_1\right) d_M\left(\vec{x}_i,\vec{x}_j\right) \leq d_G\left(\vec{x}_i,\vec{x}_j\right) \leq \left(1-\lambda_2\right) d_M\left(\vec{x}_i,\vec{x}_j\right) \tag{99}$$

is satisfied with probability $(1-\mu)$, where $d_G\left(\vec{x}_i,\vec{x}_j\right)$ is the approximation of the shortest path in the graph and $d_M\left(\vec{x}_i,\vec{x}_j\right)$ the geodesic distance in the manifold.

The ISOMAP algorithm can be divided into three main steps:

1. From input data $\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n \in \mathbb{R}^m$, construct an undirected proximity graph using either the KNN or ϵ-neighbourhood rule (LUXBURG, 2007);
2. Compute pairwise distances matrix D using n runs of the Dijkstra algorithm or one of Floyd-Warshall (CORMEN et al., 2009);
3. Estimate new coordinates for the points in an Euclidean subspace of $\mathbb{R}^d$ preserving distances using MDS.

The algorithms states that the embedding is constructed by the MDS method. Given its relevance for ISOMAP, it is described next.

2.7.1 Multidimensional Scaling

The main goal of MDS is to, given a $n \times n$ pairwise distances matrix, retrieve the coordinates of the n $\vec{x}_r \in \mathbb{R}^d$ points for $r = 1, 2, \ldots, n$ in an Euclidean subspace where d, target dimensionality, is a parameter of the algorithm (COX; COX, 2001; BORG; GROENEN, 2005). First, it is noted that the pairwise distances matrix is given by $D = \{d_{rs}^2\}$, for $r, s = 1, 2, \ldots, n$, where the distance between two arbitrary points $\vec{x}_r$ and $\vec{x}_s$ is:

$$d_{rs}^2 = \|\vec{x}_r - \vec{x}_s\|^2 = (\vec{x}_r - \vec{x}_s)^T (\vec{x}_r - \vec{x}_s). \tag{100}$$

Let B be the matrix of inner products, that is, $B = \{b_{rs}\}$, where $b_{rs} = \vec{x}_r^T \vec{x}_s$, for which MDS needs to find the embedding. Therefore, there are two problems to be solved: matrix B needs to be obtained from D and the coordinates of points in matrix B must be retrieved.

2.7.2 Finding the B matrix

In order to solve the first problem, it is assumed that the data has mean zero, that is:

$$\sum_{r=1}^{n} \vec{x}_n = 0, \tag{101}$$

otherwise there would be infinite different solutions, as the application of any arbitrary translation in the set would preserve pairwise distance. From equation (100), applying the distributive property, comes:

$$d_{rs}^2 = \vec{x}_r^T \vec{x}_r + \vec{x}_s^T \vec{x}_s - 2\vec{x}_r^T \vec{x}_s. \tag{102}$$

From matrix D, the mean of an arbitrary column s can be computed by:

$$\begin{aligned} \frac{1}{n}\sum_{r=1}^{n} d_{rs}^2 &= \frac{1}{n}\sum_{r=1}^{n} \vec{x}_r^T \vec{x}_r + \frac{1}{n}\sum_{r=1}^{n} \vec{x}_s^T \vec{x}_s - 2\frac{1}{n}\sum_{r=1}^{n} \vec{x}_r^T \vec{x}_s \\ &= \frac{1}{n}\sum_{r=1}^{n} \vec{x}_r^T \vec{x}_r + \frac{n}{n}\vec{x}_s^T \vec{x}_s - 2\vec{x}_s \frac{1}{n}\sum_{r=1}^{n} \vec{x}_r^T = \frac{1}{n}\sum_{r=1}^{n} \vec{x}_r^T \vec{x}_r + \vec{x}_s^T \vec{x}_s. \end{aligned} \tag{103}$$

In a similar manner, the mean for a r row can be computed as:

$$\frac{1}{n}\sum_{r=1}^{n} d_{rs}^2 = \frac{1}{n}\sum_{r=1}^{n} \vec{x}_r^T\vec{x}_r + \frac{1}{n}\sum_{r=1}^{n} \vec{x}_s^T\vec{x}_s - 2\frac{1}{n}\sum_{r=1}^{n} \vec{x}_r^T\vec{x}_s$$
$$= \frac{n}{n}\vec{x}_r^T\vec{x}_r + \frac{1}{n}\sum_{r=1}^{n} \vec{x}_s^T\vec{x}_s - 2\vec{x}_r^T\frac{1}{n}\sum_{r=1}^{n} \vec{x}_s = \vec{x}_r^T\vec{x}_r + \frac{1}{n}\sum_{r=1}^{n} \vec{x}_s^T\vec{x}_s. \quad (104)$$

Finally, the mean of all elements of D can be calculated as:

$$\frac{1}{n^2}\sum_{r=1}^{n}\sum_{s=1}^{n} d_{rs}^2 = \frac{1}{n^2}\sum_{r=1}^{n}\sum_{s=1}^{n} \vec{x}_r^T\vec{x}_r + \frac{1}{n^2}\sum_{r=1}^{n}\sum_{s=1}^{n} \vec{x}_s^T\vec{x}_s - 2\frac{1}{n^2}\sum_{r=1}^{n}\sum_{s=1}^{n} \vec{x}_r^T\vec{x}_s$$
$$= \frac{1}{n}\sum_{r=1}^{n} \vec{x}_r^T\vec{x}_r + \frac{1}{n}\sum_{s=1}^{n} \vec{x}_s^T\vec{x}_s = \frac{2}{n}\sum_{r=1}^{n} \vec{x}_r^T\vec{x}_r. \quad (105)$$

From equation (102), b_{rs} can be defined as:

$$b_{rs} = \vec{x}_r^T\vec{x}_s = -\frac{1}{2}\left(d_{rs}^2 - \vec{x}_r^T\vec{x}_r - \vec{x}_s^T\vec{x}_s\right). \quad (106)$$

However, from equation (103), term $-\vec{x}_r^T\vec{x}_r$ can be isolated as:

$$-\vec{x}_r^T\vec{x}_r = -\frac{1}{n}\sum_{s=1}^{n} d_{rs}^2 + \frac{1}{n}\sum_{s=1}^{n} \vec{x}_s^T\vec{x}_s, \quad (107)$$

and from equation (103), term $-\vec{x}_s^T\vec{x}_s$ can be isolated as:

$$-\vec{x}_s^T\vec{x}_s = -\frac{1}{n}\sum_{r=1}^{n} d_{rs}^2 + \frac{1}{n}\sum_{r=1}^{n} \vec{x}_r^T\vec{x}_r. \quad (108)$$

Adding equations (107) and (108) yields:

$$-\vec{x}_r^T\vec{x}_r - \vec{x}_s^T\vec{x}_s = -\frac{1}{n}\sum_{r=1}^{n} d_{rs}^2 - \frac{1}{n}\sum_{s=1}^{n} d_{rs}^2 + \frac{2}{n}\sum_{r=1}^{n} \vec{x}_r^T\vec{x}_r. \quad (109)$$

From equation (105), it is known that:

$$\frac{2}{n}\sum_{r=1}^{n} \vec{x}_r^T\vec{x}_r = \frac{1}{n^2}\sum_{r=1}^{n}\sum_{s=1}^{n} d_{rs}^2. \quad (110)$$

Finally, an arbitrary b_{rs} can be expressed as a function of the elements of the pairwise distances matrix D as:

$$b_{rs} = -\frac{1}{2}\left(d_{rs}^2 - \frac{1}{n}\sum_{r=1}^{n} d_{rs}^2 - \frac{1}{n}\sum_{s=1}^{n} d_{rs}^2 + \frac{1}{n^2}\sum_{r=1}^{n}\sum_{s=1}^{n} d_{rs}^2\right). \quad (111)$$

Taking $a_{rs} = -\frac{1}{2}d_{rs}^2$, it can be written:

$$a_{r.} = \frac{1}{n}\sum_{s=1}^{n} a_{rs}, \quad (112)$$

$$a_{.s} = \frac{1}{n}\sum_{r=1}^{n} a_{rs}, \quad (113)$$

$$a_{..} = \frac{1}{n}\sum_{r=1}^{n}\sum_{s=1}^{n} a_{rs}. \quad (114)$$

Thus, b_{rs} can be expressed as:

$$b_{rs} = a_{rs} - ar. - a_{.s} + a_{..}. \tag{115}$$

Defining matrix $A = \{a_{rs}\}$, for $r, s = 1, 2, \ldots, n$ as $A = \frac{1}{2}D$ and matrix H as:

$$H = \mathcal{I} - \frac{1}{n}\vec{1}\vec{1}^T, \tag{116}$$

where $\vec{1}^T = [1, 1, \ldots, 1]_n$, so that it becomes:

$$\vec{1}\vec{1}^T = \mathcal{U} = \begin{bmatrix} 1 & 1 & \ldots & 1 \\ 1 & 1 & \ldots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \ldots & 1 \end{bmatrix}_{n \times n}, \tag{117}$$

it is possible to compute all values of matrix B simultaneously using $B = HAH$. It must be noted that:

$$B = HAH = \left(\mathcal{I} - \frac{1}{n}\mathcal{U}\right) A \left(\mathcal{I} - \frac{1}{n}\mathcal{U}\right) - A - A\frac{\mathcal{U}}{n} - \frac{\mathcal{U}}{n}A + \frac{1}{n^2}\mathcal{U}A\mathcal{U}, \tag{118}$$

which is the matrix form of equation (115).

2.7.3 Retrieving coordinates of the points

At this point, the problem becomes finding the embedding, that is, the coordinates of points in $\mathbb{R}^d$. Initially, it is noted that the matrix of inner products B can be written as:

$$B_{n \times n} = X^T_{n \times m} X_{m \times n}, \tag{119}$$

where m and n denote, respectively, the number of samples and dimensionality, and $X_{n \times m} = [\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n]$ is the data matrix. In summary, it has been shown that matrix B has three important properties: it is symmetric, has rank m, and is positive semidefinite (COX; COX, 2001). Which means, matrix B has m non-negative eigenvalues and $n - m$ null eigenvalues. Therefore, by the spectral decomposition of B, comes:

$$B = V \Lambda V^T, \tag{120}$$

where $\Lambda = \operatorname{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$ is the diagonal matrix of eigenvalues of B and V is the matrix the columns of which are the eigenvalues of B:

$$V = \begin{bmatrix} | & | & \ldots & | \\ \vec{v}_1 & \vec{v}_2 & \ldots & \vec{v}_n \\ | & | & \ldots & | \end{bmatrix}_{n \times n}. \tag{121}$$

Without loss of generality, it can be assumed that $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$. Due to the $n - m$ null eigenvalues, matrix B can be expressed as:

$$B = \tilde{V}\tilde{\Lambda}\tilde{V}^T, \tag{122}$$

where $\tilde{\Lambda} = \text{diag}\,(\lambda_1, \lambda_2, \ldots, \lambda_m)$ is the diagonal matrix of non-null eigenvalues of B and $\tilde{V}$ is the $n \times m$ matrix the columns of which are the m eigenvectors associated to the m non-null eigenvalues:

$$\tilde{V} = \begin{bmatrix} | & | & \ldots & | \\ \vec{v}_1 & \vec{v}_2 & \ldots & \vec{v}_m \\ | & | & \ldots & | \end{bmatrix}_{n \times m} . \tag{123}$$

Hence, the following identity relates to matrix B:

$$B = X^T X = \tilde{V}\tilde{\Lambda}\tilde{V}^T = \tilde{V}\tilde{\Lambda}^{\frac{1}{2}}\tilde{\Lambda}^{\frac{1}{2}}\tilde{V}^T, \tag{124}$$

which finally leads to

$$X = \tilde{\Lambda}^{\frac{1}{2}}\tilde{V}^T, \tag{125}$$

where $\tilde{\Lambda}^{\frac{1}{2}} = \text{diag}\left(\sqrt{\lambda_1}, \sqrt{\lambda_2}, \ldots, \sqrt{\lambda_m}\right)$.

In practical terms, the intrinsic dimensionality, d, is chosen, an algorithm parameter that is lesser than the dimensionality of input data m, so that each column of X represents a sample in the manifold. In general, the $d < m$ eigenvalues associated to the d largest eigenvectors are chosen to form matrix $\tilde{V}$, thus, the algorithm returns a $d \times n$ data matrix that is the most compact representation of input data.

2.7.4 Relating Isometric Feature Mapping and Multidimensional Scaling

Defining $\vec{e}$ as:

$$\vec{e} = \frac{1}{\sqrt{n}}\left[1, 1, \ldots, 1\right]^T, \tag{126}$$

it can be noted that $1_n = \vec{e}\vec{e}^T$. Thus, replacing in equation (98), comes:

$$\begin{aligned} \tilde{K} = K - \vec{e}\vec{e}^T K - K\vec{e}\vec{e}^T + \vec{e}\vec{e}^T K \vec{e}\vec{e}^T &= \left[\left(\mathcal{I} - \vec{e}\vec{e}^T\right) K\right] - \left[\left(\mathcal{I} - \vec{e}\vec{e}^T\right) K\right] \vec{e}\vec{e}^T \\ &= \left(\mathcal{I} - \vec{e}\vec{e}^T\right) K \left(\mathcal{I} - \vec{e}\vec{e}^T\right). \end{aligned} \tag{127}$$

In ISOMAP, the first step consists in obtaining inner products matrix B from the geodesic distances matrix D, which equals:

$$B = -\frac{1}{2} HDH, \tag{128}$$

where $H = \left(\mathcal{I} - \vec{e}\vec{e}^T\right)$, which leads to (HAM et al., 2004):

$$K_{\text{iso}} = -\frac{1}{2}\left(\mathcal{I} - \vec{e}\vec{e}^T\right) D \left(\mathcal{I} - \vec{e}\vec{e}^T\right). \tag{129}$$

Therefore, KPCA becomes ISOMAP when kernel matrix K equals minus one-half of the geodesic distances matrix.

2.7.5 Enhanced Supervised Isometric Feature Mapping

Supervised DR techniques in visualisation and classification problems have been subject of many recent studies. In some ISOMAP-based algorithms, objects are represented by features arrays in an Euclidean space. Nonetheless, this representation requires selecting features, which is usually hard and domain dependent. Since the distance function does not need to be Euclidean, an alternative would be describing patterns using dissimilarity measures (RIBEIRO; VIEIRA; NEVES, 2008).

From the supposition that different data features can be captured using different dissimilarity measures, the Enhanced Supervised Isometric Feature Mapping (ES-ISOMAP) algorithm uses a dissimilarity matrix to uncover the manifold embedded in the data. The dissimilarity matrix $D(x_i, x_j)$ between two points x_i and x_j in the sample is defined by (RIBEIRO; VIEIRA; NEVES, 2008):

$$D(x_i, x_j) = \begin{cases} \sqrt{\frac{a-1}{a}} & \Longleftarrow \; c_i = c_j \\ \sqrt{a} - d_0 & \Longleftarrow \; c_i \neq c_j \end{cases}, \tag{130}$$

where $a = \exp\left(\frac{d_{ij}^2}{\sigma}\right)$ with d_{ij} defined as a distance measure (Euclidean, Co-sin, Correlation, Spearman, Kendal-τ), σ is a smoothing parameter (defined according to the "density" of data), d_0 is a constant ($0 \leq d_0 \leq 1$) and c_i, c_j are the class labels. If the dissimilarity between two samples is less than 1, the points are in the same class, and in different classes otherwise. Interclass dissimilarity is larger than intraclass, granting the method great discriminative ability.

2.8 Locally Linear Embedding

The ISOMAP algorithm is a global method, considering that, in order to find the coordinates of an input vector $\vec{x}_i \in \mathbb{R}^m$ in the manifold, it uses informations from all samples in matrix B. Comparatively, Locally Linear Embedding (LLE) is a local method, that is, the new coordinates of any $\vec{x}_i \in \mathbb{R}^m$ depend only on this point's neighbourhood. The main hypothesis behind LLE is that, for a sufficiently high sample density, it is expected that a vector $\vec{x}_i$ and its neighbours define a linear patch, that is, they belong to a single Euclidean subspace (ROWEIS; SAUL, 2000). Hence, the local geometry can be classified through linear coefficients:

$$\hat{\vec{x}}_i \approx \sum_j w_{ij} \vec{x}_j \text{ for } \vec{x}_j \in N(\vec{x}_i), \tag{131}$$

that is, a vector can be reconstructed as a linear combination of its neighbours.

Basically, the LLE algorithm requires a $n \times m$ data matrix X as input, with rows $\vec{x}_i$, a number of desired dimensions $d < m$, and an integer $k > d+1$ to find local neighbourhoods.

The output is a $n \times d$ matrix Y, with rows $\vec{y}_i$. The LLE algorithm can be divided in three main steps (ROWEIS; SAUL, 2000; SAUL; ROWEIS, 2003):

1. Starting from each $\vec{x}_i \in \mathbb{R}^m$, find its KNN;
2. Compute the weights matrix W that minimises reconstruction error for each data point $\vec{x}_i \in \mathbb{R}^m$:
$$E(W) = \sum_{i=1}^{n} \left\| \vec{x}_1 - \sum_j w_{ij} \vec{x}_j \right\|^2, \tag{132}$$
where $w_{ij} = 0$, unless $\vec{x}_j$ is one of $\vec{x}_i$'s KNN and, for each i, $\sum_j w_{ij} = 1$;
3. Find the Y coordinates that minimise reconstruction error using optimal weights:
$$\Phi(Y) = \sum_{i=1}^{n} \left\| \vec{y}_i - \sum_j w_{ij} \vec{y}_j \right\|^2 \tag{133}$$
subject to constraints: $\sum_i Y_{ij} = 0$ and $Y^T Y = \mathcal{I}$.

How to obtain the solutions for the steps in LLE is described next.

2.8.1 Finding locally linear neighbourhoods

The basic version of LLE uses a fixed number of neighbours for each sample and adopts simple Euclidean distance as a metric to classify the nearest ones. However, different criteria can be considered in choosing the nearest neighbours, such as selecting samples within a fixed radius sphere. The number of neighbours can also be different for each neighbourhood. As alternative rules, one can either:

- ❏ Select all samples within a radius R_i, up to a maximum of N_i;
- ❏ Choose a number of neighbours N_i, with none outside of a maximum radius R_i (SAUL; ROWEIS, 2003).

A relevant aspect of LLE is that the algorithm can retrieve embeddings in which the intrinsic dimensionality d is smaller than the number of neighbours k. Furthermore, assuming a linear patch forces an upper bound to k. For instance, in very curvy data sets, a large k is not reasonable, as it violates this condition. In the uncommon case that $k > m$, it has been shown that each sample can be perfectly reconstructed from its neighbours, and another problem arises: reconstructions weights are no longer unique. In order to overcome this limitation, a regularisation is needed to eliminate degeneration (SAUL; ROWEIS, 2003).

Finally, another consideration about the LLE algorithm is the connectedness of the KNN graph. If it has multiple connected components, LLE can be applied separately on each, or selection must be altered to guarantee global connectedness (SAUL; ROWEIS, 2003).

2.8.2 Estimating Minimum Square Weights

The second step in LLE is reconstruction of each data point from its nearest neighbours. The optimal reconstruction weights can be calculated in a closed form. Without loss of generality, the total reconstruction error in a point $\vec{x}_i$ can be expressed as:

$$E\left(\vec{w}\right) = \left\| \sum_j w_j \left(\vec{x}_i - \vec{x}_j\right) \right\|^2 = \sum_j \sum_k w_j w_k \left(\vec{x}_i - \vec{x}_j\right)^T \left(\vec{x}_i - \vec{x}_k\right). \tag{134}$$

Defining matrix C as:

$$C_{jk} = \left(\vec{x}_i - \vec{x}_j\right)^T \left(\vec{x}_i - \vec{x}_k\right), \tag{135}$$

results in the following expression for local reconstruction error:

$$E\left(\vec{w}\right) = \sum_j \sum_k w_j C_{jk} w_k = \vec{w}^T C \vec{w}. \tag{136}$$

Constraint $\sum_j w_j = 1$ can be understood in two different ways: geometrically and probabilistically. From the geometric point of view, it ensures invariance under translation, that is, adding any vector $\vec{c}$ to $\vec{x}_i$ and all its neighbours does not alter the reconstruction error. Let $\tilde{\vec{x}}_i = \vec{x}_i + \vec{c}$ and $\tilde{\vec{x}}_j = \vec{x}_j + \vec{c}$, the new local reconstruction error is given by:

$$\begin{aligned} \tilde{E}\left(\vec{w}\right) &= \left\| \tilde{\vec{x}}_i - \sum_j w_j \tilde{\vec{x}}_j \right\|^2 = \left\| \vec{x}_i + \vec{c} - \sum_j w_j \left(\vec{x}_j + \vec{c}\right) \right\|^2 \\ &= \left\| \vec{x}_i + \vec{c} - \sum_j w_j \vec{x}_j + - \sum_j w_j \vec{c} \right\|^2 = \left\| \vec{x}_i + \vec{c} - \sum_j w_j \vec{x}_j - \vec{c} \right\|^2 = E\left(\vec{w}\right). \end{aligned} \tag{137}$$

In terms of probability, forcing weights to sum up zero makes W a stochastic transition matrix (SAUL; ROWEIS, 2003), directly related to Markov Chains and diffusion maps. It can be demonstrated that, in the minimisation of square error, solution is found by an eigenvalues problem. Effectively, the estimation of W narrows down to n eigenvalues problems: as there are no constraints over the lines of W, optimal weights for each sample $\vec{x}_i$ can be found separately, which drastically simplifies calculations.

Then, there are n independent constrained optimisation problems given by:

$$\arg\min_{\vec{w}_i} \vec{w}_i^T C_i \vec{w}_i \text{ subject to } \vec{1}^T \vec{w}_i = 1 \text{ for } i = 1, 2, \ldots, n. \tag{138}$$

Using Lagrange multipliers, the Lagrangian is written as:

$$L\left(\vec{w}_i, \lambda\right) = \vec{w}_i^T C_i \vec{w}_i - \lambda \left(\vec{1}^T \vec{w}_i - 1\right). \tag{139}$$

Differentiating with respect to $\vec{w}_i$:

$$\frac{\partial}{\partial \vec{w}_i} L\left(\vec{w}_i, \lambda\right) = 2 C_i \vec{w}_i - \lambda \vec{1} = 0, \tag{140}$$

which leads to:

$$C_i \vec{w}_i = \frac{\lambda}{2}\vec{1}. \tag{141}$$

If matrix C_i is reversible, the closed form solution is:

$$\vec{w}_i = \frac{\lambda}{2} C_i^{-1}\vec{1}, \tag{142}$$

where λ can be set to guarantee $\sum_j w_i(j) = 1$. Indeed, there is a closed form expression of $w_i(j)$ (SAUL; ROWEIS, 2000):

$$w_i(j) = \frac{\sum_k C_i^{-1}(j,k)}{\sum_k \sum_l C_i^{-1}(k,l)}. \tag{143}$$

To speed up the algorithm, instead of computing the inverse matrix of C, it is common to solve the linear system:

$$C_i \vec{w}_i = \vec{1}, \tag{144}$$

and then normalise the solution to ensure $\sum_j w_i(j) = 1$ by dividing each coefficient of vector $\vec{w}_i$ by the sum of all coefficients:

$$w_i(j) = \frac{w_i(j)}{\sum_j w_i(j)} \text{ for } j = 1, 2, \ldots, m. \tag{145}$$

If the number of neighbours k is greater than that of features m, then (usually) the space occupied by k distinct vectors is the whole space. Therefore, $\vec{x}_i$ can be written exactly as a linear combination of its KNN. In fact, if $k > m$, there are generally infinite solutions to $\vec{x}_i = \sum_j w_j \vec{x}_j$, for there are more unknown variables (k) than equations (m). In this case, the optimisation problem is badly defined and needs regularisation. A common technique for this is Tikhonov regularisation (TIKHONOV; ARSENIN, 1977) which, instead of minimising:

$$\left\| \vec{x}_i - \sum_j w_j \vec{x}_j \right\|^2, \tag{146}$$

adds a penalty term to the least squares problem:

$$\left\| \vec{x}_i - \sum_j w_j \vec{x}_j \right\|^2 + \alpha \sum_j w_j^2, \tag{147}$$

where α controls the degree of regularisation. That is, the weights that minimise a combination of reconstruction errors and the sum of weights squared are chosen. If $\alpha \to 0$, there is a least squares problem; if $\alpha \to \infty$ the square error terms becomes negligible, and one wishes to minimise the Euclidean norm of weight vector $\vec{w}$. Typically, α is chosen as a small, but not null, value. Then, the n constrained optimisation problems become:

$$\arg\min_{\vec{w}_i} \vec{w}_i^T C_i \vec{w}_i + \alpha \vec{w}_i^T \vec{w}_i \text{ subject to } \vec{1}^T \vec{w}_i = 1 \text{ for } i = 1, 2, \ldots, n. \tag{148}$$

The Lagrangian function is defined by:

$$L\left(\vec{w}_i, \lambda\right) = \vec{w}_i^T C_i \vec{w}_i + \alpha \vec{w}_i^T \vec{w}_i - \lambda\left(\vec{1}^T \vec{w}_i - 1\right). \tag{149}$$

Differentiating with respect to $\vec{w}_i$ and equating to zero:

$$2C_i\vec{w}_i + 2\alpha\vec{w}_i = \lambda\vec{1}, \tag{150}$$

$$\left(C_i + \alpha\mathcal{I}\right)\vec{w}_i = \frac{\lambda}{2}\vec{1}, \tag{151}$$

$$\vec{w}_i = \frac{\lambda}{2}\left(C_i + \alpha\mathcal{I}\right)^{-1}\vec{1}, \tag{152}$$

where λ is chosen as to normalise $\vec{w}_i$, that is, to regularise the problem, a small disturbance is added to the main diagonal of matrix C_i.

2.8.3 Finding the coordinates

If local neighbourhoods are sufficiently small in comparison to the curvature of the manifold, optimal reconstruction weights in the embedding space and those of the manifold are approximately equal (both sets of weights are exactly the same for linear subspaces and, for manifolds in general, can be arbitrarily approximated by sufficiently reducing the size of the neighbourhood) (SHALIZI, 2009). The idea behind the third step in the LLE algorithm is using optimal weights estimated by least squares in the manifold and solving for local coordinates. Thus, fixating matrix W, the goal is to solve another quadratic minimisation problem in order to minimise:

$$\Phi\left(Y\right) = \sum_{i=1}^{n} \left\| \vec{y}_i - \sum_j w_{ij}\vec{y}_j \right\|^2. \tag{153}$$

That is, coordinates $\vec{y}_i \in \mathbb{R}^d$ (approximately in the manifold) reconstructed by these weights (W) must be found.

To avoid degeneration, two constraints are imposed:

1. The mean of transformed spaces is zero, otherwise there would be infinite solutions;
2. The covariance matrix of transformed data is the identity matrix, that is, there is no correlation between components of $\vec{y} \in \mathbb{R}^d$.

However, unlike the estimation of W, finding coordinates cannot be simplified to n independent problems, for each row in Y appears in Φ many times, as the central vector of $\vec{y}_i$ as well as neighbour to other vectors.

Initially, equation (153) is rewritten using matrices. It is noted that:

$$\Phi\left(Y\right) = \sum_{i=1}^{n} \left[\left(\vec{y}_i - \sum_j w_{ij}\vec{y}_j \right)^T \left(\vec{y}_i - \sum_j w_{ij}\vec{y}_j \right) \right]. \tag{154}$$

Applying the distributive property:

$$\Phi(Y) = \sum_{i=1}^{n} \left[\vec{y}_i^T \vec{y}_i - \vec{y}_i^T \left(\sum_j w_{ij} \vec{y}_j \right) - \left(\sum_j w_{ij} \vec{y}_j \right)^T \vec{y}_i + \left(\sum_j w_{ij} \vec{y}_j \right)^T \left(\sum_j w_{ij} \vec{y}_j \right) \right]. \tag{155}$$

Expanding the summation:

$$\Phi(Y) = \sum_{i=1}^{n} \vec{y}_i^T \vec{y}_i - \sum_{i=1}^{n} \sum_j \vec{y}_i^T w_{ij} \vec{y}_j - \sum_{i=1}^{n} \sum_j \vec{y}_j^T w_{ji} \vec{y}_i + \sum_{i=1}^{n} \sum_j \sum_k \vec{y}_j^T w_{ji} w_{ik} \vec{y}_k. \tag{156}$$

Denoting by Y the $d \times n$ matrix in which each column $\vec{y}_i$ for $i = 1, 2, \ldots, n$ stores the coordinates of the i-th point in the manifold and knowing that $\vec{w}_i(j) = 0$ unless $\vec{y}_j$ neighbours $\vec{y}_i$, $\Phi(Y)$ can be written as:

$$\begin{aligned}
\Phi(Y) &= \mathrm{Tr}\left(Y^T Y\right) - \mathrm{Tr}\left(Y^T W Y\right) - \mathrm{Tr}\left(Y^T W^T Y\right) + \mathrm{Tr}\left(Y^T W^T W Y\right) \\
&= \mathrm{Tr}\left(Y^T Y\right) - \mathrm{Tr}\left(Y^T (WY)\right) - \mathrm{Tr}\left((WY)^T Y\right) + \mathrm{Tr}\left((WY)^T (WY)\right) \\
&= \mathrm{Tr}\left(Y^T (Y - WY) - (WY)^T (Y - WY)\right) = \mathrm{Tr}\left((Y - WY)^T (Y - WY)\right) \\
&= \mathrm{Tr}\left(((\mathcal{I} - W) Y)^T ((\mathcal{I} - W) Y)\right) = \mathrm{Tr}\left(Y^T (\mathcal{I} - W)^T (\mathcal{I} - W) Y\right).
\end{aligned} \tag{157}$$

Defining $n \times n$ matrix M as:

$$M = (\mathcal{I} - W)^T (\mathcal{I} - W) \tag{158}$$

leads to the optimisation problem:

$$\arg\min_Y \ \mathrm{Tr}\left(Y^T M Y\right) \ \text{ subject to } \frac{1}{n} Y^T Y - \mathcal{I}. \tag{159}$$

Hence, the Lagrangian function is given by:

$$L(Y, \lambda) = \mathrm{Tr}\left(Y^T M Y\right) - \lambda \left(\frac{1}{n} Y^T Y - \mathcal{I} \right). \tag{160}$$

Differentiating and equating to zero:

$$2MY - 2\frac{\lambda}{n} Y = 0, \tag{161}$$

$$MY = \beta Y, \tag{162}$$

where $\beta = \frac{\lambda}{n}$, showing that Y must be made up of the eigenvector of matrix M. From a minimisation problem, the d eigenvectors associated to the least d eigenvalues must be chosen to build Y. As a $n \times n$ matrix, M has n eigenvalues and n orthogonal eigenvectors. Although these eigenvalues are real and non-negative, the least is always zero, with constant eigenvector $\vec{1}$. This eigenvector corresponds to the mean of Y and must be discarded to fulfil constraint $\sum_{i=1}^{n} \vec{y}_i = 0$ (RIDDER; DUIN, 2002). Each row of W must sum up to one,

thence:

$$W\vec{1} = \vec{1}, \tag{163}$$

$$\vec{1} - W\vec{1} = 0, \tag{164}$$

$$(\mathcal{I} - W)\,\vec{1} = 0, \tag{165}$$

$$(\mathcal{I} - W)^T\,(\mathcal{I} - W)\,\vec{1} = 0, \tag{166}$$

$$M\vec{1} = 0. \tag{167}$$

Therefore, to obtain $\vec{y}_i \in \mathbb{R}^d$, where $d < n$, the $d+1$ least eigenvectors must be selected and constant eigenvector with eigenvalue zero discarded, that is, choosing the d eigenvectors associated with the least non-zero eigenvalues.

The LLE algorithm has three main parameters: intrinsic dimensionality d, number of close neighbours k and, in some cases, regularisation parameter α, being DR very sensitive to these. If d is too high, mapping amplifies noise; if it is too low, distinct parts of the data set can be mapped overlapped. If k is too small, mapping does not reflect global properties; if too large, mapping loses its non-linear character and behaves like traditional PCA, considering the whole set as a local neighbourhood. Finally, if α is incorrect, spectral analysis may not converge (RIDDER; DUIN, 2002).

2.8.4 Supervised Locally Linear Embedding

A ML algorithm must not necessarily be unsupervised. Oft, better and more efficient results can be achieved by having a previous classification of the data. A variation of LLE, called Supervised Locally Linear Embedding (SLLE), regards data classes before applying LLE to work with data sets containing multiple manifolds, usually disjoined. To do so, the algorithm is altered to consider the difference between data sets when calculating the distances matrix. So, the values of matrix elements are given by (RIDDER; KOUROPTEVA, et al., 2003):

$$\Delta' = \Delta + \alpha \max(\Delta)\,\Lambda_{ij}, \alpha \in [0, 1]\,, \tag{168}$$

where $\max(\Delta)$ is the maximum value of Δ and $\Lambda_{ij} = 0$ if x_i and x_j belong to he same class, and 0 otherwise. When $\alpha = 0$ the result is unsupervised LLE, while $\alpha = 1$ results in fully supervised LLE (named 1-SLLE) and if $0 < \alpha < 1$, it is a partially supervised LLE (α-LLE).

For 1-SLLE, distances between samples from different classes are as large as the maximum distance in all of the data set. Thus, neighbours from a c class sample are always chosen among points in the same class. In practice, it is not necessary to compute equation (168), just to pick the nearest neighbours of a sample among those in the same class. 1-SLLE is, therefore, a non-parametrised supervised LLE. However, α-SLLE introduces an additional α parameter that controls the amount of supervision. For $0 < \alpha < 1$, a

configuration such that preserves part of the manifold structure but separates classes is found. This allows a supervised analysis of data, but may lead to a better generalisation than 1-SLLE in previously not analysed samples (RIDDER; KOUROPTEVA, et al., 2003).

2.9 Laplacian Eigenmaps

The idea behind the Laplacian Eigenmaps (LE) method is that, if a manifold is approximated by a basic undirected connected graph, it is possible to find a map of its vertices to an Euclidean subspace $\mathbb{R}^d$ such that locality is preserved, that is, neighbouring points in the graph remain close after mapping. This map is given by eigenvectors of the graph's Laplacian matrix (LI; LI; ZHANG, 2018). Basically, the LE algorithm takes as input a $n \times m$ data matrix X, with each row $\vec{x}_i$ defining a data point, a number of desired dimensions $d < m$, and an integer k, to find local neighbourhoods. The output is a $n \times d$ matrix Y, with rows $\vec{y}_i$. The algorithm can be divided into three main steps (BELKIN; NIYOGI, 2003):

1. Construct neighbourhood graph $G = (V, E)$ connecting nodes v_i and v_j if $\vec{x}_i$ and $\vec{x}_j$ are close. Its two variants are:
   - ❏ ϵ-neighbourhood: connect v_i and v_j with an edge if $\|\vec{x}_i - \vec{x}_j\|^2 \leq \epsilon$.
   - ❏ KNN: connect v_i and v_j with an edge if v_i is among the KNN of v_j or if v_j is among the KNN of v_i.
2. Choose weights to define adjacency matrix W. Here, too, there are two variants:
   - ❏ Heat kernel (with parameter $t \in \mathbb{R}$): if nodes v_i and v_j are connected,
   $$W_{ij} = \exp\left(-\frac{\|\vec{x}_i - \vec{x}_j\|^2}{t}\right), \tag{169}$$
   otherwise $W_{ij} = 0$.
   - ❏ Binary weights: $W_{ij} = 1$ if nodes v_i and v_j are connected by an edge and $W_{ij} = 0$ otherwise.
3. Embedding: find coordinates Y selecting the d eigenvectors associated to the d least non-zero eigenvalues of the graph's Laplacian L.

Next, the reasons for which LE can provide optimal embeddings in terms of locality and neighbourhood preservation.

2.9.1 Graph's Laplacian and its Properties

Let $G = (V, E)$ be an undirected graph with a set of vertices $V = \{v_1, v_2, \ldots, v_n\}$, and let it be weighed, that is, each edge between v_i and v_j has a non-negative weight $w_{ij} \geq 0$.

Typically, weights w_{ij} represent a measure of similarity or distance between pairs of vectors $\vec{x}_i \in \mathbb{R}^m$ and $\vec{x}_j \in \mathbb{R}^m$.

Definition 2. The weighed adjacency matrix of an undirected graph $G = (V, E)$ with $|V| = n$ is the symmetrical matrix $W = w_{ij}$ for $i, j = 1, 2, \ldots, n$. If $w_{ij} = 0$, vertices v_i and v_j are not connected by an edge.

Definition 3. The degree of a vertex $v_i \in V$ is defined by the sum of elements in the i-th row of W:

$$d_i = \sum_{j=1}^{n} w_{ij}. \tag{170}$$

Degrees matrix D is defined as a diagonal matrix with degrees $d_1, d_2, \ldots, d_n$.

Definition 4. The non-normalised graph's Laplacian matrix is defined by:

$$L = D - W, \tag{171}$$

where D is the degrees matrix and W the adjacency matrix.

Theorem 3. Laplacian matrix L satisfies the following properties:

1. For each column vector $\vec{f} \in \mathbb{R}^n$:

$$\vec{f}^T L \vec{f} = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} \left(f_i - f_j\right)^2. \tag{172}$$

2. L is symmetric and positive semidefinite.

3. The least eigenvalue of L is zero and the corresponding eigenvector is the constant vector $\vec{1}$.

4. L has n real non-zero eigenvalues, $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_m$.

To prove the first statement, one notes that, from the definition of L and D:

$$\begin{aligned}
\vec{f}^T L \vec{f} = \vec{f}^T D \vec{f} - \vec{f}^T W \vec{f} = \sum_{i=1}^{n} d_i f_i^2 - \sum_{i=1}^{n} \sum_{j=1}^{n} f_i w_{ij} f_j = \frac{1}{2} \left(2 \sum_{i=1}^{n} d_i f_i^2 - 2 \sum_{i=1}^{n} \sum_{j=1}^{n} f_i w_{ij} f_j \right) \\
= \frac{1}{2} \left(\sum_{i=1}^{n} d_i f_i^2 - 2 \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} f_i f_j + \sum_{j=1}^{n} d_j f_j^2 \right) \\
= \frac{1}{2} \left(\sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} f_i^2 - 2 \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} f_i f_j + \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} f_j^2 \right) = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} \left(f_i - f_j\right)^2.
\end{aligned} \tag{173}$$

The second statement has two parts: the first, about symmetry, is a direct consequence of the symmetry between matrices D and W; the second, about being positive semidefinite,

is deducted from $(f_i - f_j)^2 \geq 0, \forall f_i, f_j \in \mathbb{R}$ and $w_{ij} \geq 0$ for $i, j = 1, 2, \ldots, n$, $\vec{f}^T L \vec{f} \geq 0$. To prove the third, one notes:

$$L\vec{1} = (D - W)\,\vec{1} = D\vec{1} - W\vec{1} = \sum_{i=1}^{n} d_i - \sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij} = \sum_{i=1}^{n} d_i - \sum_{i=1}^{n} d_i = 0, \tag{174}$$

showing that constant eigenvector $\vec{1}$ has eigenvalue zero. Finally, the fourth statement is a direct consequence of equations (170) and (171).

2.9.2 Inline Laplacian Embedding

Here, it shall be proven that the embedding provided by LE is optimal in terms of preservations of local information, that is, neighbouring points in the graph are close and distant ones are apart after the embedding. Assuming a connected graph $G = (V, E)$ with nodes which are data points $X = [\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n]$, the problem can be formulated as: mapping nodes in G as a line to keep points as closely connected as possible, which is the goal of LE.

Let $\vec{y} = [y_1, y_2, \ldots, y_n]^T \in \mathbb{R}^n$ be a map of vertices $v_1, v_2, \ldots, v_n$ to a real line, a good goal function must strongly penalise neighbouring points that are mapped too far apart from each other. An adequate choice for a given adjacency matrix W is the function:

$$J(\vec{y}) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}\,(y_i - y_j)^2 = \vec{y}^T L \vec{y}, \tag{175}$$

where L is the Laplacian matrix. It can be noted that $J(\vec{y})$ is the measure of scattering of the points among the real line, therefore minimising it is an attempt to ensure that, if $\vec{x}_i$ and $\vec{x}_j$ are close in the input space, coordinates y_i and y_j are also close in line. Thus, the constrained optimisation problem can be written as:

$$\arg\min_{\vec{y}} \;\; y^T L y \text{ subject to } y^T D y = 1, \tag{176}$$

where constraint $y^T D S y = 1$ removes the arbitrary embedding scale factor (BELKIN; NIYOGI, 2003). That is, the direction of vector $\vec{y}$ is sought. If there were no constraints, the objective function could be minimised by simply dividing the components of $\vec{y}$ by a constant. Again, rewriting the Lagrangian function:

$$L(\vec{y}, \lambda) = y^T L y - \lambda\left(y^T D y - 1\right). \tag{177}$$

Differentiating with respect to $\vec{y}$ and equating the result to zero:

$$\frac{\partial}{\partial \vec{y}} L(\vec{y}, \lambda) = 2L\vec{y} - 2\lambda D\vec{y} = 0, \tag{178}$$

which leads to:

$$L\vec{y} = \lambda D\vec{y}, \tag{179}$$

$$\left(D^{-1}L\right)\vec{y} = \lambda\vec{y}, \tag{180}$$

demonstrating that this is a generalised eigenvectors problem. As a minimisation problem, the eigenvector of $D^{-1}L$ associated to the least eigenvalue must be chosen, discarding constant eigenvector $\vec{1}$, which has eigenvalue zero. Obviously, minimal scattering occurs when all points are mapped to the same coordinate, however this trivial solution has no practical use. Hence, $\vec{y}$ must be the eigenvector associated to the least non-zero eigenvalue, also called Fiedler vector (FIEDLER, 1989).

2.9.3 Laplacian Embedding in $\mathbb{R}^d$

Considering the generalised embedding problem for graph $G = (V, E)$ in a d-dimensional Euclidean space, each node $v_i \in V$ must be mapped to a point in $\mathbb{R}^d$, that is, d coordinates are to be estimated for each node. The final embedding is noted by a $n \times d$ matrix $Y = [\vec{y}_1, \vec{y}_2, \ldots, \vec{y}_d]$, where the i-th row, $\vec{y}^{(i)}$, provides the coordinates of v_i in the manifold. The objective function is generalised to:

$$J(Y) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} W_{ij}\left\|\vec{y}^{(i)} - \vec{y}^{(j)}\right\|^2, \tag{181}$$

where $\vec{y}^{(i)} = \left[\vec{y}_1^{(i)}, \vec{y}_2^{(i)}, \ldots, \vec{y}_d^{(i)}\right]$ is the d-dimensional representation of v_i. Let Y be a $n \times d$ matrix in which each row represents a $\vec{y}^{(i)}$, for $i = 1, 2, \ldots, n$, the objective function can be rewritten as:

$$J(Y) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} W_{ij}\left(\vec{y}^{(i)} - \vec{y}^{(j)}\right)\left(\vec{y}^{(i)} - \vec{y}^{(j)}\right)^T. \tag{182}$$

Expanding the expression of $J(Y)$, this can be simplified to:

$$\begin{aligned} J(Y) &= \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\left[W_{ij}\vec{y}^{(i)}\vec{y}^{(i)T} - W_{ij}\vec{y}^{(i)}\vec{y}^{(j)T} - W_{ij}\vec{y}^{(j)}\vec{y}^{(i)T} + W_{ij}\vec{y}^{(j)}\vec{y}^{(j)T}\right] \\ &= \frac{1}{2}\left[\sum_{i=1}^{n} d_i\vec{y}^{(i)}\vec{y}^{(i)T} - 2\sum_{i=1}^{n}\sum_{j=1}^{n} W_{ij}\vec{y}^{(i)}\vec{y}^{(j)T} + \sum_{j=1}^{n} d_j\vec{y}^{(j)}\vec{y}^{(j)T}\right] \\ = \frac{1}{2}\left[2\sum_{i=1}^{n} d_i\vec{y}^{(i)}\vec{y}^{(i)T} - 2\sum_{i=1}^{n}\sum_{j=1}^{n} W_{ij}\vec{y}^{(i)}\vec{y}^{(j)T}\right] &= \sum_{i=1}^{n} d_i\vec{y}^{(i)}\vec{y}^{(i)T} - \sum_{i=1}^{n}\sum_{j=1}^{n} W_{ij}\vec{y}^{(i)}\vec{y}^{(j)T}. \end{aligned} \tag{183}$$

Let $Y_{n\times d}$ be the matrix of coordinates for n points, $D_{n\times n}$ the diagonal matrix of degrees d_i, and $W_{n\times n}$ the adjacency matrix, the equation can be written using matrix-vector notation as:

$$J(Y) = \mathrm{Tr}\left(DYY^T\right) - \mathrm{Tr}\left(WYY^T\right). \tag{184}$$

Being the trace operator invariant to cyclic permutations, comes:

$$\begin{aligned} J(Y) = \mathrm{Tr}\left(Y^TDY\right) - \mathrm{Tr}\left(Y^TWY\right) &= \mathrm{Tr}\left(Y^T\left(DY - WY\right)\right) \\ &= \mathrm{Tr}\left(Y^T\left(D - W\right)Y\right) = \mathrm{Tr}\left(Y^TLY\right). \end{aligned} \tag{185}$$

Thus, leads to the constrained optimisation problem:

$$\arg\min_{Y} \ \mathrm{Tr}\left(Y^T L Y\right) \ \text{subject to } Y^T D Y = \mathcal{I}, \tag{186}$$

and its Lagrangian function is given by:

$$L\left(Y, \lambda\right) = \mathrm{Tr}\left(Y^T L Y\right) - \lambda\left(Y^T D Y - \mathcal{I}\right). \tag{187}$$

Differentiating and equating to zero:

$$\frac{\partial}{\partial Y} L\left(Y, \lambda\right) = 2LY - 2\lambda DY = 0, \tag{188}$$

which leads to the eigenvectors problem:

$$LY = \lambda DY. \tag{189}$$

This result shows that to compose the columns of matrix Y, one must select the d eigenvectors associated to the d least non-zero eigenvalues of normalised Laplacian $D^{-1}L$. Some variants of the algorithm include the spectral decomposition of different versions of the graph Laplacian. Most common choices are another form of the normalised Laplacian, given by $L_{\text{sym}} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}}$, and pure non-normalised Laplacian $L = D - W$. By applying LE to real datasets, limitations were found, such as uneven sampling of data, out of sample problems, small sample size, extraction and selection of discriminant features, etc. To overcome these issues, a large number of extensions was proposed for LE (LI; LI; ZHANG, 2018).

The LE algorithm is rather simple, consisting of few local operations and a sparse eigenvalues problem. Its solution reflects the intrinsic geometry of the manifold due to the Laplacian operator's role in providing an optimal embedding. The solution graph Laplacian obtained from data points can be viewed as an approximation of the Laplace-Beltrami operator defined in the manifold (BELKIN; NIYOGI, 2002). The algorithm is relatively insensitive to outliers and noise and, thus, emphasises data groupings.

Given n points $x_1, \ldots, x_n$ in $\mathbb{R}^D$, a weighed graph is constructed with n nodes, one for each point, and the set of edges connecting neighbouring points. There is an edge between nodes i and j if x_i and x_j are near, based on a maximum distance ϵ between two points or a number k of nearest points. There are also two possibilities for edge weights: based on heat core (equation (169)), or a simpler one with $W_{ij} = 1$ (BELKIN; NIYOGI, 2002). Assuming the resultant G graph is connected, eigenvectors and eigenvalues are calculated for the generalised eigenvectors problem:

$$Ly = \lambda Dy \tag{190}$$

where D is the diagonal of the weights matrix, with its inputs being the sums of the columns (or rows, as W is symmetric) of W, $D_{ii} = \sum_j W_{ji}$. $L = D - W$ is the Laplacian

matrix, which is a symmetric, positive and semidefinite matrix that can be considered an operator of functions defined in the edges of G. Thus, $y_0, \ldots, y_{n-1}$ are solutions of equation (190), ordered by their eigenvalues with y_0 having the least eigenvalue (actually 0). The image of x_i in low dimensional space $\mathbb{R}^d$ is given by $\left(y_1^{(i)}, \ldots, y_d^{(i)}\right)$.

2.9.4 Supervised Laplacian Eigenmaps

Although LE might yield good results for non-linear DR, many experiments show that the recognition rate in the reduced space depends on the size of the chosen neighbourhood. A way to use the information provided by data classification, not used by LE, is through Supervised Laplacian Eigenmaps (S-LE). This method has two interesting properties: it estimates adaptively local neighbourhoods of each sample based on data density and similarity, and its objective function simultaneously maximises local margins between heterogeneous samples and approximates homogeneous samples. Hence, it avoids the need to choose a predefined neighbourhood for graph construction and exploits information discrimination to find a non-linear embedding (RADUCANU; DORNAIKA, 2012).

To uncover the geometric and discriminant structure of the data manifold, the global graph is divided in two components: intraclass G_w and interclass G_b. Being $l(y_i)$ the label of y_i class, for each y_i data point two sets are computed: $N_w(y_i)$, containing neighbours with the same label as y_i, and $N_b(y_i)$, containing different labelled neighbours. Unlike classic LE, this algorithm adapts the size of these two sets according to local sample point y_i and its similarities to other samples. To do so, each set is defined for each point y_i and calculated in two steps. First, mean similarity is computed from the total of all similarities to the rest of the set:

$$AS(y_i) = \frac{1}{N} \sum_{k=1}^{N} \exp\left(-\frac{\|y_i - y_k\|^2}{\beta}\right). \tag{191}$$

Then, sets are calculated by:

$$N_w(y_i) = \left\{ y_j \mid l(y_j) = l(y_i), \exp\left(-\frac{\|y_i - y_j\|^2}{\beta}\right) > AS(y_i) \right\}, \tag{192}$$

made up of data samples with the same label as y_i and larger than mean similarity associated to y_i and (RADUCANU; DORNAIKA, 2012):

$$N_b(y_i) = \left\{ y_j \mid l(y_j) \neq l(y_i), \exp\left(-\frac{\|y_i - y_j\|^2}{\beta}\right) > AS(y_i) \right\}, \tag{193}$$

made up of data samples with labels different from that of y_i and larger than mean similarity associated to y_i. These equations show that these sets are not equally sized for all data samples, adapting their neighbourhood according to local density and similarity between samples in the original space.

Since graphs G_w and G_b have weight matrices W_w and W_b, respectively (RADUCANU; DORNAIKA, 2012):

$$W_{w,ij} = \begin{cases} \exp\left(-\frac{\|y_i - y_j\|^2}{\beta}\right), & \text{if } y_j \in N_w(y_i) \text{ or } y_i \in N_w(y_j) \\ 0, & \text{otherwise} \end{cases}, \tag{194}$$

$$W_{b,ij} = \begin{cases} 1, & \text{if } y_j \in N_b(y_i) \text{ or } y_i \in N_b(y_j) \\ 0, & \text{otherwise} \end{cases}. \tag{195}$$

If the same weight is used for both, it is simple to demonstrate that global affinity matrix W associated to LE is:

$$W = W_w + W_b. \tag{196}$$

Each data point y_i is mapped to a vector x_i, in order to retrieve the coordinates of x_i in low dimension space for each sample, from objective functions:

$$\min \frac{1}{2} \sum_{i,j} \|x_i - x_j\|^2 W_{w,ij}, \tag{197}$$

$$\max \frac{1}{2} \sum_{i,j} \|x_i - x_j\|^2 W_{b,ij}, \tag{198}$$

considering matrix Z as $\left[x_1^T; \ldots; x_N^T\right]$, these equations can be written as (RADUCANU; DORNAIKA, 2012):

$$\min \operatorname{Tr}\left(Z^T L_w Z\right), \tag{199}$$

$$\max \operatorname{Tr}\left(Z^T L_b Z\right), \tag{200}$$

where $L_w = D_w - W_w$ and $L_b = D_b - W_b$. Using scale restriction $Z^T D_w Z = I$ and equation $L_w = D_w - W_w$, both functions can be combined into:

$$\arg\max_Z \left(\gamma \operatorname{Tr}\left(Z^T L_b Z\right) + (1-\gamma) \operatorname{Tr}\left(Z^T W_w Z\right)\right) \text{ subject to } Z^T D_w Z = I. \tag{201}$$

By using matrix $B = \gamma L_b + (1-\gamma) W_w$, the problem becomes:

$$\arg\max_Z \operatorname{Tr}\left(Z^T B Z\right) \text{ subject to } Z^T D_w Z = I, \tag{202}$$

resulting in a generalised eigenvalues problem in the form (RADUCANU; DORNAIKA, 2012):

$$Bz = \lambda D_w z. \tag{203}$$

S-LE differs from classic LE by having as a goal solely preserving the locality of samples and not regarding data classes, while the former divides the graph in two depending on how classes match. Another disadvantage of classic LE is that neighbourhood size must be chosen before the user runs the algorithm and can affect its results, which does not happen with S-LE, as it determines neighbourhood size from the samples. This way, graph construction automatically adapts to any data set with adjusting parameters (RADUCANU; DORNAIKA, 2012).

2.10 t-Distributed Stochastic Neighbour Embedding

In order to understand t-Distributed Stochastic Neighbour Embedding (t-SNE), one needs to know its predecessor, Stochastic Neighbour Embedding (SNE). Initially, this method converts the Euclidean distances between data points $\vec{x}_i \in \mathbb{R}^m$, for $i = 1, 2, \ldots, n$, into conditional probabilities to represent similarities. Similarity between two points $\vec{x}_i$ and $\vec{x}_j$ is the conditional probability $p_{j|i}$ that $\vec{x}_i$ picks $\vec{x}_j$ as neighbour if selection is made in proportion to a probability density under a Gaussian centred in $\vec{x}_i$ (MAATEN; HINTON, 2008):

$$p_{j|i} = \frac{\exp\left(-\frac{\|\vec{x}_i - \vec{x}_j\|^2}{2\sigma_i^2}\right)}{\sum_{k \neq i} \exp\left(-\frac{\|\vec{x}_i - \vec{x}_k\|^2}{2\sigma_i^2}\right)}, \tag{204}$$

where σ_i^2 is the variance of the Gaussian centred in $\vec{x}_i$. It can be noted that, if $\vec{x}_i$ and $\vec{x}_j$ are close, $p_{j|i}$ has a significantly high value, but if they are far apart, $P_{j|i}$ tends to zero. It is possible to calculate a similar conditional probability for the low dimension representation of vectors $\vec{y}_i$ and $\vec{y}_j$ to $\mathbb{R}^d$, denoted by $q_{j|i}$. Setting the variance of Gaussian in $\frac{1}{\sqrt{2}}$, the similarity measure is given by:

$$q_{j|i} = \frac{\exp\left(-\|\vec{y}_i - \vec{y}_j\|^2\right)}{\sum_{k \neq i} \exp\left(-\|\vec{y}_i - \vec{y}_k\|^2\right)}. \tag{205}$$

Since similarities are modelled pairwise, $p_{i|i} = q_{i|i} = 0$. The idea is that, if points $\vec{y}_i$ and $\vec{y}_j$ of the low dimension representation correctly model the similarities between vectors $\vec{x}_i$ and $\vec{x}_j$ of high dimension, then probabilities $p_{j|i}$ and $q_{j|i}$ are equal. The goal of SNE is to find a low dimension representation that minimises the distance between both probabilities, approximating them as much as possible. A statistical measure for the closeness of two probability distributions is KL-divergence, also known as relative entropy (KULLBACK; LEIBLER, 1951). SNE minimises the sum of KL-divergences over all data points using the gradient descent method. The objective function to be minimised is (HINTON; ROWEIS, 2003):

$$C = \sum_{i=1}^{n} KL\left(P_i \| Q_i\right) = \sum_{i=1}^{n} \sum_{j=1}^{n} p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}, \tag{206}$$

where P_i represents the conditional probability distribution over all other data points from point $\vec{x}_i$, and Q_i the conditional probability distribution over all other map points from point $\vec{y}_i$. Which is to say, the SNE cost function focuses on keeping the local structure of data in the map (for reasonable values of variance for the Gaussian in high dimension space, σ_i^2).

2.10.1 Defining variance for Gaussian functions

In order to work properly, SNE needs to define variance σ_i^2 for the Gaussian centred on each high dimension point $\vec{x}_i \in \mathbb{R}^m$. It is unreasonable to assume there is a single value of σ_i^2 that is optimal for every point. In sparse patches, a smaller value of σ_i^2 is more adequate than in denser patches. A given σ_i^2 induces a distribution probability P_i, which has as entropy a crescent function of variance. The measure of perplexity is defined as (MAATEN; HINTON, 2008):

$$\text{Perp}\left(P_i\right) = 2^{H(P_i)}, \tag{207}$$

where $H\left(P_i\right)$ is the Shannon entropy (in bits):

$$H\left(P_i\right) = -\sum_{j=1}^{n} p_{j|i} \log_2 p_{j|i}. \tag{208}$$

SNE seeks a value of σ_i^2 that produces a fixed, user defined, perplexity P_i, which can be interpreted as a smooth measure of the effective number of neighbours, and its values are usually between 5 and 50 (MAATEN; HINTON, 2008). Minimisation of objective function (KL-divergence) is done using gradient descent with momentum to achieve convergence, that is, after initialisation, coordinates of the low dimension points are iteratively updated by:

$$\mathcal{Y}^{(t)} = \mathcal{Y}^{(t-1)} - \eta \frac{\partial C}{\partial \mathcal{Y}} + \alpha\left(t\right)\left(\mathcal{Y}^{(t-1)} - \mathcal{Y}^{(t-2)}\right), \tag{209}$$

where $\mathcal{Y}^{(t)}$ denotes the solution in iteration t, η the learning rate, and $\alpha\left(t\right)$ the momentum of iteration t.

2.10.2 Calculating gradient in Stochastic Neighbour Embedding

Equation (209) shows that the main component of the iterative algorithm is calculating the gradient. In each step of the optimisation, it begins from a set of points $\vec{y}_1, \vec{y}_2, \ldots, \vec{y}_n$ and requires estimating the gradient. The process can be summarised as (MELVILLE, 2015):

1. Create a distances matrix, where element d_{ij} represents the Euclidean distance between $\vec{y}_i$ and $\vec{y}_j$;

2. Transform distances to create f_{ij} (usually, they are squared);

3. Apply a weighing function to define weight w_{ij}, such that, the larger the weight, smaller the distance (similarity measure);

4. Convert weights into probabilities $q_{ij} = q_{j|i}$, normalising their sum.

Having the probabilities, it is possible to compute the gradient. In the original SNE algorithm, weights are converted into probability by:

$$q_{j|i} = q_{ij} = \frac{w_{ij}}{\sum_k w_{ik}} = \frac{w_{ij}}{Z_i}, \tag{210}$$

where $w_{ij} = \exp\left(-\left\|\vec{y}_i - \vec{y}_j\right\|^2\right)$ and $w_{ij} = w_{ji}$. Renaming variables i, j into k, l, the objective function becomes (ERRICA, 2018):

$$C = \sum_{k=1}^{n}\sum_{l=1}^{n}\left(p_{kl}\log p_{kl} - p_{kl}\log p_{kl}\right) = \sum_{k=1}^{n}\sum_{l=1}^{n}\left(p_{kl}\log p_{kl} - p_{kl}\log w_{kl} + p_{kl}\log Z_k\right). \tag{211}$$

Differentiating with respect to $\vec{y}_i$, comes:

$$\frac{\partial C}{\partial \vec{y}_i} = -\sum_{k=1}^{n}\sum_{l=1}^{n} p_{kl}\frac{\partial}{\partial \vec{y}_i}\log w_{kl} + \sum_{k=1}^{n}\sum_{l=1}^{n} p_{kl}\frac{\partial}{\partial \vec{y}_i} p_{kl}\log Z_k. \tag{212}$$

It can be noted that, in the first term, the derivative is not zero when $k = i$ or $l = i$, which leads to:

$$-\sum_{k=1}^{n}\sum_{l=1}^{n} p_{kl}\frac{\partial}{\partial \vec{y}_i}\log w_{kl} = -\sum_{j=1}^{n}\left(p_{ij}\frac{\partial}{\partial \vec{y}_i}\log w_{ij} + p_{ji}\frac{\partial}{\partial \vec{y}_i}\log w_{ji}\right). \tag{213}$$

Differentiating $\log w_{ij}$ with respect to $\vec{y}_i$:

$$\frac{\partial}{\partial \vec{y}_i}\log w_{ij} = -2\frac{1}{w_{ij}} w_{ij}\left(\vec{y}_i - \vec{y}_j\right) \tag{214}$$

and the derivative of $\log w_{ji}$ with respect to $\vec{y}_i$ is:

$$\frac{\partial}{\partial \vec{y}_i}\log w_{ji} = -2\frac{1}{w_{ji}} w_{ji}\left(\vec{y}_j - \vec{y}_i\right). \tag{215}$$

Then, the first term of the gradient becomes:

$$\begin{aligned} -\sum_{j=1}^{n}\left(-2p_{ij}\frac{1}{w_{ij}}w_{ij}\left(\vec{y}_i - \vec{y}_j\right) + 2p_{ji}\frac{1}{w_{ji}}w_{ji}\left(\vec{y}_j - \vec{y}_i\right)\right) = \\ -\sum_{j=1}^{n}\left(-2p_{ij}\left(\vec{y}_i - \vec{y}_j\right) - 2p_{ji}\left(\vec{y}_i - \vec{y}_j\right)\right) = 2\sum_{j=1}^{n}\left(p_{ij} + p_{ji}\right)\left(\vec{y}_i - \vec{y}_j\right). \end{aligned} \tag{216}$$

In the second term, Z_k does not depend on l, and can therefore be removed from the inner summation:

$$\sum_{k=1}^{n}\sum_{l=1}^{n} p_{kl}\frac{\partial}{\partial \vec{y}_i}\log Z_k = \sum_{k=1}^{n}\frac{\partial}{\partial \vec{y}_i}\log Z_k\sum_{l=1}^{n} p_{kl} = \sum_{k=1}^{n}\frac{\partial}{\partial \vec{y}_i}\log Z_k, \tag{217}$$

as the sum of elements of P_k (a row of matrix P) is one. Replacing variable k by j and differentiating the logarithm, comes:

$$\sum_{j=1}^{n}\frac{1}{Z_j}\sum_{k=1}^{n}\frac{\partial w_{jk}}{\partial \vec{y}_i}. \tag{218}$$

Since the partial derivative is not zero only when $k = i$ or $l = i$, it can be written:

$$\sum_{j=1}^{n} \left[\frac{1}{Z_j} \frac{\partial w_{ji}}{\partial \vec{y}_i} + \frac{1}{Z_i} \frac{\partial w_{ij}}{\partial \vec{y}_i} \right]. \tag{219}$$

Differentiating leads to:

$$\begin{aligned} \sum_{j=1}^{n} \left(\frac{1}{Z_j} 2 w_{ji} \left(\vec{y}_j - \vec{y}_i \right) \right) - \sum_{j=1}^{n} \left(\frac{1}{Z_i} w_{ij} \left(\vec{y}_i - \vec{y}_j \right) \right) = \\ - 2 \sum_{j=1}^{n} \left(\frac{w_{ji}}{Z_j} \left(\vec{y}_i - \vec{y}_j \right) \right) - 2 \sum_{j=1}^{n} \left(\frac{w_{ij}}{Z_i} \left(\vec{y}_i - \vec{y}_j \right) \right) = -2 \sum_{j=1}^{n} \left(q_{ji} + q_{ij} \right) \left(\vec{y}_i - \vec{y}_j \right). \end{aligned} \tag{220}$$

Finally, combining equations (216) and (220), comes:

$$\begin{aligned} \frac{\partial C}{\partial \vec{y}_i} &= 2 \sum_{j=1}^{n} \left(p_{ij} + p_{ji} \right) \left(\vec{y}_i - \vec{y}_j \right) - 2 \sum_{j=1}^{n} \left(q_{ji} + q_{ij} \right) \left(\vec{y}_i - \vec{y}_j \right) \\ &= 2 \sum_{j=1}^{n} \left(p_{ij} - q_{ij} + p_{ji} - q_{ji} \right) \left(\vec{y}_i - \vec{y}_j \right). \end{aligned} \tag{221}$$

2.10.3 Limitations of Stochastic Neighbour Embedding

SNE, although constructing reasonably fine visualisations, is harmed by a hard to optimise cost function and a "crowding problem". This can be summarised as: the available area in a low dimension map to position the points is not large enough in comparison to the available area for close points, that is, to adequately model small distances in the map, most distant points would have to be too scattered. The t-SNE algorithm aims to mitigate these problems using a long-tail distribution in the low dimension space (MAATEN; HINTON, 2008).

2.10.4 Calculating gradient in t-Distributed Stochastic Neighbour Embedding

The cost function of t-SNE differs from that of SNE in two ways: it uses a symmetrised version of the SNE cost function with simpler gradients (COOK et al., 2007) and a Student's t distribution to compute the similarity between two points in the low dimensional space instead of a Gaussian. An alternative to minimising the sums of KL-divergences between conditional probabilities $p_{j|i}$ and $q_{j|i}$ is minimising a single divergence over joint probability distributions P, in the high dimension space, and Q, in the low (MAATEN; HINTON, 2008):

$$C = KL\left(P \| Q\right) = \sum_{i} \sum_{j} p_{ij} \log \frac{p_{ij}}{q_{ij}}, \tag{222}$$

where $p_{ii} = q_{ii} = 0$. This type of SNE, named symmetrical for having $p_{ij} = p_{ji}$ and $q_{ij} = q_{ji}$, $\forall i, j$, has pairwise similarities in the high dimension map p_{ij} given by:

$$p_{ij} = \frac{\exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)}{\sum_{k \neq l} \exp\left(-\frac{\|x_k - x_l\|^2}{2\sigma^2}\right)}, \tag{223}$$

where the sum of the denominator of normalisation involves all pairs of points, not only those connected to $\vec{x}_i$, as in SNE.

Using the Student's t distribution with one degree of freedom, joint probabilities q_{ij} are defined as:

$$q_{ij} = \frac{\left(1 + \|\vec{y}_i - \vec{y}_j\|^2\right)^{-1}}{\sum_{k \neq l} \left(1 + \|\vec{y}_k - \vec{y}_l\|^2\right)^{-1}} = \frac{w_{ij}^{-1}}{\sum_{k \neq l} w_{kl}^{-1}} = \frac{w_{ij}^{-1}}{Z}. \tag{224}$$

For the first term of the gradient, it can be noted that the derivative is non-zero when $\forall j, k = i$ or $l = i$, and that $p_{ij} = p_{ji}$ and $q_{ij} = q_{ji}$, which leads to:

$$\begin{aligned} -\sum_{k=1}^{n}\sum_{l=1}^{n} p_{kl} \frac{\partial}{\partial \vec{y}_i} \log w_{kl}^{-1} &= -\sum_{j=1}^{n} \left(p_{ij} \frac{\partial}{\partial \vec{y}_i} \log {w_{ij}}^{-1} + p_{ji} \frac{\partial}{\partial \vec{y}_i} \log w_{ji}^{-1} \right) \\ &= -2 \sum_{j=1}^{n} p_{ij} \frac{\partial}{\partial \vec{y}_i} \log w_{ij}^{-1}. \end{aligned} \tag{225}$$

The derivative of reverse weight is:

$$\frac{\partial w_{ij}^{-1}}{\partial \vec{y}_i} = -2 w_{ij}^{-2} \left(\vec{y}_i - \vec{y}_j\right), \tag{226}$$

therefore the first term of the gradient is:

$$4 \sum_{j=1}^{n} p_{ij} w_{ij}^{-1} \left(\vec{y}_i - \vec{y}_j\right). \tag{227}$$

To differentiate the second term of the gradient, it is assumed that Z depends on neither k nor l, and that the sum of probabilities equals one:

$$\begin{aligned} \sum_{k=1}^{n}\sum_{l=1}^{n} p_{kl} \frac{\partial}{\partial \vec{y}_i} \log Z = \frac{\partial}{\partial \vec{y}_i} \log Z \sum_{k=1}^{n}\sum_{l=1}^{n} p_{kl} = \frac{\partial}{\partial \vec{y}_i} \log Z = \frac{\partial}{\partial \vec{y}_i} \log Z \\ = \frac{1}{Z} \frac{\partial}{\partial \vec{y}_i} Z = \frac{1}{Z} \sum_{k=1}^{n}\sum_{l=1}^{n} \frac{\partial}{\partial \vec{y}_i} w_{kl}^{-1}. \end{aligned} \tag{228}$$

Again, the derivative is non-zero when $\forall j, k = i$ or $l = i$, that is:

$$\frac{1}{Z} \sum_{k=1}^{n}\sum_{l=1}^{n} \frac{\partial}{\partial \vec{y}_i} w_{kl}^{-1} = \frac{1}{Z} \sum_{j=1}^{n} \left(\frac{\partial}{\partial \vec{y}_i} w_{ij}^{-1} \frac{\partial}{\partial \vec{y}_i} w_{ji}^{-1} \right). \tag{229}$$

As $w_{ij} = w_{ji}$, comes:

$$\frac{1}{Z} \sum_{j=1}^{n} \left(\frac{\partial}{\partial \vec{y}_i} w_{ij}^{-1} \frac{\partial}{\partial \vec{y}_i} w_{ji}^{-1} \right) = 2 \sum_{j=1}^{n} \frac{1}{Z} \frac{\partial}{\partial \vec{y}_i} w_{ij}^{-1}. \tag{230}$$

From equation (226), one can write:

$$2\sum_{j=1}^{n}\frac{1}{Z}\frac{\partial}{\partial\vec{y}_i}w_{ij}^{-1}=-4\sum_{j=1}^{n}\frac{w_{ij}^{-1}}{Z}w_{ij}^{-1}\left(\vec{y}_i-\vec{y}_j\right)=-4\sum_{j=1}^{n}q_{ij}w_{ij}^{-1}\left(\vec{y}_i-\vec{y}_j\right). \tag{231}$$

Finally, combining equations (227) and (231) leads to the gradient for an iteration of t-SNE:

$$\frac{\partial C}{\partial\vec{y}_i}=4\sum_{j=1}^{n}\left(p_{ij}-q_{ij}\right)w_{ij}^{-1}\left(\vec{y}_i-\vec{y}_j\right)=4\sum_{j=1}^{n}\left(p_{ij}-q_{ij}\right)\left(1+\left\|\vec{y}_i-\vec{y}_j\right\|^2\right)^{-1}\left(\vec{y}_i-\vec{y}_j\right). \tag{232}$$

2.11 Uniform Manifold Approximation and Projection

A method with many similarities to t-SNE is Uniform Manifold Approximation and Projection (UMAP), based on the assumptions that there is a distance measure according to which data are approximately uniformly distributed in the manifold and that this is locally connected. To ensure the validity of the first assumption, a measure that approaches the distances of the k nearest points to a given set X centred in X_i is chosen. This creates a different notion of distance for each X_i which must be united in a consistent global structure. To ensure the second assumption, these metric spaces must be converted into simplicial fuzzy sets. However, from a computational standpoint, this can be described as a weighed graph (MCINNES; HEALY; MELVILLE, 2020).

Like other algorithms based on graphs with k neighbours, UMAP can be described in steps. In the first one, the weighed graph with k neighbours is constructed and in the second, the graph layout in low dimension is calculated. The UMAP algorithm presumes the validity of three axioms:

1. A manifold exists in which data are uniformly distributed.
2. The underlying manifold of interest is locally connected.
3. The main goal is to preserve the topological structure of this manifold.

All algorithms that use a mathematical structure such as that of a graph with k neighbours to approximate a manifold have a similar basic structure (MCINNES; HEALY; MELVILLE, 2020):

❏ Graph Construction:

1. Construct a weighed graph with k neighbours;
2. Apply a transform on the vertices to ambient local distance;
3. Solve the inherent symmetry to the graph with k neighbours.

❏ Graph Layout:

1. Define an objective function that preserves desired features of this graph with k neighbours;
2. Find a representation in low dimension that optimises this objective function.

Many DR algorithms can be divided into these steps as they are fundamental for a particular class of solutions.

2.11.1 Graph Construction

Constructing the graph with k neighbours can be considered the first step of UMAP. Let $X = \{x_1, x_2, \ldots, x_N\}$ be the input dataset, with a metric (or dissimilarity measure) $d : X \times X \to \mathbb{R}_{\geq 0}$. Given an input hyper-parameter k, for each x_i is calculated a set $\{x_{i_1}, x_{i_2}, \ldots, x_{i_k}\}$ of the KNN of x_i under metric d. This calculation can be done using any algorithm for search of nearest neighbours, being the one chosen in this case the descending nearest neighbour algorithm (DONG; MOSES; LI, 2011).

For each x_i, ρ_i and σ_i are defined. Let:

$$\rho_i = \min\left(d\left(x_i, x_{i_j}\right) \mid 1 \leq j \leq k, d\left(x_i, x_{i_j}\right) > 0\right), \tag{233}$$

and σ_i defined as a value such that:

$$\sum_{j=1}^{n} \exp\left(\frac{-\max\left(0, d\left(x_i, x_{i_j}\right) - \rho_i\right)}{\sigma_i}\right) = \log_2(k). \tag{234}$$

The choice of ρ_i guarantees that x_i is connected to at least one other data point with an edge of weight 1, which equals to the fuzzy simplicial set being locally connected in x_i. The choice of σ_i corresponds to a (smooth) normalisation factor, defining the local Riemannian metric in the point x_i.

Then can be defined a weighed graph $\bar{G} = (V, E, w)$, the vertices V of which are the X set. Thus the $E = \left\{\left(x_i, x_{i_j}\right) \mid 1 \leq j \leq k, 1 \leq i \leq N\right\}$ set of directed edges is formed, and the weight function w defined as:

$$w\left(\left(x_i, x_{i_j}\right)\right) = \exp\left(\frac{-\max\left(0, d\left(x_i, x_{i_j}\right) - \rho_i\right)}{\sigma_i}\right). \tag{235}$$

For a given point x_i, exists an inducted graph of x_i and edges inbound to x_i. The weight of an edge can be considered as the probability that this edge exists. Given this set of local graphs (represented as a single directed graph), a method to combine them into a unified topological representation is needed. Let A be the weighed adjacency matrix of $\bar{G}$, considering the symmetric matrix:

$$B = A + A^{\top} - A \circ A^{\top}, \tag{236}$$

where $\circ$ is the Hadamard (or pointwise) product. If the value of A_{ij} is interpreted as the probability of directed edge from x_i to x_j existing, then B_{ij} is the probability of at least

one directed edge (from x_i to x_j and from x_j to x_i) existing. Graph G of UMAP is, then, an undirected graph with its adjacency matrix is given by B.

2.11.2 Graph Layout

In practice, UMAP uses a force directed graph layout algorithm in low dimension space, which uses a set of attractive forces applied along the edges and a set of repulsive forces among the vertices. Every force directed layout algorithm needs a description of attractive and repulsive forces, and is applied iteratively over vertices and edges, which results in a non-convex optimisation problem. Convergence to a local minimum is assured by slowly reducing attractive and repulsive forces in a similar way to that used in simulated annealing (MCINNES; HEALY; MELVILLE, 2020).

In UMAP, attractive force between two vertices i and j in coordinates $\vec{y}_i$ and $\vec{y}_j$, respectively, is determined by:

$$\frac{-2ab\left\|\vec{y}_i-\vec{y}_j\right\|_2^{2(b-1)}}{1+\left\|\vec{y}_i-\vec{y}_j\right\|_2^2} w\left(\left(x_i,x_i\right)\right)\left(\vec{y}_i-\vec{y}_j\right), \tag{237}$$

where a and b are hyper-parameters. Repulsive forces are calculated by sampling due to computational restrictions. Hence, when an attractive force is applied along an edge, a vertex of this edge is repelled by a sampling of the other vertices. Repulsive force is given by:

$$\frac{2b}{\left(\epsilon+\left\|\vec{y}_i-\vec{y}_j\right\|_2^2\right)\left(1+a\left\|\vec{y}_i-\vec{y}_j\right\|_2^{2b}\right)}\left(1-w\left(\left(x_i,x_j\right)\right)\right)\left(\vec{y}_i-\vec{y}_j\right), \tag{238}$$

where ϵ is a small value to prevent division by zero.

These forces derive from the gradient that minimises cross entropy of edges between the weighed graph G and an equivalent H constructed from points $\{\vec{y}_i\}_{i=1\ldots N}$. That is, one wishes to position points y_i in a way that the weighed graph induced by them approaches G as much as possible, being the difference between weighed graphs measured by the total cross entropy over all probabilities of existence of the edges. Since weighed graph G captures the topology of original data, equivalent H constructed from points $\{\vec{y}_i\}_{1\ldots N}$ approaches the topology as much as optimisation allows, and therefore provides a good representation in low dimension of the general topology of data.

2.12 Locality Preserving Projections

At first, Locality Preserving Projections (LPP) seems to be the same as LE, but there is an important difference: as LE is a ML method, it suffers from the out-of-sample problem, that is, once the low dimension representation of data is found, any new data points analysed after that cannot be mapped directly to the low dimension representation (these new data points would have to be incorporated in the dataset and LE rerun in this larger

dataset). LPP was proposed as a way to avoid this out-of-sample problem. Basically, it linearises the LE method to compute a projection matrix W that can be used to map any new data points to the low dimension space. Therefore, in summary, LPP could be understood as a linear approximation of LE. In practice, both use the Laplacian matrix of the input graph, but in different ways: in LPP the Laplacian matrix must be multiplied by the data matrix before spectral decomposition, leading to a slightly higher computational cost. However, with the advantage of overcoming the out-of-sample problem, overall, LPP becomes less costly than multiple executions of LE.

The goal is to find a smooth map that preserves locality, similar to the LE approach. That is, proximity in the graph must imply proximity in the line. It can be demonstrated that, if the following criterion is minimised, map $\vec{y} = [y_1, y_2, \ldots, y_n]$ is optimal in that sense (XIAOFEI; NIYOGI, 2003):

$$\vec{y}^T L \vec{y} = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} \left(y_i - y_j\right)^2, \tag{239}$$

where L is the Laplacian matrix of the KNN graph induced by the $m \times n$ data matrix $X = [\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n]$. The edge weights w_{ij} are a major factor in the optimisation problem. The intuition behind this equations is that proximity in the graph must imply in proximity in the resulting embedding. In this method, the objective is to find a better measure than the regular Euclidean distance, by employing the Probabilistic Nearest Neighbours (PNN) method, knowing that the Euclidean distance is sensitive to outliers in data.

In LPP, it is assumed that the relationship between $\vec{x}_i \in R^m$ and $y_i \in R$ is linear, that is, $y_i = \vec{a}^T \vec{x}_i$, where $\vec{a} \in R^m$ is a column vector. Hence, the objective function is:

$$\begin{aligned}
\vec{y}^T L \vec{y} = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} \left(\vec{a}^T \vec{x}_i - \vec{a}^T \vec{x}_j\right)^2 &= \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} \left[\vec{a}^T \vec{x}_i \vec{x}_i^T \vec{a} - 2\vec{a}^T \vec{x}_i \vec{x}_j^T \vec{a} + \vec{a}^T \vec{x}_j \vec{x}_j^T \vec{a}\right] \\
&= \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} 2 w_{ij} \vec{a}^T \vec{x}_i \vec{x}_i^T \vec{a} - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} 2 w_{ij} \vec{a}^T \vec{x}_i \vec{x}_j^T \vec{a} \\
&= \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} \vec{a}^T \vec{x}_i \vec{x}_i^T \vec{a} - \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} \vec{a}^T \vec{x}_i \vec{x}_j^T \vec{a}.
\end{aligned} \tag{240}$$

Since $d_i = \sum_{j=1}^{n} w_{ij}$, comes:

$$\vec{y}^T L \vec{y} = \sum_{i=1}^{n} \vec{a}^T \vec{x}_i d_i \vec{x}_i^T \vec{a} - \sum_{i=1}^{n} \sum_{j=1}^{n} \vec{a}^T \vec{x}_i w_{ij} \vec{x}_j^T \vec{a}, \tag{241}$$

which is equivalent to:

$$\vec{y}^T L \vec{y} = \vec{a}^T X D X^T \vec{a} - \vec{a}^T X W X^T \vec{a}, \tag{242}$$

where X is the $m \times n$ data matrix, D is the $n \times n$ diagonal matrix of degrees, and W is the $n \times n$ weights matrix. Knowing that $L = D - W$, finally:

$$\vec{y}^T L \vec{y} = \vec{a}^T X \left(D - W\right) X^T \vec{a} = \vec{a}^T X L X^T \vec{a}. \tag{243}$$

Thus, the following constrained minimisation problem must be solved:

$$\arg\min_{\vec{a}} \vec{a}^T X L X^T \vec{a} \text{ subject to } \vec{a}^T X D X^T \vec{a} = 1, \tag{244}$$

where the constraint is a general form to express that the norm of vector $\vec{a}$ is a constant (as the direction of the vector is important in this case, not its magnitude). The Lagrangian function is given by:

$$L(\vec{a}, \lambda) = \vec{a}^T X L X^T \vec{a} - \lambda \left(\vec{a}^T X D X^T \vec{a} - 1 \right). \tag{245}$$

Differentiating with respect to vector $\vec{a}$ and setting the result to zero:

$$\frac{\partial}{\partial \vec{a}} L(\vec{a}, \lambda) = X L X^T \vec{a} - \lambda X D X^T \vec{a} = 0. \tag{246}$$

Which leads to a generalised eigenvectors problem:

$$X L X^T \vec{a} = \lambda X D X^T \vec{a}, \tag{247}$$

$$\left(X D X^T \right)^{-1} X L X^T \vec{a} = \lambda \vec{a}, \tag{248}$$

showing that, in order to minimise the objective function, vector a must be selected as the least eigenvector of matrix $\left(X D X^T \right)^{-1} X L X^T$. The multivariate version of the problem considers a $m \times d$ matrix A where each column $\vec{a}_j$ represents a direction in which data is to be projected:

$$\left(X D X^T \right)^{-1} \left(X L X^T \right) A = \lambda A. \tag{249}$$

In this situation must be chosen the d eigenvectors associated with the d least eigenvalues of $(XDXT)^{-1} XLXT$ to build the columns of matrix A. It must be noted that the transformation matrix A has m rows and d columns, while the output matrix Y has d rows and n columns, implying that each column $\vec{y}_j$ for $j = 1, 2, \ldots, n$ stores the coordinates of the j-th sample in the output space (after DR).

2.12.1 Graph-Based Learning

Graph-based learning methods are usually organised in two categories: graph construction and label inference. In this method, the focus is on the graph construction process. To build a graph that is a discrete approximation of a manifold, there are two basic steps: first, to define, from the set of all possible edges, which of them are to be created, and second, to assign a weight to every edge created in the previous step. In the following, two strategies for graph construction shall be discussed: Clustering with Adaptive Neighbours (CAN) (NIE; WANG; HUANG, 2014) and PNN (JUNLIANG; BING; CHENG, 2020).

2.12.1.1 Clustering with Adaptive Neighbours

As the name suggests, this graph construction method is more suitable for data clustering than classification. Let s_{ij} denote the probability that sample $\vec{x}_j$ is a neighbour of $\vec{x}_i$. Then, vector $\vec{s}_i \in R^n$ is composed by the probabilities of each sample in the dataset being a neighbour of $\vec{x}_i$. Intuitively, the smaller the distance $d(\vec{x}_i, \vec{x}_j)$, greater the probability s_{ij} should be. The optimal probabilities for a single sample $\vec{x}_i$ are given by the solution to the following optimisation problem:

$$\min J(\vec{s}_i) = \sum_{j=1}^{n} \left(\|\vec{x}_i - \vec{x}_j\|^2 s_{ij} + \gamma_i s_{ij}^2 \right) \text{ subject to } \vec{s}_i^T \vec{1} = 1, \tag{250}$$

where $0 \leq s_{ij} \leq 1$, n denotes the number of samples, $\gamma_i > 0$ is a regularisation parameter, and the constraint is necessary to enforce that the sum of probabilities equals one. The first term of the objective function expresses that the edge weights are penalised by the Euclidean distance between the vertices, while the second term serves a smooth constraint about the solution, attempting to assess if all samples can be neighbours of $\vec{x}_i$ with roughly the same probability. Clearly, none of the limiting cases are especially interesting, but the goal is to find a good trade-off between data fidelity and prior knowledge.

Let $\vec{d}_i$ be the vector of squared Euclidean distances between the i-th sample $\vec{x}_i$ and all the other samples in the dataset. Then, the optimisation problem can be expressed in terms of a vector norm:

$$\min J(\vec{s}_i) = \left\| \vec{s}_i + \frac{1}{2\gamma_i} \vec{d}_i \right\|^2 \text{ subject to } \vec{s}_i^T \vec{1} = 1, \tag{251}$$

where $0 \leq s_{ij} \leq 1$.

The Lagrangian function incorporates all the constraints into the objective function, leading to:

$$L\left(\vec{s}_i, \eta, \vec{\beta}_i\right) = \frac{1}{2} \left\| \vec{s}_i + \frac{1}{2\gamma_i} \vec{d}_i \right\|^2 - \eta \left(\vec{s}_i^T \vec{1} - 1 \right) + \vec{\beta}_i^T \vec{s}_i, \tag{252}$$

where η and $\vec{\beta}_i$ are the Lagrange multipliers. Differentiating with respect to s_{ij} and setting the result to zero leads to:

$$s_{ij} = -\frac{d_{ij}}{2\gamma_i} + \eta, \tag{253}$$

where $d_{ij} = \|\vec{x}_i - \vec{x}_j\|^2$. To preserve the local geometric properties of the manifold, it is recommended that each sample is linked only to a fixed number $k < n$ of neighbours. Without loss of generality, one can assume the distances in $\vec{d}_i$ are sorted in ascending order.

Considering that only the first k components in the optimal $\vec{s}_i$ are non-zero, it can be written:

$$-\frac{d_{ik}}{2\gamma_i} + \eta > 0, \tag{254}$$

$$-\frac{d_{ik+1}}{2\gamma_i} + \eta = 0. \tag{255}$$

By the constraint that the sum of all the elements of $\vec{s_i}$ must equal (since they denote probabilities) one, comes:

$$\sum_{j=1}^{k}\left(-\frac{d_{ij}}{2\gamma_i}+\eta\right)=1. \tag{256}$$

From here, it can be shown that this minimisation problem has a closed form solution, given by (NIE; WANG; HUANG, 2014):

$$s_{ij}=\frac{d_{ik+1}-d_{ij}}{kd_{ik+1}-\sum_{j=1}^{k}d_{ij}}, \tag{257}$$

where the vector $\vec{d_i}$ stores all the distances from $\vec{x}_i$ to other samples in ascending order.

2.12.1.2 Probabilistic Nearest Neighbours

The PNN approach is more suitable for classification problems in label propagation algorithms (JUNLIANG; BING; CHENG, 2020). Intuitively, the reason is that, the larger the s_{ij}, more influence the sample $\vec{x}_i$ has on the labelling of a neighbouring sample $\vec{x}_j$. In classification problems, ideally, the variance of the propagation probabilities should be large enough to reflect the decision boundaries defined by samples belonging to different classes. To achieve this goal, a min-max normalisation scheme is employed, which is usual to normalise data. After this transformation, all variables have the exact same scale, converting the data into the $[0,1]$ interval. According to this process, the normalisation of a vector $\vec{x}_i$ is:

$$x_{ij}^{*}=\frac{x_{ij}-\min\left(\vec{x}_i\right)}{\max\left(\vec{x}_i\right)-\min\left(\vec{x}_i\right)}. \tag{258}$$

The main difference between CAN and PNN is the computation of γ_i, which is:

$$\gamma_i=\frac{d_{ik+1}-d_{i1}}{d_{ij}-d_{i1}}\left(\frac{k}{2}d_{ik+1}-\frac{1}{2}\sum_{j=1}^{k}d_{ij}\right). \tag{259}$$

It has been shown that the optimal probabilities in the PNN method are given by (JUNLIANG; BING; CHENG, 2020):

$$s_{ij}=\frac{d_{ik+1}-d_{ij}}{d_{ik+1}-d_{i1}}, \tag{260}$$

where the vector $\vec{d_i}$ stores all the distances from $\vec{x}_i$ to other samples in ascending order. It is worth mentioning that the computational complexity of the KNN graph construction is $O\left(mn^2\right)$ (quadratic), where m is the dimension of the input space and n is the number of samples. According to the authors, CAN and PNN methods, on the other hand, have closed form solutions, leading to an $O\left(n\right)$ (linear) computational complexity (JUNLIANG; BING; CHENG, 2020).

2.13 Evaluating Results Over Traditional Methods'

To ascertain the extent of the proposed methods' relevance in comparison to the traditional ones, it is necessary to test them according to some metric. Thus, some tests were considered as ways to evaluate the effectiveness of proposed methods when applied over the same datasets as the originals. These are described in this section.

2.13.1 Silhouette Coefficient

Silhouette Coefficient (SC) is a method for interpretation and validation of consistency in data clusters (ROUSSEEUW, 1987). Let C_i be the i-th cluster, for each data point $i \in C_i$, $a(i)$ is the mean distance between i and every other point in the same cluster C_i:

$$a(i) = \frac{1}{|C_i| - 1} \sum_{j \in C_i, j \neq i} d(i,j), \tag{261}$$

where $d(i,j)$ is the distance between data points i and j in cluster C_i. One can interpret $a(i)$ as a measure of how adequate the assignment of a data point i is to its cluster (lower is better). Thus, mean dissimilarity of a data point i to a cluster C is defined as the average of all distances from i to every point in C. For each point i, $b(i)$ is the least mean distance from i to all points in any cluster it does not belong to:

$$b(i) = \min_{k \neq i} \frac{1}{|C_k|} \sum_{j \in C_k} d(i,j). \tag{262}$$

The cluster with least mean dissimilarity is a neighbour of i, as it is the second best choice for clustering point i. Let:

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}, \text{ if } |C_i| \leq 1, \tag{263}$$

the silhouette value of data point i is:

$$s(i) = 0, \text{ if } |C_i| = 1. \tag{264}$$

Combining both definitions, comes:

$$s(i) = \begin{cases} 1 - \frac{a(i)}{b(i)}, & \text{if } a(i) < b(i) \\ 0, & \text{if } a(i) = b(i) \\ \frac{b(i)}{a(i)} - 1, & \text{if } a(i) > b(i) \end{cases}. \tag{265}$$

It can be noted that $-1 \leq s(i) \leq 1$. A $s(i)$ close to one means that data are properly clustered. If $s(i)$ tends to minus one, i must be grouped in the neighbouring cluster instead. A $s(i)$ close to zero means that the point is in the border between two natural clusters. Mean $s(i)$ of all points in a cluster is the measure of how closely grouped are all points in the cluster. Therefore, mean $s(i)$ of all data in the set, known as SC, is a measure of how adequate the clustering is.

Table 1 – Confusion matrix.

	p	n
Y	True Positives	False Positives
N	False Negatives	True Negatives

Source: (FAWCETT, 2006).

2.13.2 Confusion Matrix and Classification Accuracy

Considering a classification problem with two classes, an instance I can be mapped to a set object $\{p, n\}$ with positive and negative class labels, respectively. Mapping instances to predicted classes defines a *classification model*, or *classifier*. In order to distinguish between real and predicted classes, labels $\{Y, N\}$ are used for class predictions produced by a model. Given a classifier and an instance, there are four possible results: if the instance is positive and classified as positive, it counts as a *true positive*, and if classified as negative, it is a *false negative*; if the instance is negative and classified as negative, it is a *true negative*, and if classified as positive, it is a *false positive*. Given a classifier and an instances set (test set), a 2×2 *confusion matrix* (also called *contingency table*) can be constructed to represent the layout of instances. This matrix is the basis for many common metrics (FAWCETT, 2006).

Table 1 presents a confusion matrix, where columns (with sums P and N, respectively) represent true classes, and rows the predicted ones. Values in the main diagonal are correct choices and the others errors (or confusion) in classification. Some of the metrics that can be obtained from it are:

$$\text{fp rate} = \frac{FP}{N}, \tag{266}$$

$$\text{tp rate} = \frac{TP}{P}, \tag{267}$$

$$\text{fn rate} = \frac{FN}{P}, \tag{268}$$

$$\text{tn rate} = \frac{TN}{N}, \tag{269}$$

$$\text{precision} = \frac{TP}{TP + FP}, \tag{270}$$

$$\text{accuracy} = \frac{TP + TN}{P + N}. \tag{271}$$

True positive rate is also called *hit*, *recall*, or *sensitivity*, and false positives *false alarm* or *fall-out*.

2.13.3 Friedman Test

The Friedman test (FRIEDMAN, 1937 apud DEMŠAR, 2006) ranks the algorithms for each data set separately, the best performing algorithm getting the rank of 1, the second

best rank 2, and so forth. In case of ties, average ranks are assigned. Let r_i^j be the rank of the j-th algorithms on the i-th of N data sets. The Friedman test compares the average ranks of algorithms, $R_j = \frac{1}{N}\sum_i r_i^j$. Under the null-hypothesis, which states that all the algorithms are equivalent and so their ranks R_j should be equal, the Friedman statistic:

$$\chi_F^2 = \frac{12N}{k\left(k+1\right)}\left[\sum_j R_j^2 - \frac{k\left(k+1\right)^2}{4}\right] \tag{272}$$

is distributed according to χ_F^2 with $k-1$ degrees of freedom, when N and k are big enough (as a rule of a thumb, $M > 10$ and $k > 5$). This statistic, however, was considered undesirably conservative, and another was derived (DEMŠAR, 2006):

$$F_F = \frac{\left(N-1\right)\chi_F^2}{N\left(k-1\right) - \chi_F^2}, \tag{273}$$

which is distributed according to the F-distribution with $k-1$ and $\left(k-1\right)\left(N-1\right)$ degrees of freedom.

2.13.4 Nemenyi Test

If the null-hypothesis is rejected, one can proceed with a post-hoc test. The Nemenyi test (NEMENYI, 1963 apud DEMŠAR, 2006) is used when all classifiers are compared to one another. The performance of two classifiers is significantly different if the corresponding average ranks differ by at least the critical difference:

$$CD = q_\alpha\sqrt{\frac{k\left(k+1\right)}{6N}}, \tag{274}$$

where critical values q_α are based on the Studentised range statistic divided by $\sqrt{2}$.

Chapter 3

Proposed Methods

In this chapter alternatives to the usage of shortest path to minimise distances between points in G are proposed. Particularly, by using information theory concepts, the adjacency matrix can be treated as a probability distribution for a random variable. Thus, by minimising its distance values, more accurate and computationally efficient results than those found by differential and integral functions can be obtained. In general, the goal is to use information theory-based measures (such as KL-divergence, Bhattacharyya, Hellinger, and Cauchy-Schwarz) to construct modified versions of these methods, using a multivariate parametric model for each class involved in the classification problem.

Using these measures may provide a dataset more adequate for the application of those methods than the initial one, as they are based on probability distributions for the occurrence of each datum. Thus, the algorithms are modified to provide possibly more relevant results, observing the relationship between each data class and its impact on the overall set. Further details about these methods are presented next.

3.1 Extending Manifold Learning Methods Using Stochastic Distances

An initial approach to this problem is to replace the Euclidean distance used in regular ML methods with the aforementioned stochastic distances. This section describes in general terms how these methods can be altered to include these measures in an attempt to improve their performance.

3.1.1 Isometric Feature Mapping

For the ISOMAP algorithm, proposed methodology consists of modifying step 2 presented in section 2.7, including stochastic distances in the estimation of geodesic distances matrix $D_G = \{d_G(i,j)\}$. To do so, initially one investigates the listed stochastic distances in case where classes are mapped to a multivariate Gaussian distribution, with the new distances matrix expressed as:

$$D_G = \{\alpha d_G(i,j) + (1-\alpha)\, d_E(i,j)\}, \tag{275}$$

that is, a convex combination of geodesic and stochastic distances between sample classes i and j. It is also possible to assess the possibility of adapting the ES-ISOMAP method using stochastic distances in the dissimilarity matrix (equation (130)). This way, one expects to penalise neighbouring samples that belong to different classes.

3.1.2 Locally Linear Embedding

To extend the LLE algorithm, a modification to the calculation of the distances matrix defined in SLLE (equation (168)) is proposed. Including stochastic distances, the new matrix is:

$$\Delta' = \Delta + \alpha D_{ij}^E \Lambda_{ij}, \tag{276}$$

where D_{ij}^E is the stochastic distance between sample classes i and j, with $\Lambda_{ij} = 0$ if x_i and x_j belong to the same class, and $\Lambda_{ij} = 1$ if they are in distinct classes. Using stochastic distances, one expects to penalise samples belonging to different classes, increasing dissimilarity between them.

3.1.3 Laplacian Eigenmaps

One way to adapt the LE algorithm would be altering the edge weights inserting a term based on a stochastic distance. Therefore, equation (169), that can be written as:

$$W_{ij} = \exp\left(-\frac{\|x_i - x_j\|^2}{t}\right) = \exp\left(-D_G(x_i, x_j)\right), \tag{277}$$

where D_G is the Gaussian distance between x_i and x_j, becomes:

$$W_{ij} = \exp\left(-\left(D_G(x_i, x_j) + D_E(\omega_i, \omega_j)\right)\right), \tag{278}$$

being D_E the stochastic distance, ω_i and ω_j classes in samples i and j, respectively. If i and j are in the same class, $D_E(\omega_i, \omega_j) = 0$ and the weight is unchanged. If they are in different classes, $D_E(\omega_i, \omega_j) > 0$, reducing similarity. Different distances can also be conditioned to a parameter in a manner that results in the convex combination:

$$W_{ij} = \exp\left(-\left(\alpha D_G(x_i, x_j) + (1-\alpha)\, D_E(\omega_i, \omega_j)\right)\right), \alpha \in [0,1]. \tag{279}$$

Moreover, other distributions can be used, such as multivariate Student's t, for which new distance formulas for the proper stochastic distances would need to be derived.

3.2 Entropic Isometric Feature Mapping

A limitation of high dimensionality data analysis is the weak discriminating power of the Euclidean metric. It has been shown that, as the number of features increases, the degree of contrast provided by regular Euclidean distance becomes insufficient (LEE; VERLEYSEN, 2007). A variation of ISOMAP defined in terms of relative entropy, or KL-divergence, is, then, proposed. The main goal is to replace pairwise distances matrix D, obtained from the KNN graph with edges weighed by Euclidean distances, by the entropic distances matrix E, obtained from the KNN graph with edges weighed by symmetrised KL-divergence between multivariate Gaussian distributions estimated from local patches. It is known that KL-divergence (given by equation (31)) is not symmetric, therefore, symmetrised KL-divergence must be calculated by:

$$D_{KL}^{\text{sym}}(p,q) = \frac{1}{2}\left[D_{KL}(p,q) + D_{KL}(q,p)\right], \tag{280}$$

which has as closed form expression:

$$D_{KL}^{\text{sym}}(p,q) = \frac{1}{2}\left[\frac{1}{2}\operatorname{Tr}\left[\Sigma_1^{-1}\Sigma_2 + \Sigma_2^{-1}\Sigma_1\right] + \frac{1}{2}\left(\vec{\mu}_1 - \vec{\mu}_2\right)^T \Sigma_1^{-1}\left(\vec{\mu}_1 - \vec{\mu}_2\right) + \frac{1}{2}\left(\vec{\mu}_2 - \vec{\mu}_1\right)^T \Sigma_2^{-1}\left(\vec{\mu}_2 - \vec{\mu}_1\right) - d\right]. \tag{281}$$

The input matrix is denoted by $X = \{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n\}, \vec{x}_i \in \mathbb{R}^m$. A KNN graph $G = (V, E)$, with $|V| = n$, can be constructed, connecting each sample $\vec{x}_i$ with its KNN. Since the neighbourhood can be approximated by a linear patch, this step uses Euclidean distance as similarity. Let a patch P_i and set $\{\vec{x}_i\} \cup \{\vec{x}_j \in N(i)\}$, where $N(i)$ is the set of the neighbourhood of $\vec{x}_i$, patch matrix P_i can be defined by:

$$P_i = \left[\vec{x}_{i1}, \vec{x}_{i2}, \ldots, \vec{x}_{ik}\right], \tag{282}$$

to denote data matrix $d \times (k+1)$ which makes up the i-th patch. It is assumed that P_i is a random sample of a multivariate Gaussian distribution of size k. Therefore, estimators for maximum likelihood of model parameters can be calculated as:

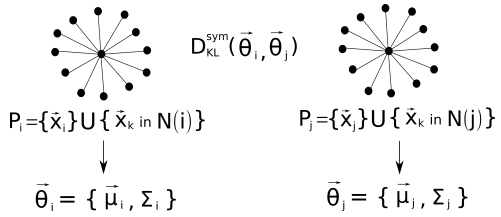
$$\vec{\mu}_i = \frac{1}{(k+1)}\sum_{j=1}^{k+1}\vec{x}_{ij}, \tag{283}$$

$$\Sigma_i = \frac{1}{k}\sum_{j=1}^{k+1}\left(\vec{x}_{ij} - \vec{\mu}_i\right)\left(\vec{x}_{ij} - \vec{\mu}_i\right)^T. \tag{284}$$

Figure 3 illustrates the mapping of local patches in the KNN graph to a parametric representation.

The next step is constructing entropic KNN graph (or KL-KNN graph), as a substitute for the traditional KNN graph used in ISOMAP. Basically, this is done by updating edges' weights in the KNN graph. Instead of Euclidean distances between vectors $\vec{x}_i$ and $\vec{x}_j$,

Figure 3 – Mapping from a patch P_i on the graph to a parametric feature vector.



Source: (CERVATI NETO; LEVADA, 2020).

symmetrised KL-divergence D_{KL}^{sym} between respective patches P_i and P_j is calculated using equation (281). It is noted that $D_{KL}^{\text{sym}}(P_i, P_j)$ is a patch-based similarity measure, which means it is less sensitive to the presence of outliers and noise in data than pairwise Euclidean distance, used by the traditional ISOMAP algorithm (CERVATI NETO; LEVADA, 2020).

Given an entropic KNN graph, computing geodesic distances follows through shortest pairwise paths. By the end of this process, Entropic Isometric Feature Mapping (ISOMAP-KL) produces entropic distances matrix E, which replaces ISOMAP's pairwise distances matrix. The following steps are the same as ISOMAP's, that is, from entropic distances matrix E, the Gram matrix of inner products B is calculated and, through spectral decomposition, come the principal eigenvectors.

3.3 Supervised t-Distributed Stochastic Neighbour Embedding for Metric Learning using Stochastic and Geodesic Distances

In this section, two extensions of the t-SNE are proposed, namely the Supervised Geodesic t-Distributed Stochastic Neighbour Embedding (SGt-SNE) and Supervised Entropic t-Distributed Stochastic Neighbour Embedding (SEt-SNE). Their distinction relies on how the graph edges are weighed prior to the computation of the pairwise probabilities employed in the iterative minimisation of the KL-divergence between the input graph and its low dimension representation. The objective is to enlarge the cost of edges connecting samples from different classes. This intends to penalise such samples, while preventing them from remaining close in the low dimension representation.

3.3.1 Supervised Geodesic t-Distributed Stochastic Neighbour Embedding

In SGt-SNE, the Euclidean distances $\|\vec{x}_i - \vec{x}_j\|^2$ and $\|\vec{y}_i - \vec{y}_j\|^2$ employed in the computation of the pairwise probabilities p_{ij} and q_{ij} are replaced by geodesic distances between samples in the KNN graph in the input space and low dimension spaces, respectively. A

rule of thumb is to set the number of neighbours $k = \sqrt{n}$ in order to create the KNN based on data, where n is the number of samples. The Floyd-Warshall algorithm is employed in the KNN graph to capture the pairwise distance matrix, which utilises dynamic programming to solve the all-pairs shortest path problem. Algorithm 1 details a pseudocode for the Floyd-Warshall method (CORMEN et al., 2009).

Algorithm 1 Floyd-Warshall for pairwise distances matrix.

```
FLOYD-WARSHALL(W)
    n = W.rows    //W is the adjacency matrix
    D^(0) = W
    for k = 1 to n
        Let D^(k) = (d_ij^(k)) be a new matrix
        for i = 1 to n
            for j = 1 to n
                d_ij^(k) = min(d_ij^(k-1), d_ik^(k-1) + d_kj^(k-1))
    return D^(n)
```

In the first supervised strategy, following the construction of the pairwise geodesic distances matrix, the class labels are employed to calculate a final distance matrix. The rationale relies on the fact that penalising pairwise distances between samples belonging to different classes should improve class separability. Algorithm 2 details the pseudocode for the function that learns the pairwise geodesic distance matrix in SGt-SNE as a substitute for the regular Euclidean distances matrix in the t-SNE. It is worth mentioning that increasing the distance between neighbouring samples of different classes should impose they remain apart in the low dimension space. Hence, the clusters in such an output space are expected to contain reduced overlaps compared to the ones produced through a standard t-SNE, enhancing the overall classification performance.

Algorithm 2 Distance learning in SGt-SNE.

```
GEODESIC-DISTANCE(X, y)
    Compute the pairwise Euclidean distance matrix D.
    Build the KNN graph from X to yield W.
    Compute the pairwise geodesic distance matrix G.
    for i = 1 to n
        for j = 1 to n
            if W[i, j] > 0
                if y[i] ≠ y[j]
                    G[i, j] = G[i, j] + D[i, j]
return G
```

Algorithm 3 describes the pseudocode for SGt-SNE. The calculation of the pairwise probabilities p_{ij} and q_{ij} is dependent on the modified geodesic distance matrix G generated in algorithm 2. The geodesic matrix calculated in the input space is represented as G_x, and G_y being the geodesic matrix calculated in the low dimension space.

Algorithm 3 Supervised Geodesic t-Distributed Stochastic Neighbour Embedding.

SGt-SNE$(X, y, \text{Perp}, T, \eta, \alpha(t))$
  Compute the geodesic distance matrix G_x.
  Compute pairwise probabilities p_{ij} with equation (223) using G_x.
  Sample initial solution $\mathcal{Y}^{(0)}$ from $\mathcal{N}(0, 10^{-4}I)$.
  For $t = 1$ to T:
    Compute the geodesic distance matrix $G_{\mathcal{Y}}$.
    Compute low dimensional affinities q_{ij} with equation (224).
    Compute the gradient using equation (232).
    Update the coordinates with equation (209).
return $\mathcal{Y}$

3.3.2 Supervised Entropic t-Distributed Stochastic Neighbour Embedding

The Euclidean distance consists of a pointwise similarity measure and, as such, it may be highly sensitive to data noise or outliers (SHIRKHORSHIDI; AGHABOZORGI; WAH, 2015). This may critically affect the performance of classification tasks (AGGARWAL; HINNEBURG; KEIM, 2001).

A core characteristic of SEt-SNE is to substitute the geodesic distances matrix by a stochastic distances matrix. Nevertheless, stochastic distances are computed by local multivariate Gaussian densities estimated from a neighbourhood of the KNN graph. Thus, for each patch P_i in the graph, the sample mean vector $\vec{\mu}_i$ and the sample covariance matrix Σ_i are estimated. The adoption of KL-divergence, Bhattacharyya distance, or Cauchy-Schwarz divergence is introduced due to the fact that they have a closed form expression for multivariate Gaussian random variables.

The KL-divergence between two multivariate Gaussian densities may be represented as equation (31). The Bhattacharyya distance consists of a similarity measure between two densities, calculated using the Bhattacharyya coefficient, quantifying the overlapping size between two random samples (BHATTACHARYYA, 1943). The Bhattacharyya coefficient is proportional to the area of intersection between two pdfs, as shown by equation (38). A generalisation of the Shannon entropy is the Rényi entropy of order α. Thus, one may generalise the KL-divergence through the Rényi entropy:

$$D_R^{\alpha}(p, q) = \frac{1}{\alpha - 1} \log \int \frac{p(x)^{\alpha}}{q(x)^{\alpha - 1}} \, \mathrm{d}x. \tag{285}$$

For $\alpha = 2$, the quadratic entropy that leads to the Cauchy-Schwarz divergence is given by (NIELSEN; SUN; MARCHAND-MAILLET, 2017):

$$D_{CS}(p, q) = -\log \frac{\int p(x) \, q(x) \, \mathrm{d}x}{\sqrt{\int p(x)^2 \, \mathrm{d}x \int q(x)^2 \, \mathrm{d}x}}. \tag{286}$$

In the multivariate Gaussian case, the Cauchy-Schwarz divergence may be represented through the following closed form expression (NIELSEN; SUN; MARCHAND-MAILLET,

2017):

$$D_{CS}(p,q) = \frac{1}{2}\vec{\mu}_p^T\Sigma_p^{-1}\vec{\mu}_p + \frac{1}{2}\vec{\mu}_q^T\Sigma_q^{-1}\vec{\mu}_q + \frac{1}{4}\log\left|\frac{\Sigma_q}{2}\right| + \frac{1}{4}\log\left|\frac{\Sigma_p}{2}\right| + \frac{1}{2}\log\left|\Sigma_p^{-1}+\Sigma_q^{-1}\right| - \frac{1}{2}\left(\Sigma_p^{-1}\vec{\mu}_p + \Sigma_q^{-1}\vec{\mu}_q\right)^T\left(\Sigma_p^{-1}+\Sigma_q^{-1}\right)^{-1}\left(\Sigma_p^{-1}\vec{\mu}_p + \Sigma_q^{-1}\vec{\mu}_q\right). \quad (287)$$

Algorithm 4 details the pseudocode for the function which learns the pairwise stochastic distance matrix in SEt-SNE as a substitute for the standard Euclidean distance matrix in the t-SNE.

Algorithm 4 Distance learning in SEt-SNE.

ENTROPIC-DISTANCE(X, y)
  Compute the pairwise Euclidean distance matrix D.
  Build the KNN graph from X to yield W.
  Estimate the local means and covariance matrices.
  Compute the pairwise entropic distance matrix E.
  for $i = 1$ to n
    for $j = 1$ to n
      if $W[i,j] > 0$
        if $y[i] = y[j]$
          $D[i,j] = \min(D[i,j], E[i,j])$
        else
          $D[i,j] = D[i,j] + E[i,j]$
return G

Similarly to the geodesic framework, the main objective is to penalise neighbouring edges connecting samples of different classes. Therefore, algorithm 5 represents the pseudocode of SEt-SNE. The entropic distances matrix in the input space is denoted as as E_x, and $E_{\mathcal{Y}}$ refers to the entropic distances matrix in the low dimension space.

Algorithm 5 Supervised Entropic t-Distributed Stochastic Neighbour Embedding.

SEt-SNE($X, y, \text{Perp}, T, \eta, \alpha(t)$)
  Compute the entropic distance matrix E_x.
  Compute pairwise probabilities p_{ij} with equation (223) using E_x.
  Sample initial solution $\mathcal{Y}^{(0)}$ from $\mathcal{N}(0, 10^{-4}I)$.
  for $t = 1$ to T
    Compute the entropic distance matrix $E_{\mathcal{Y}}$.
    Compute low dimensional affinities q_{ij} with equation (224).
    Compute the gradient using equation (232).
    Update the coordinates with equation (209).
return $\mathcal{Y}$

3.4 Probabilistic Nearest Neighbours-Based Locality Preserving Projections

The main goal of the proposed Probabilistic Nearest Neighbours-Based Locality Preserving Projections (PNN-LPP) method is to combine the PNN graph construction method to approximate the underlying data manifold in LPP to perform DR-based unsupervised metric learning. The goal of PNN-LPP is to overcome two known limitations of regular LPP:

1. To make it less sensitive to noise and outliers in data, through the replacement of the Euclidean distance by a probabilistic measure;

2. To improve the performance of regular LPP in small sample size problems (when n is limited).

Algorithm 6 shows a summary of the proposed PNN-LPP for DR-based unsupervised metric learning. Basically, the method has four parameters: data matrix X, number of neighbours K, dimension of the output space d, and variance of the Gaussian kernel t.

Algorithm 6 PNN-based LPP.

PNN-LPP(X, K, d, t)
For each sample $\vec{x}_i$ of the dataset:
Compute the distances to all other samples $\vec{x}_j$, storing them in the vector $\vec{d}_i$
Use a Gaussian Kernel to define the edge weights D.

$$D_{ij} = \begin{cases} \exp\left(-\frac{d_{ij}}{t}\right), & \text{if } v_j \in N(v_i) \\ 0, & \text{if } v_j \notin N(v_i) \end{cases} \tag{288}$$

Sort the distances in D in ascending order
Compute the edge weights as:

$$s_{ij} = \frac{d_{ik+1} - d_{ij}}{d_{ik+1} - d_{i1}} \tag{289}$$

Compute the diagonal matrix D with degrees d_i for $i = 1, 2, \ldots, n$.
Compute the Laplacian matrix $L = D - W$
Select the least d eigenvectors of the matrix $\left(XDX^T\right)^{-1} XLX^T$ to build A.
Project the data using matrix A: $Y = A^T X$
return Y

3.5 Kernel Density Estimation-based Isometric Feature Mapping

The Parametric PCA metric learning algorithm, which computes the entropic covariance matrix of the data using information-theoretic divergences between densities estimated in local patches along the neighbourhood graph (LEVADA, 2020), was the main inspiration for the Kernel Density Estimation-based Isometric Feature Mapping (KDE-ISOMAP).

The primary distinction between ISOMAP and KDE-ISOMAP is in the first phase of the algorithm. Let $X = \{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n\}$ be the data matrix, where each column $\vec{x}_i \in \mathbb{R}^m$, represents an observation, and by $G = (V, E)$ the ϵ-neighbourhood graph induced from X by creating an edge between each pair of samples $\vec{x}_i$ and $\vec{x}_j$ if $d_E(\vec{x}_i, \vec{x}_j) < \epsilon$, where $d_E(.,.)$ is the regular Euclidean distance. As a result, a patch P_i may be defined as the set formed by a sample $\vec{x}_i$ and its neighbourhood, which, for a sufficiently high sample density, belong to a single Euclidean subspace (ROWEIS; SAUL, 2000). Let k_i be the number of $\vec{x}_i$ neighbours in G. The patch P_i is thus a $(k+1) \times m$ matrix:

$$P_i = \begin{bmatrix} \vec{x}_i \\ \vec{x}_{i1} \\ \vec{x}_{i2} \\ \vdots \\ \vec{x}_{ik_i} \end{bmatrix} = \begin{bmatrix} \vec{x}_i(1) & \vec{x}_i(2) & \ldots & \vec{x}_i(m) \\ \vec{x}_{i1}(1) & \vec{x}_{i1}(2) & \ldots & \vec{x}_{i1}(m) \\ \vec{x}_{i2}(1) & \vec{x}_{i2}(2) & \ldots & \vec{x}_{i2}(m) \\ \vdots & \vdots & \ddots & \vdots \\ \vec{x}_{ik_i}(1) & \vec{x}_{ik_i}(2) & \ldots & \vec{x}_{ik_i}(m) \end{bmatrix}. \tag{290}$$

The k_i neighbours of $\vec{x}_i$ in the ϵ-neighbourhood network are denoted by $\{\vec{x}_{i1}, \vec{x}_{i2}, \ldots, \vec{x}_{ik_i}\}$. Let each column of the matrix P_i be a sample of a 1D random variable x_k with a pdf $f(x_k)$, these are estimated for $k = 1, 2, \ldots, m$ for each patch of the ϵ-neighbourhood graph using KDE, a non-parametric approach. Because each patch includes m 1D densities and the graph has n patches, the total number of non-parametric densities is nm, making the computational cost of the proposed KDE-ISOMAP $O(n^3 m)$.

The relative entropies (KL-divergences) between the densities predicted for each pair of nearby patches P_i and P_j are used to replace the pointwise Euclidean distances in the edges $(\vec{x}_i, \vec{x}_j) \in E$. This is known as the entropic ϵ-neighbourhood graph. Having precisely m pairings of 1D densities, there are m KL-divergences to compute for each pair of patches P_i and P_j.

The KL-divergence of distributions $\vec{p} = [p_1, p_2, \ldots, p_L]$ and $\vec{q} = [q_1, q_2, \ldots, q_L]$, where L is the number of points (bins) employed in KDE, may be calculated as follows:

$$D_{KL}(\vec{p}, \vec{q}) = \frac{1}{L} \sum_{i=1}^{L} p_i \log\left(\frac{p_i}{q_i}\right). \tag{291}$$

The symmetrised KL-divergence can be calculated as:

$$D_s\left(\vec{p},\vec{q}\right) = \frac{1}{2}\left(D_{KL}(\vec{p},\vec{q}) + D_{KL}\left(\vec{q},\vec{p}\right)\right) = \frac{1}{2}\left(H\left(\vec{p},\vec{q}\right) - H\left(\vec{p}\right) + H\left(\vec{q},\vec{p}\right) - H\left(\vec{q}\right)\right)$$
$$= \frac{1}{2}\left(H\left(\vec{p},\vec{q}\right) + H\left(\vec{q},\vec{p}\right)\right) - \frac{1}{2}\left(H\left(\vec{p}\right) + H\left(\vec{q}\right)\right), \quad (292)$$

that is, the average of the cross-entropies minus the average of the individual entropies. A vector of relative entropies $\vec{\Psi}_{ij}$ is built after computing the KL-divergences between the m pairings of 1D densities in P_i and P_j:

$$\vec{\Psi}_{ij} = \left[D_s\left(\vec{p}_1,\vec{q}_1\right), D_s\left(\vec{p}_2,\vec{q}_2\right), \ldots, D_s\left(\vec{p}_m,\vec{q}_m\right)\right]. \quad (293)$$

Finally, the weight of the edge $(\vec{x}_i,\vec{x}_j) \in E$ is replaced by:

$$w_{ij} = \vec{\Psi}_{ij}^T\vec{\Psi}_{ij} = \left\|\vec{\Psi}_{ij}\right\|^2, \quad (294)$$

which leads to the entropic neighbourhood graph. KDE-ISOMAP's second and third phases are identical to those of regular ISOMAP.

Chapter 4

Experiments and Results

For the analysis of results, a both qualitative, through dispersion graphs to visualise transformed data in 2D cases, and a quantitative approach, based on accuracy metrics of different methods for supervised classification, such as Support Vector Machines (SVM), KNN, Bayesian classifiers under Gaussian hypothesis, among others, are used. It is expected that supervising non-linear DR methods using information theory leads to a better classification performance when compared to the original versions of the proposed methods. These are tested on datasets containing real world data, retrieved from `openml.org`, with Table 2 having specific information about each of them, including the number of samples, features, and classes. In all computational experiments, 50% of the samples are chosen for training and 50% for testing. Finally, non-parametric hypothesis tests are employed to determine if including information theory-based measures can significantly improve classification performance.

4.1 Entropic Isometric Feature Mapping

In order to test and evaluate the proposed method for unsupervised metric learning in classification tasks, its performance was compared against traditional PCA, KPCA, ISOMAP, LLE, and LE in several public datasets available at `www.openml.org`. It is worth mentioning that the selected datasets have significant variations in the number of samples and features, as well as different numbers of classes. In the first set of experiments, an internal index was used to assess the quality of clusters obtained after the unsupervised metric learning provided by different DR methods using SC.

Table 3 shows the obtained results for 20 different datasets, where column *ISOKL* denotes the proposed parametric method under multivariate Gaussian hypothesis. It is

Table 2 – Number of samples, features, and classes of the used openML datasets.

Dataset	Samples	Features	Classes	Dataset	Samples	Features	Classes
SPECTF	349	44	2	first-order-theorem	2000	51	6
veteran	137	7	2	car	1728	7	4
sleuth_ex1605	62	5	2	tae	151	6	2
AIDS	50	4	2	transplant	131	4	2
cloud	108	7	2	servo	167	5	2
FL2000	67	15	5	mu284	284	10	2
analcatdata_creditscore	100	6	2	triazines	186	60	2
corral	160	6	2	page-blocks	5473	10	2
cars1	392	7	3	arsenic-male-lung	559	3	2
LED-display-domain-7digit	500	7	10	diggle_table_a2	310	8	9
hayes-roth	160	4	3	rmftsa_ladata	508	11	0
Diabetes130US (1%)	1017	49	3	prnn_crabs	200	6	2
blogger	100	5	2	parity5	32	0	2
user-knowledge	403	5	5	bolts	40	7	2
rabe_131	50	5	2	threeOf9	512	10	2
haberman	306	3	2	fri_c3_100_5	100	5	2
prnn_synth	250	2	2	baskball	96	4	2
visualizing_environmental	111	3	2	newton_hema	140	2	2
vineyard	52	2	2	strikes	625	6	2
monks-problems-1	566	6	2	datatrieve	130	8	2
acute-inflammations	120	6	2	prnn_fglass	214	9	2
planning-relax	182	12	2	pwLinear	200	10	2
sensory	576	11	2	breast-cancer	286	10	2
auto_price	159	15	2	backache	180	5	2
wisconsin	194	32	2	heart-statlog	270	13	2
fri_c4_250_100	250	100	2	balance_scale	625	4	3
thoracic_surgery	470	16	2	mux6	128	7	2
conference_attendance	246	6	2	pm10	500	7	2
analcatdata_boxing1	120	3	2	disclosure_z	662	4	0
fri_c2_100_10	100	10	2	KnuggetChase3	194	39	2
lupus	87	3	2	breast-tissue	106	9	4
fruitfly	125	4	2	Engine1	383	5	3
iris	150	4	3	diabetes_numeric	43	3	0
wine	178	13	2	parkinsons	195	23	2
mfeat-fourier	2000	76	2	prnn_viruses	61	10	4
texture	5500	40	11	confidence	72	3	2
satimage	6430	36	6	plasma_retinol	315	11	0

Source: `openml.org`.

worth mentioning that the definition of the parameter k (patch size) plays an important role in the proposed ISOMAP-KL. The method is sensitive to variations on this parameter, which essentially controls the patch size. Different values of k can lead to significantly different classification results. In all experiments, a simple heuristic was performed: to evaluate the classification accuracy for the all values of k ranging from 10 to 200, using an increment of 10 units. In other words, were considered as candidates the values of k belonging to the interval $S = [10, 20, 30, 40, \ldots, 100, \ldots, 200]$. The best result was then selected for each dataset. An intuition behind this choice is that a low k is usually preferred in small datasets, to preserve locality of the patches. Ideally, one should keep in mind the trade-off between locality preservation, which means choosing a small k, and having a large enough sample size for suitable parameter estimation.

The results suggest that, on average, the proposed ISOMAP-KL is more efficient in building a meaningful representation in terms of the consistency within clusters of data than the other methods for these datasets. Moreover, it should be noted that, in 12 of 20 datasets, ISOMAP-KL obtained the highest SC, that is, in 60% of the cases,

Table 3 – Silhouette coefficients for clusters produced by PCA, KPCA, ISOMAP, LLE, LE, and ISOMAP-KL for several datasets from OpenML.org (2D case).

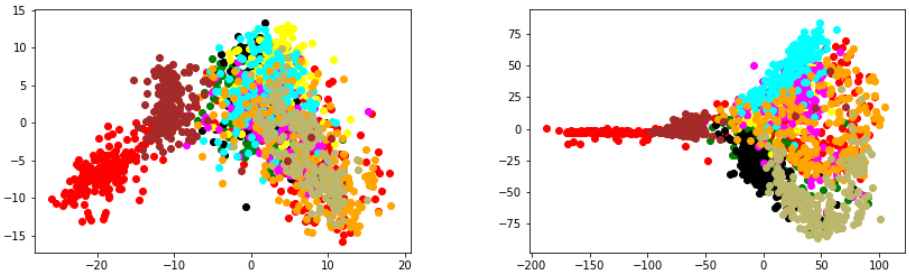
	PCA	KPCA	ISO	LLE	LE	*ISOKL*
iris	0.401	0.469	0.423	0.297	0.539	**0.576**
wine	0.526	0.610	0.533	0.140	**0.728**	0.656
mfeat-fourier	0.000	0.011	0.016	−0.073	−0.006	**0.145**
texture	−0.058	−0.050	0.086	0.068	0.245	**0.348**
satimage	0.219	0.247	0.232	0.037	0.233	**0.349**
theorem	−0.168	−0.105	**−0.099**	−0.113	−0.466	−0.156
synthetic	0.346	0.459	0.361	0.146	0.501	**0.512**
car	−0.110	−0.129	−0.075	**0.000**	−0.111	−0.034
tae	−0.059	**−0.004**	−0.069	−0.017	−0.019	−0.118
transplant	0.485	0.436	0.483	0.407	0.439	**0.582**
hayes	−0.023	0.038	−0.020	0.085	−0.012	**0.234**
SPECTF	−0.018	0.093	−0.028	−0.083	0.046	**0.106**
servo	**0.121**	0.105	0.102	0.097	0.085	0.034
mu284	0.301	0.338	0.288	**0.346**	0.306	0.306
triazines	0.009	**0.064**	0.017	0.001	0.017	0.023
pageblock	0.419	0.218	**0.534**	0.402	0.299	0.450
male-lung	0.563	−0.182	0.676	0.629	0.019	**0.988**
retinol	−0.008	0.004	0.004	0.001	0.015	**0.038**
diggle	0.406	0.409	0.412	0.272	0.363	**0.430**
rmftsa	0.228	0.242	0.235	0.188	0.231	**0.258**
Average	0.179	0.164	0.206	0.142	0.173	**0.286**
Std. Dev.	0.236	0.229	0.243	**0.196**	0.271	0.292

Source: (CERVATI NETO; LEVADA, 2020).

the proposed method produced better defined clusters. To test if the differences are significant, a statistical test was performed to compare the different groups. According to a non-parametric Friedman test, there is strong evidences against the null hypothesis that there are no significant differences between the groups (p-value $= 7.49 \times 10^{-5}$) for a significance level $\alpha = 0.05$. According to a post-hoc Nemenyi test, for the same significance level, ISOMAP-KL produced significantly better clusters (in terms of SC) than PCA (p-value $= 4.15 \times 10^{-5}$), KPCA (p-value $= 0.0425$), ISOMAP (p-value $= 0.0201$), LLE (p-value $= 2.87 \times 10^{-5}$), and LE (p-value $= 0.0046$).

In the second set of experiments, the performance of the proposed method was compared against PCA, KPCA, ISOMAP, LLE, and LE in supervised classification. For this purpose, 8 different parametric and non-parametric classifiers were selected: KNN with $k = 7$ (kept constant to have a baseline against which all methods are compared; a value close to $\sqrt{n}$ was found, in experiments, to be a reasonable guess, so this was chosen to fit the smallest datasets), SVM (linear), Naive Bayes (NB), Decision Trees (DT), Quadratic Discriminant Analysis (QDA) under Gaussian hypothesis, Multilayer Perceptron (MLP) (with a hidden layer size of 100, activated using a logistic function, over up to 5000 iterations), Gaussian Process Classifier (GPC), and Random Forest Classifier (RFC). In all experiments, 50% of the samples were selected for training and 50% for testing. Table 4 shows the classification accuracies for several datasets after DR to 2D spaces. The results show that there is no method that is uniformly superior to all the other ones. However, looking at the average

Figure 4 – Scatter-plots of mfeat-fourier dataset for the 2D case: ISOMAP (left) versus ISOMAP-KL (right).



Source: (CERVATI NETO; LEVADA, 2020).

accuracy, the results are more conclusive. Table 5 shows the average and standard deviation of all accuracies for each DR algorithm. The results indicate that for these datasets, in average, the proposed parametric ISOMAP-KL outperformed all the other methods. A hypothesis test was also performed to check whether the differences are statistically significant. According to a non-parametric Friedman test, there are strong evidences for rejecting the null hypothesis that all DR methods are equivalent (p-value $= 1.12 \times 10^{-15}$) for a significant level $\alpha = 0.05$. According to a post-hoc Nemenyi test, ISOMAP-KL produced significantly better classification accuracies than PCA (p-value $= 3.11 \times 10^{-12}$), KPCA (p-value $= 10^{-10}$), LLE (p-value $= 10^{-19}$) and LE (p-value $= 10^{-19}$).

The obtained results emphasise that the proposed ISOMAP-KL is competitive with the existing DR algorithms, since, overall, it is capable of producing features that are more discriminant than those generated by PCA, KPCA, and some ML algorithms. In other words, it can be concluded that ISOMAP-KL is a viable option for unsupervised metric learning in pattern classification tasks. To illustrate how the proposed method is capable of producing better defined clusters, some scatter-plots for the two dimensional case are presented, comparing ISOMAP and ISOMAP-KL. Figure 4 and Figure 5 show the clusters for the mfeat-fourier and texture datasets. Note that the clusters produced by ISOMAP-KL show less overlapping, that is, they tend to be easier to discriminate by pattern classifiers.

4.2 Supervised t-Distributed Stochastic Neighbour Embedding

The computational experiments compare the classification performance of competing supervised methods after DR-based metric learning. Considering the following seven classifiers: KNN, NB, SVM, Quadratic Bayesian classifier (under the Gaussian hypothesis), DT, RFC, and GPC (THEODORIDIS; KOUTROUMBAS, 2008; DUDA; HART; STORK,

Table 4 – Supervised classification accuracy obtained by classifiers after PCA, KPCA, ISOMAP, LLE, LE, and ISOMAP-KL (2D case).

	PCA	KPCA	ISO	LLE	LE	*ISOKL*	PCA	KPCA	ISO	LLE	LE	*ISOKL*
	iris dataset ($k = 20$)						first-order-theorem dataset ($k = 40$)					
KNN	**0.960**	0.866	0.866	**0.960**	0.826	**0.960**	0.460	0.478	0.467	0.421	0.445	**0.511**
SVM	**0.946**	0.800	0.880	0.413	0.440	**0.946**	0.447	0.410	0.496	0.410	0.410	**0.521**
NB	0.906	0.826	0.826	0.906	0.866	**1.000**	0.410	0.410	0.413	0.418	0.138	**0.422**
DT	0.933	0.800	0.760	0.933	0.800	**0.960**	0.423	0.449	0.457	0.377	0.434	**0.498**
QDA	**0.946**	0.800	0.866	0.946	0.813	**0.946**	0.410	0.423	**0.430**	0.418	0.150	0.405
MLP	0.946	0.826	0.866	**0.960**	0.306	0.946	0.412	0.410	0.431	0.410	0.410	**0.457**
GPC	0.906	0.826	0.853	0.613	0.440	**0.946**	0.443	0.414	0.493	0.410	0.410	**0.512**
RFC	0.920	0.880	0.840	**0.960**	0.813	0.946	0.483	0.492	0.490	0.409	0.442	**0.516**
	wine dataset ($k = 40$)						hayes-roth dataset ($k = 15$)					
KNN	0.966	0.977	**0.988**	0.752	**0.988**	0.966	0.424	0.651	0.545	**0.833**	0.590	0.742
SVM	0.955	0.977	0.943	0.382	0.382	0.966	0.606	0.606	0.530	0.606	0.606	**0.742**
NB	0.943	**0.955**	**0.955**	0.730	**0.955**	0.943	0.606	0.560	0.606	0.636	0.606	**0.803**
DT	0.943	0.932	0.943	0.629	0.966	**0.977**	0.621	0.803	0.636	0.757	0.636	**0.818**
QDA	0.966	0.966	0.966	0.808	0.955	**0.977**	0.606	0.575	0.606	0.681	0.606	**0.833**
MLP	0.955	**0.988**	0.966	0.382	0.382	0.977	0.606	0.606	0.606	0.606	0.606	**0.848**
GPC	0.966	0.966	0.966	0.404	0.382	**0.988**	0.500	0.606	0.515	0.606	0.606	**0.818**
RFC	0.966	0.955	0.932	0.685	0.977	**0.988**	0.666	0.696	0.636	**0.803**	0.621	**0.803**
	mfeat-fourier dataset ($k = 40$)						SPECTF dataset ($k = 80$)					
KNN	0.415	0.435	0.398	0.454	0.451	**0.626**	0.771	0.754	0.685	0.725	0.731	**0.788**
SVM	0.424	0.382	0.401	0.088	0.088	**0.450**	0.742	0.742	0.714	0.742	0.742	**0.811**
NB	0.415	0.431	0.428	0.469	0.409	**0.576**	0.725	0.742	0.720	0.640	0.702	**0.754**
DT	0.366	0.400	0.351	0.405	0.405	**0.542**	0.782	0.754	**0.794**	0.714	0.760	**0.794**
QDA	0.436	0.459	0.439	0.482	0.428	**0.595**	0.742	0.742	0.720	0.605	0.742	**0.760**
MLP	0.433	0.450	0.430	0.370	0.088	**0.637**	0.742	**0.794**	0.771	0.742	0.742	0.771
GPC	0.428	0.410	0.406	0.179	0.088	**0.547**	0.754	**0.777**	0.691	0.742	0.742	0.765
RFC	0.389	0.430	0.370	0.427	0.448	**0.580**	0.794	0.731	0.771	0.742	0.777	**0.840**
	texture dataset ($k = 40$)						servo dataset ($k = 10$)					
KNN	0.583	0.543	0.712	0.460	0.622	**0.846**	0.761	0.750	0.821	0.797	0.750	**0.916**
SVM	0.579	0.469	0.730	0.083	0.083	**0.732**	0.797	0.750	0.750	0.750	0.750	**0.916**
NB	0.491	0.460	0.594	0.485	0.621	**0.800**	0.928	0.738	0.821	0.809	0.821	**0.905**
DT	0.485	0.470	0.646	0.408	0.545	**0.810**	**0.904**	0.738	0.773	0.690	0.797	0.845
QDA	0.541	0.461	0.661	0.705	0.760	**0.819**	**0.916**	0.714	0.821	0.809	0.809	**0.916**
MLP	0.568	0.419	0.714	0.474	0.087	**0.840**	0.821	0.809	0.821	0.750	0.750	**0.905**
GPC	0.578	0.463	0.732	0.304	0.083	**0.826**	0.833	0.750	0.821	0.750	0.750	**0.905**
RFC	0.538	0.522	0.704	0.408	0.559	**0.844**	**0.916**	0.773	0.785	0.726	0.821	0.845
	satimage dataset ($k = 200$)						page-blocks dataset ($k = 100$)					
KNN	0.826	0.800	0.837	0.621	0.835	**0.852**	0.921	0.928	0.952	0.932	0.941	**0.957**
SVM	0.835	0.792	0.852	0.230	0.230	**0.854**	0.925	0.925	0.953	0.900	0.900	**0.956**
NB	0.806	0.781	0.794	0.616	0.739	**0.835**	0.879	0.922	0.897	0.903	0.942	**0.942**
DT	0.779	0.753	0.780	0.534	0.798	**0.803**	0.889	0.916	0.940	0.904	0.924	**0.955**
QDA	0.827	0.787	**0.830**	0.622	0.822	0.827	0.892	0.924	0.901	0.926	**0.942**	0.941
MLP	0.828	0.788	0.840	0.604	0.230	**0.847**	0.904	0.911	**0.949**	0.920	0.900	0.948
GPC	0.837	0.778	0.845	0.372	0.230	**0.852**	0.923	0.924	0.950	0.900	0.900	**0.962**
RFC	0.818	0.799	0.829	0.605	0.831	**0.846**	0.919	0.931	0.955	0.927	0.943	**0.963**

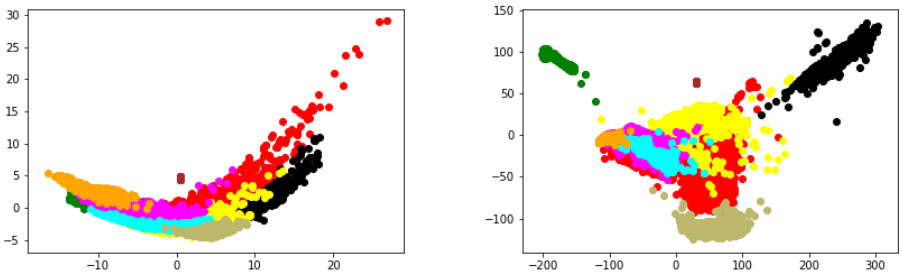
Source: (CERVATI NETO; LEVADA, 2020).

Table 5 – Average classification accuracies obtained by different classifiers for OpenML.org datasets in Table 4.

	PCA	KPCA	ISO	LLE	LE	*ISOKL*
Average	0.721	0.698	0.723	0.621	0.613	**0.807**
Std. Dev.	0.203	0.191	0.184	0.218	0.263	**0.160**

Source: (CERVATI NETO; LEVADA, 2020).

Figure 5 – Scatter-plots of texture dataset for the 2D case: ISOMAP (left) versus ISOMAP-KL (right).



Source: (CERVATI NETO; LEVADA, 2020).

2000; WEBB; COPSEY, 2011). Firstly, regular t-SNE, SGt-SNE, SEt-SNE, and Linear Discriminant Analysis (LDA) are applied to decrease the input data dimensionality to two. Subsequently, adopting a holdout strategy, half of the samples are employed to train the classifiers while the other half is utilised for testing.

To asses the results, Cohen's Kappa coefficient — which measures the level of agreement between two rankings of specialists — is calculated. The Kappa coefficient is preferable in comparison with accuracy due to the fact that it can tackle unbalanced data sets while removing the possibility of agreement between the classifier and a random guess. In this section, one ranking is provided by the true class labels (ground truth), while the remaining ranking is given by the output labels of a specific classifier. The Kappa coefficient may be computed directly from the confusion matrix as (HUDSON; RAMM, 1987):

$$\kappa = \frac{n \sum_{i=1}^{C} c_{ii} - \sum_{i=1}^{C} c_{i+} c_{+i}}{n^2 - \sum_{i=1}^{C} c_{i+} c_{+i}}, \tag{295}$$

where n is the number of samples, C refers to the number of classes (number of rows in the confusion matrix), c_{i+} denotes the sum of the i-th row of the confusion matrix, and c_{+i} represents the sum of the i-th column of the confusion matrix. Table 6 reports the results. Each value in the table refers to the maximum Kappa coefficient over the seven supervised classifiers used. The symmetrised KL-divergence is used in all experiments involving SEt-SNE, as reported in Table 6.

The results in Table 6 show a superior performance of the proposed SGt-SNE. The introduction of class labels into regular t-SNE resulted in a significant increase of the posterior supervised classification performance. A Friedman test was conducted, followed by a post-hoc Nemenyi test in order to check the results for a significance level $\alpha = 0.05$. Both SGt-SNE and SEt-SNE show significantly superior performance compared to regular t-SNE ($p < 10^{-3}$). Notwithstanding, the test indicates that there is no evidence that the proposed methods performed better in comparison with LDA.

Arguably, LDA is a benchmark for two-class supervised classification problems. However, a major limitation of LDA refers to the number of possible features that can be

Table 6 – Maximum Kappa coefficients obtained by KNN, SVM, NB, SVM, QDA, DT, RFC, and GPC classifiers after metric learning with t-SNE, LDA, SGt-SNE, and SEt-SNE.

Data sets	t-SNE	LDA	*SGt-SNE*	*SEt-SNE*
iris	0.960	0.979	**1.000**	**1.000**
digits	0.627	0.685	**0.820**	0.649
prnn_crabs	0.739	**1.000**	0.799	0.800
balance_scale	0.397	**0.829**	0.723	0.793
parity5	0.500	—	**1.000**	**1.000**
hayes-roth	0.409	**0.623**	0.448	0.478
rabe_131	0.834	**1.000**	**1.000**	**1.000**
servo	0.346	**0.936**	0.873	0.444
monks-problem-1	0.396	0.577	**0.849**	0.791
bolts	0.783	0.886	**1.000**	0.875
fri_c2_100_10	0.600	0.600	0.760	**0.880**
threeOf9	0.445	0.661	**1.000**	**1.000**
fri_c3_100_5	0.401	0.199	**0.633**	0.480
baskball	0.377	**0.706**	0.633	0.483
newton_hema	0.482	0.500	0.571	**0.626**
strikes	0.463	0.373	**0.738**	0.660
datatrieve	0.000	0.245	**0.469**	0.204
diggle_table_a2	0.948	**1.000**	0.961	0.974
fl2000	0.351	**0.751**	0.435	0.551
triazines	0.365	**0.675**	0.669	0.522
veteran	0.324	0.404	0.367	**0.607**
diabetes	0.335	**0.552**	0.438	0.461
car	0.179	0.459	**0.508**	0.362
prnn_fglass	0.328	0.386	0.508	**0.559**
creditscore	0.200	0.534	0.297	**0.580**
pwlinear	0.360	0.780	**0.940**	0.880
breast_cancer	0.907	0.961	0.976	**0.984**
wine	0.948	**1.000**	**1.000**	0.982
backache	0.370	0.587	**0.625**	0.110
heart-statlog	0.676	0.706	**0.734**	0.706
Average	0.502	0.676	**0.726**	0.681
Median	0.405	0.675	**0.736**	0.654
Max	0.960	**1.000**	**1.000**	**1.000**
Min	0.000	0.199	**0.297**	0.110

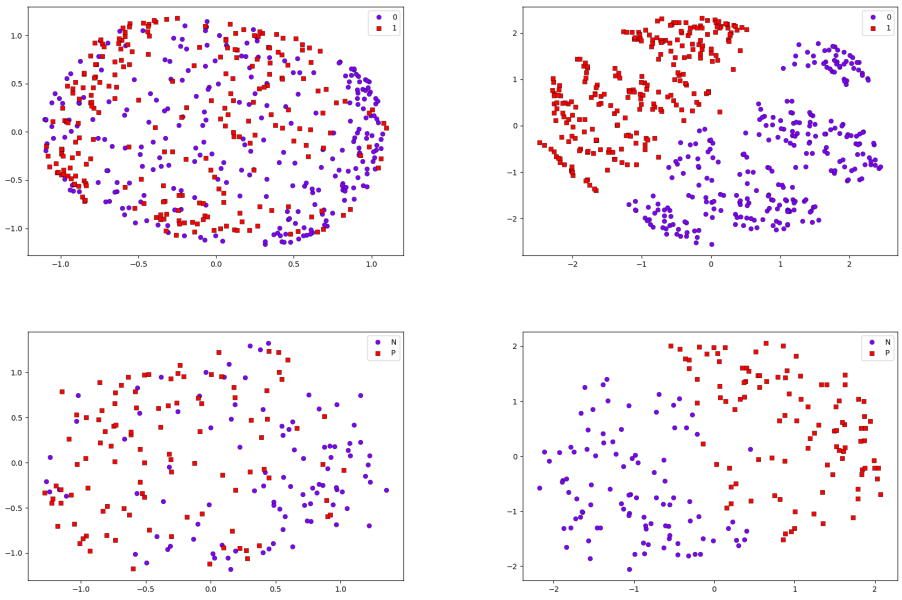
Source: the author.

Note: The parity5 set has no numeric features and, therefore, cannot be classified using LDA.

extracted from the input data. In a problem with C classes, the dimension of the output space is at most $C - 1$. Thus, when $C = 2$, LDA should extract at most one single feature. Therefore, an advantage of the proposed Supervised t-SNE methods over LDA is that the former are a viable alternative to LDA in such cases. For example, in the monks-problem-1 data set, which contains only two classes, the best Kappa subsequent to LDA is $\kappa = 0.577$, while the same performance measure after SGt-SNE equals $\kappa = 0.849$, being approximately 47% larger.

On the other hand, a caveat of the proposed methods relates to a comparatively higher computational cost. In SGt-SNE, the pairwise geodesic distance matrix is computed, with a complexity of $O(n^3)$ as a consequence of the Floyd-Warshall algorithm. Similarly, in SEt-SNE, the pairwise stochastic distance matrix is computed, which may be approximately

Figure 6 – Visual comparison between regular t-SNE (left) and Supervised Geodesic t-SNE (right) in the threeOf9 (top) and pwLinear (bottom) data sets.



Source: the author.

performed in $O(n^2m)$, where m represents the input data dimensionality. In both cases, the total computational cost for the DR process is larger compared to the regular t-SNE.

Figure 6 depicts scatter-plots of regular t-SNE and SGt-SNE involving the threeOf9 and pwLinear data sets. It is clear that the results of regular t-SNE show a considerably large clustering overlapping. In the supervised exercise, the class separability is appreciably more satisfactory, significantly improving the classification performance.

4.3 Probabilistic Nearest Neighbours-Based Locality Preserving Projections

A set of computational experiments was conducted to compare the average classification accuracies obtained by eight supervised classifiers (KNN, NB, SVM, DT, Bayesian classifier under Gaussian hypothesis, MLP, GPC, and RFC) after DR to 2D spaces in order to test and evaluate the proposed PNN-LPP method. An objective comparison of the proposed PNN-LPP against six unsupervised metric learning techniques based on DR: PCA, ISOMAP, LLE, LE, regular LPP, and UMAP.

It is widely known that state-of-the-art DR-based unsupervised metric learning algorithms, such as t-SNE and UMAP have excellent performance in large datasets, in which the density of points in the underlying data manifold is high. However, when the number of samples is somehow limited, lowering the density of points in the input

Table 7 – Average classification accuracies produced after dimensionality reduction based unsupervised metric learning with PCA, ISOMAP, LLE, LE, LPP, UMAP, and PNN-LPP for 32 openML datasets (2D case).

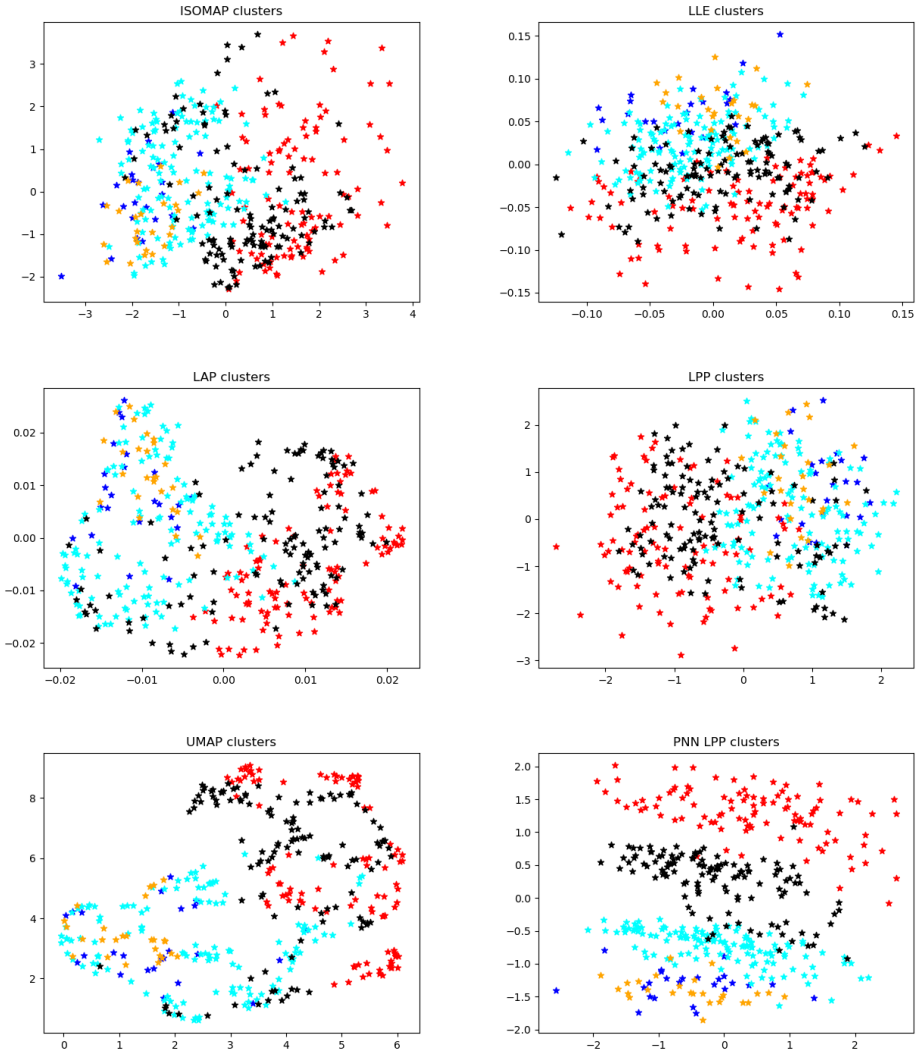
Dataset	PCA	ISOMAP	LLE	LE	LPP	UMAP	*PNN-LPP*
SPECTF	0.754	0.720	0.715	0.759	0.724	0.764	**0.794**
veteran	0.666	0.666	0.666	0.682	0.677	0.708	**0.735**
sleuth_ex1605	0.556	0.608	0.641	0.540	0.588	0.604	**0.734**
AIDS	0.374	0.355	0.370	0.415	0.360	0.410	**0.630**
cloud	0.657	0.643	0.664	0.625	0.634	0.655	**0.680**
FL2000	0.632	0.643	0.562	0.562	0.654	0.610	**0.691**
analcatdata_creditscore	0.795	0.795	0.730	0.750	0.737	0.757	**0.825**
corral	0.829	0.814	0.826	0.751	0.853	0.810	**0.911**
cars1	0.684	0.700	0.658	0.648	0.690	0.689	**0.705**
LED-display-domain-7digit	0.574	0.553	0.337	0.361	0.546	0.585	**0.599**
hayes-roth	0.615	0.476	0.479	0.403	0.465	0.440	**0.633**
Diabetes130US (1%)	0.523	0.525	0.527	0.529	0.534	0.526	**0.565**
blogger	0.679	0.667	0.707	0.600	0.697	0.622	**0.795**
user-knowledge	0.505	0.581	0.448	0.450	0.518	0.628	**0.785**
rabe_131	0.740	0.910	0.865	0.815	0.899	0.904	**0.940**
haberman	0.750	0.748	0.725	0.717	0.742	0.732	**0.752**
prnn_synth	0.857	0.858	0.761	0.708	0.857	0.846	**0.868**
visualizing_enviromental	0.671	0.649	0.587	0.560	0.687	0.642	**0.714**
vineyard	0.793	0.778	0.812	0.759	0.793	0.807	**0.817**
monks-problems-1	0.583	0.580	0.549	0.566	0.529	0.581	**0.605**
acute-inflammations	0.893	0.929	0.760	0.762	0.906	0.968	**1.000**
planning-relax	0.673	0.666	0.666	0.693	0.657	0.681	**0.710**
sensory	0.563	0.554	0.567	0.560	0.559	0.579	**0.600**
auto_price	0.924	0.920	0.762	0.840	0.914	0.929	**0.958**
wisconsin	0.610	0.595	0.552	0.567	0.592	0.614	**0.639**
fri_c4_250_100	0.549	0.559	0.571	0.588	0.487	0.536	**0.595**
thoracic_surgery	0.813	0.814	0.813	0.805	0.820	0.815	**0.823**
conference_attendance	0.853	0.852	0.845	0.847	0.856	0.851	**0.861**
analcatdata_boxing1	0.691	0.639	0.679	0.622	0.668	0.641	**0.729**
fri_c2_100_10	0.700	0.617	0.537	0.585	0.702	0.635	**0.722**
lupus	0.784	0.803	0.707	0.673	0.690	0.792	**0.818**
fruitfly	0.521	0.500	0.567	0.521	0.539	0.507	**0.623**
Average	0.682	0.679	0.645	0.633	0.674	0.683	**0.746**
Median	0.676	0.658	0.665	0.624	0.682	0.649	**0.732**
Minimum	0.374	0.355	0.337	0.361	0.360	0.410	**0.565**
Maximum	0.924	0.929	0.865	0.847	0.914	0.968	**1.000**

Source: (CERVATI NETO; LEVADA, 2024).

space, the performance of these algorithms tends to significantly drop. As deep neural networks, these methods require numerical optimisation algorithms for error/distance minimisation, such as stochastic gradient descent. Table 7 contains all of the acquired findings. The approach with the bold value is the best for that dataset. It can be seen that, for these datasets, PNN-LPP outperformed not only regular LPP, but also UMAP, a state-of-the-art algorithm, implying that the proposed method is a viable alternative for DR-based unsupervised metric learning.

In order to verify whether the PNN-LPP performance is significantly superior than the performances of the other methods in these datasets, a Friedman test, a statistical test that is considered to be a non-parametric version of Analysis of Variance, was performed. For a significance level $\alpha = 0.01$, it can be concluded that there are strong evidences against the null hypothesis (all methods have the same performance) (p-value $= 1.21 \times 10^{-16}$).

Figure 7 – Scatter-plots of the user-knowledge dataset for the 2D case. From left to right, top to bottom: ISOMAP, LLE, LE, LPP, UMAP, and the proposed PNN-LPP for $k = 9$.



Source: (CERVATI NETO; LEVADA, 2024).

Moreover, to check which methods are statistically different, a post-hoc Nemenyi test was performed for pairwise comparisons. According to the test, there are strong evidences that PNN-LPP produced significantly higher average accuracies than PCA (p-value $< 10^{-3}$), ISOMAP (p-value $< 10^{-3}$), LLE (p-value $< 10^{-3}$), LE (p-value $< 10^{-3}$), regular LPP (p-value $< 10^{-3}$), and UMAP (p-value $< 10^{-3}$). A visual comparison of the clusters obtained after DR-based metric learning is performed in the user-knowledge dataset is shown in Figure 7. It must be noted that the discrimination between classes is more evident in the proposed method, as there is less overlap between the clusters.

Despite the fact that the results are intriguing, the proposed strategy has certain drawbacks. The most important is that PNN-LPP is quite sensitive to parameter k, which sets the size of the neighbourhood in PNN. How this parameter is defined has a direct

impact on the results: during the experiments, it was observed that the classification accuracies are quite sensitive to changes in the value of k. A simple strategy was used in this study: to perform a line search in the integers belonging to the interval $\left[2, \max\left(\frac{n}{2}, 50\right)\right]$ for each dataset. The best model is defined as one that optimises classification accuracy over all k values. It is worth mentioning that, although class labels are used to perform model selection, the DR-based metric learning is completely unsupervised. Currently, there is no automated strategy for the estimation of optimal parameter k. For all experiments described in this section, the parameter $t = 1$ and $d = 2$ were fixed. It is expected that better results can be obtained by optimising the values of these parameters.

Alternatively, one advantage of the proposed technique is that it has been shown in various computational experiments that PNN-LPP usually performs better than its competitors when the number of samples is limited. In other words, the proposed strategy appears to be promising for dealing with difficulties involving small sample sizes. The t-SNE and UMAP methods, for example, are state-of-the-art algorithms for DR-based metric learning that require a large number of samples for providing good results, since, as the optimisation problems do not have closed form solutions, they require numerical algorithms (stochastic gradient descend) that demand more data for convergence.

4.4 Kernel Density Estimation-based Isometric Feature Mapping

Two sets of computational experiments were conducted to evaluate the performance of the proposed KDE-ISOMAP for DR-based metric learning:

1. A comparison of the clusters obtained after mapping the data to a two-dimensional subspace, with SC, a measure of how well different clusters fit the data;

2. A comparison of the average classification accuracies for three supervised classifiers, KNN, SVM, and Bayesian classifier under Gaussian hypothesis, after the same feature extraction process.

If the underlying metrics are successfully learnt, it is likely that these two measures are able to reflect it, resulting in a large rise in average scores when examining numerous multivariate datasets. Seven distinct methods were used to compare this technique against, namely: PCA, KPCA, ISOMAP, LLE, LE, t-SNE, and UMAP. The proposed KDE-ISOMAP comes in three different variations: Kernel Density Estimation-based Isometric Feature Mapping with fixed bandwidth $h = 0.1$ for all probability density functions (K-ISO-F), Kernel Density Estimation-based Isometric Feature Mapping with Silverman's rule for bandwidth estimation (K-ISO-SIL), and Kernel Density Estimation-

Table 8 – Silhouette coefficients for clusters generated by algorithms PCA, KPCA, ISOMAP, LLE, and LE for a number of openML.org datasets (2D case).

	PCA	KPCA	ISOMAP	LLE	LE
iris	0.401	0.469	0.452	0.365	0.541
wine	0.526	0.610	0.547	0.242	0.750
prnn_crabs	0.040	0.030	0.037	0.022	0.028
happiness	−0.067	−0.062	−0.066	−0.068	−0.063
mux6	0.072	0.064	−0.015	**0.084**	−0.014
parity5	−0.062	−0.047	−0.048	−0.051	−0.036
Hayes-roth	−0.023	0.038	−0.010	−0.013	−0.013
aids	−0.022	−0.027	−0.027	−0.037	−0.033
pm10	0.000	0.002	0.000	0.000	0.000
strikes	0.007	0.008	0.004	0.002	0.007
disclosure_z	−0.002	0.006	−0.002	−0.001	−0.002
diggle_table_a2	0.406	0.409	0.450	0.328	0.304
Monks-problem	0.024	0.001	0.000	0.000	−0.002
Breast-tissue	−0.029	−0.030	−0.017	−0.081	−0.018
planning-relax	−0.002	−0.011	−0.004	0.003	−0.004
haberman	0.060	−0.024	0.061	−0.004	−0.032
KnuggetChase3	0.199	0.070	0.187	0.077	0.091
bolts	0.337	0.254	0.286	0.028	0.317
fl2000	0.180	0.043	0.119	0.073	0.025
Engine1	−0.133	−0.032	−0.149	−0.170	−0.168
fri_c2_100_10	0.099	0.059	0.083	0.021	0.093
Vineyard(2)	0.277	0.262	0.280	0.191	0.252
diabetes_numeric(2)	0.092	0.081	0.093	0.085	0.089
prnn_fglass	0.018	0.004	0.011	0.029	−0.009
parkinsons	0.130	0.155	0.114	0.002	0.242
Acute-inflammations(2)	0.278	0.315	0.266	0.113	0.081
blogger	0.036	−0.011	0.052	0.029	0.003
prnn_viruses	0.371	0.118	0.112	0.496	0.232
analcatdata_creditscore	0.111	0.081	0.131	0.071	0.049
Confidence(2)	0.173	0.214	0.123	−0.142	−0.087
Average	0.117	0.102	0.102	0.056	0.087
Median	0.066	0.041	0.057	0.022	0.005
Minimum	−0.133	−0.062	−0.149	−0.170	−0.168
Maximum	0.526	0.610	0.547	0.496	0.750
Std.Dev.	0.165	0.167	0.164	**0.144**	0.192

Source: (CERVATI NETO; LEVADA; HADDAD, 2024).

based Isometric Feature Mapping with Scott's rule for bandwidth estimation (K-ISO-SC). In the experiments, the number of density points (bins) used in KDE is set to $L = 256$.

Table 8 and Table 9 contain all of the results for the first set of experiments. It should be noted that in 28 out of the 30 datasets, one of the KDE-ISOMAP versions obtained the best SC, which corresponds to almost 93% of the cases. It can be observed from the averages and medians that the proposed strategy outperformed the alternatives in these datasets.

A non-parametric Friedman test was run to see if the results produced by KDE-ISOMAP are statistically superior to others. The null hypothesis that all groups are identical is strongly refuted (p-value $= 1.11 \times 10^{-16}$) with a significance threshold of $\alpha = 0.05$. Moreover, a post-hoc Nemenyi test was used to determine whether groups were statistically distinct from the others. The test found that, for a significance level of $\alpha = 0.05$, K-ISO-F, K-ISO-SIL, and K-ISO-SC obtained considerably higher SC than PCA, KPCA, ISOMAP, LLE, LE, t-SNE, and UMAP. Table 10 shows the specific p-values for these tests. There

Table 9 – Silhouette coefficients for clusters generated by algorithms t-SNE, UMAP, and KDE-ISOMAP (K-ISO-F, K-ISO-SIL, K-ISO-SC) for a number of openML.org datasets (2D case).

	t-SNE	UMAP	*K-ISO-F*	*K-ISO-SIL*	*K-ISO-SC*
iris	0.494	0.526	0.588	0.597	**0.619**
wine	0.556	0.605	0.742	**0.766**	0.765
prnn_crabs	0.038	0.048	**0.156**	0.117	0.130
happiness	−0.064	−0.052	**0.000**	**0.000**	**0.000**
mux6	0.048	0.037	0.028	0.068	0.038
parity5	−0.016	−0.053	**0.000**	**0.000**	**0.000**
Hayes-roth	−0.012	−0.013	0.090	0.160	**0.215**
aids	−0.013	−0.018	**0.090**	0.054	0.054
pm10	0.000	0.000	**0.006**	0.004	0.002
strikes	0.008	0.019	**0.027**	0.025	0.025
disclosure_z	0.000	0.000	0.005	**0.008**	0.008
diggle_table_a2	0.431	0.199	**0.673**	0.645	0.639
Monks-problem	0.029	−0.001	**0.062**	0.04	0.036
Breast-tissue	−0.024	**−0.001**	−0.007	−0.012	−0.022
planning-relax	−0.006	−0.003	**0.076**	0.035	0.046
haberman	−0.017	−0.027	**0.269**	0.175	0.188
KnuggetChase3	0.062	0.063	0.488	0.502	**0.510**
bolts	0.159	0.347	**0.573**	0.425	0.556
fl2000	0.045	0.014	0.128	**0.302**	0.301
Engine1	−0.118	−0.091	−0.243	−0.049	**0.053**
fri_c2_100_10	0.096	0.092	**0.122**	0.113	0.117
Vineyard(2)	0.301	0.379	**0.415**	0.357	**0.415**
diabetes_numeric(2)	0.069	0.072	0.167	**0.171**	0.170
prnn_fglass	0.052	0.026	**0.078**	0.017	0.039
parkinsons	0.193	0.203	0.191	**0.339**	0.249
Acute-inflammations(2)	0.247	0.355	**0.371**	0.367	0.367
blogger	0.019	−0.033	**0.398**	**0.398**	0.397
prnn_viruses	0.213	0.079	**0.490**	0.358	0.363
analcatdata_creditscore	0.007	0.003	**0.389**	0.257	0.315
Confidence(2)	0.122	0.056	0.540	**0.546**	0.533
Average	0.097	0.094	0.230	0.226	**0.238**
Median	0.042	0.023	0.142	0.166	**0.179**
Minimum	−0.118	−0.091	−0.243	−0.049	**−0.022**
Maximum	0.556	0.605	0.742	**0.766**	0.765
Std.Dev.	0.164	0.176	0.246	0.228	0.230

Source: (CERVATI NETO; LEVADA; HADDAD, 2024).

is no indication that K-ISO-F and K-ISO-SC vary in terms of SC for these datasets (p-value = 0.965) and the same holds true for K-ISO-F and K-ISO-SIL (p-value = 0.982) and K-ISO-SIL and K-ISO-SC (p-value = 0.949).

For each of the datasets, in the second set of experiments, 50% of the samples were used to train three different classifiers after DR-based metric learning: the Bayesian classifier under the Gaussian hypothesis with different covariance matrices for each class (a parametric and quadratic classifier), the SVM with no kernel (a non-parametric and linear classifier), and the KNN with $k = 7$ (a non-parametric and non-linear classifier). The 50% remaining samples from the test set were then classified using each one of them, and the classifier with the highest accuracy was chosen to assess how each metric learning method affects supervised classification. Table 11 and Table 12 contain all of the findings. Notably, one of the three variations of the proposed KDE-ISOMAP obtained the greatest classification accuracy in 26 out of the 30 datasets, which corresponds to almost 86% of

Table 10 – Post-hoc Nemenyi tests for Silhouette Coefficient.

	K-ISO-F	K-ISO-SIL	K-ISO-SC
PCA	1.79×10^{-6}	1.99×10^{-6}	1.44×10^{-6}
KPCA	1.75×10^{-7}	1.97×10^{-7}	1.39×10^{-7}
ISOMAP	4.81×10^{-8}	5.43×10^{-8}	3.78×10^{-8}
LLE	1.20×10^{-11}	1.39×10^{-11}	8.95×10^{-12}
LE	2.42×10^{-10}	2.77×10^{-10}	1.83×10^{-10}
t-SNE	5.43×10^{-7}	6.07×10^{-7}	4.35×10^{-7}
UMAP	1.56×10^{-7}	1.75×10^{-7}	1.24×10^{-8}

Source: (CERVATI NETO; LEVADA; HADDAD, 2024).

the cases.

Once again, a non-parametric Friedman test was used to determine if the classification accuracy results achieved by KDE-ISOMAP are statistically superior. Strong evidence is shown against the null hypothesis that all groups are identical, with a significance level of $\alpha = 0.05$. A post-hoc Nemenyi test was applied to determine whether groups were equivalent or not. The test found that K-ISO-F, K-ISO-SIL, and K-ISO-SC produced significantly higher classification accuracies than PCA, KPCA, ISOMAP, LLE, LE, t-SNE, and UMAP. Table 13 shows the specific p-values for these tests. There is no indication that K-ISO-F and K-ISO-SC vary in terms of classification accuracy for these datasets (p-value = 0.550), and the same holds true for K-ISO-F and K-ISO-SIL (p-value = 0.508), and K-ISO-SIL and K-ISO-SC (p-value = 0.949).

A positive aspect of the proposed KDE-ISOMAP is related to the out-of-sample problem in ML algorithms. Most unsupervised metric learning algorithms are not capable of dealing with new samples that are not part of the training set in a straightforward manner. The natural choice is to add these new samples to the set and perform another full training round, which can be time consuming. It has been shown that ISOMAP is directly related to KPCA: in fact, KPCA becomes ISOMAP when the kernel matrix $K(\vec{x}_i, \vec{x}_j)$ is defined as minus one-half the geodesic distance matrix (HAM et al., 2004). Thus, using this relation, it is possible to deal with out-of-sample instances in KDE-ISOMAP using the same projection strategy of KPCA.

The specification of parameter ϵ (radius), which determines the patch size (the number of neighbours of a particular sample in the ϵ-neighbourhood graph), is one of the method's limitations. Tests showed that the classification accuracy and SC are quite sensitive to changes in this parameter. Here, the following strategy was employed: for each dataset, the complete graph is built by linking a sample to every other sample. Then, for each sample $\vec{x}_i$, the approximate distribution of the distances from $\vec{x}_i$ to any other sample $\vec{x}_j$ is computed. Computational studies show that the percentiles p of these distribution falling within the range $P = [1, 20]$ produce the most suitable values of ϵ. In other words, several percentiles of this distribution are tested as radius and the one that maximises the

Table 11 – Maximum accuracies among KNN, SVM, and Bayesian classifiers after dimensionality reduction with PCA, KPCA, ISOMAP, LLE, and LE for several `openML.org` datasets (2D case).

	PCA	KPCA	ISOMAP	LLE	LE
iris	0.960	0.866	0.920	0.973	0.840
wine	0.966	0.977	0.989	0.797	0.989
prnn_crabs	0.620	0.660	0.610	0.710	0.620
happiness	0.333	0.266	0.333	0.266	0.300
mux6	0.609	0.703	0.703	0.734	0.546
parity5	0.500	0.437	**0.625**	0.437	0.437
Hayes-roth	0.606	0.621	0.636	0.606	0.606
aids	0.480	0.480	0.440	0.480	0.480
pm10	0.532	0.532	0.536	0.512	0.512
strikes	0.638	0.648	0.648	0.661	0.661
disclosure_z	0.531	0.558	0.525	0.519	0.519
diggle_table_a2	0.877	0.974	0.916	0.883	0.929
Monks-problem	0.604	0.600	0.589	0.647	0.669
Breast-tissue	0.415	0.584	0.490	0.547	0.547
planning-relax	0.714	0.714	0.714	0.714	0.714
haberman	0.790	0.764	0.764	0.764	0.764
KnuggetChase3	0.804	0.793	0.793	0.793	0.793
bolts	0.850	0.950	0.900	0.700	0.850
fl2000	0.647	0.676	0.647	0.647	0.617
Engine1	0.791	0.89	0.885	0.765	0.885
fri_c2_100_10	0.740	0.680	0.700	0.540	0.680
Vineyard(2)	0.846	0.807	0.807	0.846	0.846
diabetes_numeric(2)	0.681	0.590	0.681	0.681	0.636
prnn_fglass	0.757	0.719	0.757	0.635	0.710
parkinsons	0.897	0.897	0.816	0.836	0.806
Acute-inflammations(2)	**1.000**	**1.000**	**1.000**	**1.000**	0.967
blogger	0.660	0.680	0.760	0.740	0.680
prnn_viruses	0.839	0.806	0.741	0.774	0.774
analcatdata_creditscore	**0.840**	0.760	0.820	0.760	0.780
Confidence(2)	0.833	0.833	0.888	0.861	0.833
Average	0.712	0.716	0.721	0.694	0.700
Median	0.727	0.709	0.728	0.712	0.695
Minimum	0.333	0.266	0.333	0.266	0.300
Maximum	**1.000**	**1.000**	**1.000**	**1.000**	0.989
Std.Dev.	0.169	0.172	0.162	0.160	0.164

Source: (CERVATI NETO; LEVADA; HADDAD, 2024).

classification accuracy among all values of $p \in P$ is chosen as the best model. It must be noted that, unlike a KNN graph, this graph is not regular, in the sense that the degrees of the vertices can be quite different. It is also worth mentioning that, besides using the class labels to perform model selection, the feature extraction stage is fully unsupervised, in the sense that KDE-ISOMAP performs unsupervised metric learning.

In order to illustrate how the proposed KDE-ISOMAP can improve the clusters and classification accuracy by learning a suitable metric, Figure 8 shows scatter-plots for the AIDS dataset, which illustrates the differences between methods more acutely than most, after reducing the number of features to two. It can be noted that, in comparison to the original ISOMAP, t-SNE, and UMAP, the proposed method produced less overlapping samples in terms of data discrimination. The two classes (Male — circle and Female — cross) are better identified in KDE-ISOMAP than in the other methods.

Table 12 – Maximum accuracies among KNN, SVM, and Bayesian classifiers after dimensionality reduction with t-SNE, UMAP, and KDE-ISOMAP (K-ISO-F, K-ISO-SIL, K-ISO-SC) for several `openML.org` datasets (2D case).

	t-SNE	UMAP	*K-ISO-F*	*K-ISO-SIL*	*K-ISO-SC*
iris	0.986	**1.000**	0.973	**1.000**	**1.000**
wine	0.943	0.943	0.989	0.988	**1.000**
prnn_crabs	0.820	0.810	0.860	**0.900**	0.870
happiness	0.400	0.400	**0.567**	0.400	0.433
mux6	**0.812**	0.734	0.656	0.750	0.688
parity5	0.562	0.375	0.500	0.500	0.500
Hayes-roth	0.727	0.606	**0.818**	**0.818**	0.788
aids	0.520	0.520	0.560	**0.760**	0.600
pm10	**0.588**	0.536	0.564	0.576	0.560
strikes	**0.853**	0.750	0.757	0.767	0.770
disclosure_z	0.558	0.549	0.568	0.565	**0.586**
diggle_table_a2	0.948	0.948	0.968	0.968	**0.974**
Monks-problem	0.748	0.733	**0.766**	0.676	0.687
Breast-tissue	0.566	0.509	0.642	**0.660**	0.623
planning-relax	0.714	0.714	**0.725**	**0.725**	**0.725**
haberman	0.764	0.777	**0.797**	0.791	**0.797**
KnuggetChase3	0.804	0.793	**0.814**	**0.814**	0.804
bolts	0.850	0.900	**0.950**	**0.950**	**0.950**
fl2000	0.647	0.617	0.676	0.676	**0.706**
Engine1	0.901	0.906	0.906	0.927	**0.938**
fri_c2_100_10	0.760	0.700	0.820	0.820	**0.860**
Vineyard(2)	0.807	0.807	**0.885**	**0.885**	**0.885**
diabetes_numeric(2)	0.681	0.590	0.681	0.681	**0.727**
prnn_fglass	0.728	0.710	0.738	**0.757**	0.710
parkinsons	0.897	0.846	0.898	**0.908**	**0.908**
Acute-inflammations(2)	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
blogger	**0.800**	0.650	**0.800**	**0.800**	0.780
prnn_viruses	0.774	0.774	0.839	**0.871**	**0.871**
analcatdata_creditscore	0.820	0.820	0.820	**0.840**	**0.840**
Confidence(2)	0.833	0.833	**0.917**	**0.917**	**0.917**
Average	0.760	0.728	0.782	**0.790**	0.783
Median	0.787	0.742	**0.807**	**0.807**	0.793
Minimum	0.400	0.375	**0.500**	0.400	0.433
Maximum	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
Std.Dev.	0.145	0.167	**0.143**	0.151	0.154

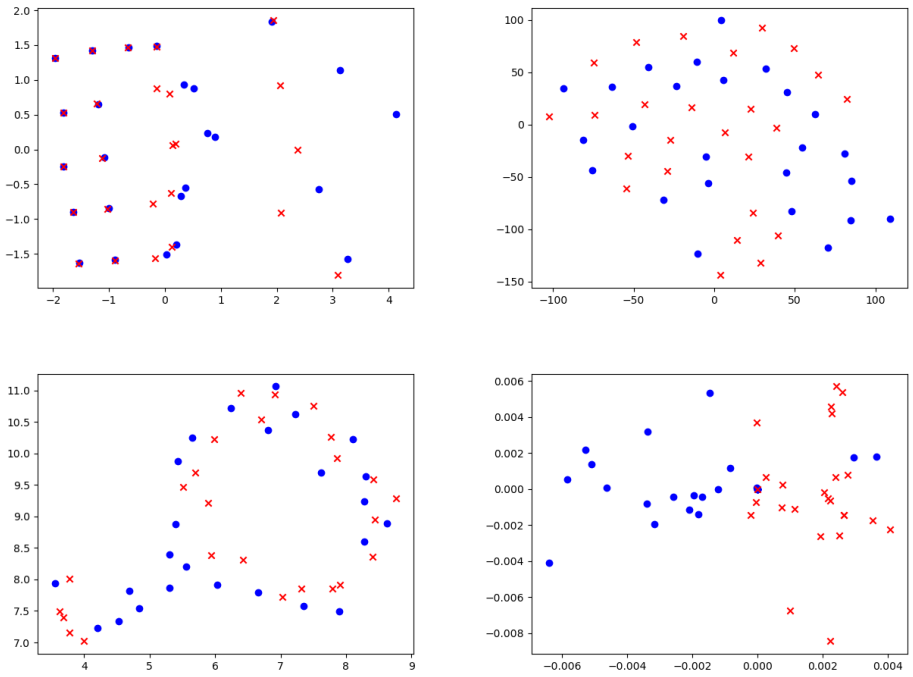
Source: (CERVATI NETO; LEVADA; HADDAD, 2024).

Table 13 – Post-hoc Nemenyi tests for classification accuracy.

	K-ISO-F	K-ISO-SIL	K-ISO-SC
PCA	6.84×10^{-6}	2.47×10^{-7}	3.47×10^{-7}
KPCA	5.56×10^{-6}	1.56×10^{-7}	2.21×10^{-7}
ISOMAP	4.57×10^{-6}	1.57×10^{-7}	2.20×10^{-7}
LLE	1.25×10^{-8}	2.10×10^{-10}	3.18×10^{-10}
LE	8.24×10^{-10}	1.03×10^{-11}	1.61×10^{-11}
t-SNE	0.028	0.004	0.005
UMAP	2.01×10^{-5}	8.43×10^{-7}	1.16×10^{-6}

Source: (CERVATI NETO; LEVADA; HADDAD, 2024).

Figure 8 – Scatter-plots for the AIDS dataset after dimensionality reduction: from left to right, top to bottom, they are ISOMAP, t-SNE, UMAP, and KDE-ISOMAP.



Source: (CERVATI NETO; LEVADA; HADDAD, 2024).

Note: When the number of samples is limited, t-SNE and UMAP tend to perform below the expectations due to the numerical optimisation algorithms.

Chapter 5

Conclusions

Unsupervised metric learning is a fundamental step in many pattern recognition problems dealing with high dimension data. In this scenario, algorithms for DR play an important role, as besides learning an adaptive distance function for each dataset, they also learn an optimal representation for the observed data in terms of compression. In this work was presented ISOMAP-KL, a parametric patch-based method using KL-divergence that maps neighbourhoods of the KNN graph to a feature space in which a surrogate for the pairwise distance matrix is obtained by replacing the usual Euclidean distance by the symmetrised relative entropy between local statistical models. Results with several real datasets indicate that besides improving the quality of the clusters, which is a desirable feature in unsupervised classification, the proposed method can also improve the supervised classification accuracy, indicating that it can be better suited to unsupervised metric learning than regular PCA, KPCA, and some ML algorithms.

DR-based metric learning combines linear or non-linear transforms with the preservation of pairwise relationships in data to produce superior distance functions for classification tasks. The idea is to learn a transform that maps the original high dimension data onto a low dimension space, while optimising a metric capable of capturing the intrinsic geometry of data. This strategy combines DR with metric learning. It decreases data dimensionality while facilitating or eliminating noisy or useless data patterns. It also improves the discriminative capacity and performance of ML algorithms by optimising a distance measure that condenses intended data associations.

Basically, the main positive points of ISOMAP-KL can be summarised as:

1. ISOMAP-KL is a patch-based approach so it is less sensitive to the presence of noise and outliers in data (the entropic distance matrix is computed between pairs of

patches instead of pairs of isolated points);

2. The method can be easily extended to different statistical models and divergences, such as Bhattacharyya and Hellinger distances.

On the other hand, ISOMAP-KL has limitations, the major one being its sensitivity to the patch size k. Experiments have shown that variations on this parameter can produce significantly different classification results.

Among modern DR-based metric learning algorithms, t-SNE is considered the state-of-the-art method. It converts Euclidean distances between samples into probabilities, while attempting to minimise relative entropies between input and output probabilities. Two limitations of t-SNE are that it is an unsupervised learning technique, and it relies on the Euclidean distance, thus being sensitive to outliers.

In the present work are proposed two supervised extensions of t-SNE. One incorporates geodesic distances from the KNN graph, and the other considers stochastic distances based on local multivariate Gaussian densities. Experimental analysis indicate that, despite the increase in terms of computational cost, the proposed methods are capable of yielding superior classification performance compared to standard t-SNE. Particularly, the proposed methods improve the performance of t-SNE when working with small samples. It is widely known that standard t-SNE requires a significant amount of data for convergence, as it depends on numerical optimisation algorithms (gradient descend). With the introduction of the present supervised methodological extensions, this problem is considerably reduced.

PNN-LPP was proposed as a non-parametric approach for DR-based unsupervised metric learning. The goal was to replace the pointwise Euclidean distance by a patch-based probabilistic distance to make LPP more resilient against the presence of variations in data, such as noise and outliers. The proposed PNN-LPP features can be more discriminative in supervised classification than features produced from conventional ML techniques, according to computational experiments. Moreover, one of the main problems with state-of-the-art approaches such as t-SNE and UMAP is the unreasonable performance in small sample size problems due to the necessity of numerical optimisation algorithms. The results indicate that PNN-LPP improves the performance of regular LPP in situations where the number of samples is limited, showing that it can be a viable option in unsupervised metric learning.

This study presents the relative entropy between distributions estimated from patches along the ϵ-neighbourhood graph as a replacement for the Euclidean distance in an entropic ISOMAP that is based on KDE. Since the computational studies validated two key arguments, it can be asserted that the suggested KDE-ISOMAP is a potential substitute for the numerous learning algorithms already described in the literature. The two key arguments are:

1. The KDE-ISOMAP's non-linear features may be more discriminative in supervised classification than features produced by other state-of-the-art ML algorithms;

2. The quality of the clusters produced by KDE-ISOMAP may be superior to that obtained by state-of-the-art ML algorithms.

The superiority of the proposed KDE-ISOMAP in comparison to t-SNE and UMAP becomes clearer when dealing with datasets with limited number of samples, as the numerical optimisation methods required by these two algorithms depend heavily on data.

The main contribution of the proposed framework, which uses a patch-based distance function to measure the similarity between the samples and is more resilient than the pointwise Euclidean distance to deal with the presence of noise and outliers in data, is roughly what accounts for the good performance of KDE-ISOMAP. Furthermore, since a projection matrix can be created, dealing with out-of-sample data is simple thanks to the relationship between KPCA and ISOMAP. The fact that KDE-ISOMAP is suited for small sample size issues and does not require a huge amount of data for convergence should also be mentioned, in contrast to auto-encoders and other deep-learning based algorithms.

Therefore, in general, while the initial intuition that replacing Euclidean distances in existing methods would create improved versions of those was not verified for all cases, most tests show a significantly better performance in classification, confidently indicating that the research hypothesis merits further investigation. Although a few of the methods show no difference to the originals when evaluating classification performance in the studied datasets, there are still sufficient benefits to their usage in specific circumstances to consider them a relevant contribution to the field. Thus, there are many other paths this research could take, and some are suggested next.

5.1 Contributions

The methods proposed in this work have resulted in some publications during its development:

- CERVATI NETO, Alaor. **Exploring information theory-based measures for non-linear dimensionality reduction in manifold learning**. UFSCar, São Carlos, Brazil, 12 Feb. 2020. Poster session. Program: `http://wpsm.icmc.usp.br/8WPSM/Programa8WPSM.pdf`.

- CERVATI NETO, Alaor; LEVADA, Alexandre L. M. ISOMAP-KL: a parametric approach for unsupervised metric learning. In: 2020 33rd SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI). 2020. P. 287–294. DOI: `10.1109/SIBGRAPI51738.2020.00046`.

- CERVATI NETO, Alaor; LEVADA, Alexandre L. M. Probabilistic Nearest Neighbors Based Locality Preserving Projections for Unsupervised Metric Learning. **JUCS — Journal of Universal Computer Science**, Journal of Universal Computer Science, v. 30, n. 5, p. 603–616, 2024. ISSN 0948-695X. DOI: 10.3897/jucs.107081. eprint: https://doi.org/10.3897/jucs.107081. Available from: <https://doi.org/10.3897/jucs.107081>.
- CERVATI NETO, Alaor; LEVADA, Alexandre L. M.; HADDAD, Michel F. C. A Kernel Density Estimation based entropic Isometric Feature Mapping for Unsupervised Metric Learning. **Annals of Data Science**, 25 May 2024. Accepted for publication.

5.2 Future Works

As was done with the methods presented, other DR and ML algorithms can be adapted to use different distances to construct their graphs' neighbourhoods. Measures such as Fisher information (PORTO, 2013) or Jeffreys-Matusita distance (BRUZZONE; ROLI; SERPICO, 1995) can be used to replace Euclidean distances for these methods in a similar manner to those described in this work, along the same intuition that a more stochastic metric may improve the performance of such methods. Another form of optimisation that could be attempted is to investigate adaptive ways to set the patch size based on local properties of the data. Yet another adjust that could yield interesting results is to optimise the methods' hyper-parameters dynamically, using a part of the initial datasets as validation. Finally, comparing the usage of these methods to the results obtained by variational auto-encoders is also a possible avenue for future research.

References

AGGARWAL, Charu C.; HINNEBURG, Alexander; KEIM, Daniel A. On the Surprising Behavior of Distance Metrics in High Dimensional Space. In_____. **Database Theory — ICDT 2001**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001. P. 420–434.

BELKIN, Mikhail; NIYOGI, Partha. Laplacian Eigenmaps and Spectral Techniques for Embedding and Clustering. In: DIETTERICH, T. G.; BECKER, S.; GHAHRAMANI, Z. (Eds.). **Advances in Neural Information Processing Systems 14**. MIT Press, 2002. P. 585–591. Available from: <http://papers.nips.cc/paper/1961-laplacian-eigenmaps-and-spectral-techniques-for-embedding-and-clustering.pdf>.

_____. Laplacian Eigenmaps for Dimensionality Reduction and Data Representation. **Neural Computation**, MIT Press, v. 15, n. 6, p. 1373–1396, June 2003.

BERNSTEIN, M. et al. Graph Approximations to Geodesics on Embedded Manifolds, 2000.

BHATTACHARYYA, Anil Kumar. On a measure of divergence between two statistical populations defined by their probability distributions. **Bulletin of the Calcutta Mathematical Society**, Calcutta Mathematical Society, v. 35, p. 99–109, 1943.

BORG, I.; GROENEN, P. **Modern Multidimensional Scaling: theory and applications**. 2. ed.: Springer-Verlag, 2005.

BRUZZONE, Lorenzo; ROLI, Fabio; SERPICO, Sebastiano B. An Extension of the Jeffreys-Matusita Distance to Multiclass Cases for Feature Selection. **IEEE Transactions on Geoscience and Remote Sensing**, IEEE, v. 33, p. 061318–061321, 1995.

CAYTON, Lawrence. **Algorithms for manifold learning**. 2005. Available from: <http://www.vis.lbl.gov/~romano/mlgroup/papers/manifold-learning.pdf>. Visited on: 26 Feb. 2018.

CERVATI NETO, Alaor. **Exploring information theory-based measures for non-linear dimensionality reduction in manifold learning**. UFSCar, São Carlos, Brazil, 12 Feb. 2020. Poster session. Program: `http://wpsm.icmc.usp.br/8WPSM/Programa8WPSM.pdf`.

CERVATI NETO, Alaor; LEVADA, Alexandre L. M. ISOMAP-KL: a parametric approach for unsupervised metric learning. In: 2020 33rd SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI). 2020. P. 287–294. DOI: `10.1109/SIBGRAPI51738.2020.00046`.

_____. Probabilistic Nearest Neighbors Based Locality Preserving Projections for Unsupervised Metric Learning. **JUCS — Journal of Universal Computer Science**, Journal of Universal Computer Science, v. 30, n. 5, p. 603–616, 2024. ISSN 0948-695X. DOI: `10.3897/jucs.107081`. eprint: `https://doi.org/10.3897/jucs.107081`. Available from: <`https://doi.org/10.3897/jucs.107081`>.

CERVATI NETO, Alaor; LEVADA, Alexandre L. M.; HADDAD, Michel F. C. A Kernel Density Estimation based entropic Isometric Feature Mapping for Unsupervised Metric Learning. **Annals of Data Science**, 25 May 2024. Accepted for publication.

COOK, James et al. Visualizing similarity data with a mixture of maps. In: ARTIFICIAL Intelligence and Statistics. 2007. P. 67–74.

CORMEN, Thomas H. et al. **Introduction to Algorithms**. 3. ed.: MIT Press, 2009. P. 1312.

COVER, Thomas M.; THOMAS, Joy A. **Elements of Information Theory**. 2. ed. Hoboken: Wiley-Interscience, 2006. 748 pp.

COX, Trevor F.; COX, Michael A. A. **Multidimensional Scaling**. Chapman & Hall, 2001. v. 88, p. 295. (Monographs on Statistics and Applied Probability).

CROOKS, Gavin E. **On Measures of Entropy and Information**. 2017. Available from: <`http://threeplusone.com/info`>. Visited on: 3 Apr. 2018.

CUNNINGHAM, John P.; GHAHRAMANI, Zoubin. Linear Dimensionality Reduction: Survey, Insights, and Generalizations. **Journal of Machine Learning Research**, v. 16, p. 2859–2900, 2015.

DEMŠAR, Janez. Statistical Comparisons of Classifiers over Multiple Data Sets. **Journal of Machine Learning Research**, v. 7, p. 1–30, 2006.

DIFFERENCES between Bhattacharyya distance and KL divergence. 28 Dec. 2014. Available from: <`https://stats.stackexchange.com/questions/130432/differences-between-bhattacharyya-distance-and-kl-divergence`>. Visited on: 4 Apr. 2018.

DONG, Wei; MOSES, Charikar; LI, Kai. Efficient k-nearest neighbor graph construction for generic similarity measures. In: WWW '11, 2011. **Proceedings of the 20th International Conference on World Wide Web**. New York, NY, USA: ACM, 2011. P. 577–586.

DUCHI, John C. **Derivations for Linear Algebra and Optimization**. 2007. Available from: <http://web.stanford.edu/~jduchi/projects/general_notes.pdf>. Visited on: 31 Mar. 2018.

DUDA, Richard O.; HART, Peter E.; STORK, David G. **Pattern Classification**. 2. ed.: Wiley-Interscience, 2000.

ERRICA, Federico. **Step-By-Step Derivation of SNE and t-SNE gradients**. 2018. Available from: <http://pages.di.unipi.it/errica/curious/derivations-sne-tsne>.

FAWCETT, Tom. An introduction to ROC analysis. **Pattern recognition letters**, Elsevier, v. 27, n. 8, p. 861–874, 2006.

FIEDLER, Miroslav. Laplacian of graphs and algebraic connectivity. **Banach Center Publications**, v. 25, n. 1, p. 57–70, 1989.

FRENAY, Benoit; VERLEYSEN, Michel. Classification in the Presence of Label Noise: A Survey. **IEEE Transactions on Neural Networks and Learning Systems**, v. 25, n. 5, p. 845–869, 2014. DOI: 10.1109/TNNLS.2013.2292894.

FRIEDMAN, Milton. The Use of Ranks to Avoid the Assumption of Normality Implicit in the Analysis of Variance. **Journal of the American Statistical Association**, Taylor & Francis, v. 32, n. 200, p. 675–701, 1937. DOI: 10.1080/01621459.1937.10503522. eprint: https://www.tandfonline.com/doi/pdf/10.1080/01621459.1937.10503522. Available from: <https://www.tandfonline.com/doi/abs/10.1080/01621459.1937.10503522>.

HAM, Jihun et al. A Kernel View of the Dimensionality Reduction of Manifolds. In: PROCEEDINGS of the Twenty-first International Conference on Machine Learning. New York, NY, USA: ACM, 2004. (ICML '04), p. 47–54.

HINTON, Geoffrey E.; ROWEIS, Sam T. Stochastic Neighbor Embedding. In: BECKER, S.; THRUN, S.; OBERMAYER, K. (Eds.). **Advances in Neural Information Processing Systems 15**. MIT Press, 2003. P. 857–864.

HONARKHAH, Mehrdad; CAERS, Jef. Stochastic Simulation of Patterns Using Distance-Based Pattern Modeling. **Mathematical Geosciences**, v. 42, Honarkhah2010, p. 487–517, 5 1 July 2010. ISSN 1874-8953. DOI: 10.1007/s11004-010-9276-7.

HUDSON, W. D.; RAMM, C. W. Correct Formulation of the Kappa Coefficient of Agreement. **Photogrammetric Engineering & Remote Sensing**, v. 4, p. 421–422, 1987.

HYVARINEN, A.; KARHUNEN, J.; OJA, E. **Independent Component Analysis**. John Wiley & Sons, 2001.

JOLLIFFE, I. T. **Principal Component Analysis**. 2. ed.: Springer, 2002. P. 487.

JUNLIANG, Ma; BING, Xiao; CHENG, Deng. Graph based semi-supervised classification with probabilistic nearest neighbors. **Pattern Recognition Letters**, v. 133, p. 94–101, 2020. ISSN 0167-8655. DOI: https://doi.org/10.1016/j.patrec.2020.01.021. Available from:
<https://www.sciencedirect.com/science/article/pii/S0167865520300337>.

KULLBACK, Solomon; LEIBLER, Richard A. On Information and Sufficiency. **The Annals of Mathematical Statistics**, The Institute of Mathematical Statistics, v. 22, p. 79–86, 1951.

LEE, John A.; VERLEYSEN, Michel. **Nonlinear Dimensionality Reduction**. Springer, 2007.

LEVADA, Alexandre L. M. **A brief introduction to information-theoretic divergences: from relative entropy to Fisher information**. July 2019. Available from:
<https://www.researchgate.net/publication/334576900_A_brief_introduction_to_information-theoretic_divergences_from_relative_entropy_to_Fisher_information>. Visited on: 30 Aug. 2019.

_____. Parametric PCA for unsupervised metric learning. **Pattern Recognition Letters**, v. 135, p. 425–430, 2020.

LI, Bo; LI, Yan-Rui; ZHANG, Xiao-Long. A survey on Laplacian eigenmaps based manifold learning methods. **Neurocomputing**, v. 335, p. 336–351, 2018.

LUXBURG, Ulrike von. A tutorial on spectral clustering. **Statistics and Computing**, v. 17, p. 395–416, 4 2007.

MAATEN, Laurens van der; HINTON, Geoffrey. Visualizing High-dimensional Data using t-SNE. **Journal of Machine Learning Research**, JMLR, v. 9, p. 2579–2605, 2008.

MCINNES, Leland; HEALY, John; MELVILLE, James. **UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction**. 2020. arXiv: 1802.03426 [stat.ML].

MEILĂ, Marina; ZHANG, Hanyu. Manifold Learning: What, How, and Why. **Annual Review of Statistics and Its Application**, Annual Reviews, v. 11, Volume 11, 2024, p. 393–417, 2024. ISSN 2326-831X. DOI:
https://doi.org/10.1146/annurev-statistics-040522-115238. Available from:
<https://www.annualreviews.org/content/journals/10.1146/annurev-statistics-040522-115238>.

MELVILLE, James. **sneer: Stochastic Neighbor Embedding Experiments in R**. 2015. Available from: <http://jlmelville.github.io/sneer/gradients.html>.

NEMENYI, Peter Bjorn. **Distribution-free multiple comparisons.** Princeton University, 1963.

NIE, Feiping; WANG, Xiaoqian; HUANG, Heng. Clustering and projected clustering with adaptive neighbors. In: PROCEEDINGS of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York, New York, USA: Association for Computing Machinery, 2014. (KDD '14), p. 977–986. ISBN 9781450329569. DOI: 10.1145/2623330.2623726. Available from: <https://doi.org/10.1145/2623330.2623726>.

NIELSEN, F.; SUN, K.; MARCHAND-MAILLET, S. On Hölder projective divergences. **Entropy**, v. 19, n. 3, 2017.

PARZEN, Emanuel. On Estimation of a Probability Density Function and Mode. **Annals of Mathematical Statistics**, The Institute of Mathematical Statistics, v. 33, n. 3, p. 1065–1076, 1962.

PORTO, Julianna Pinele Santos. **Geometria da Informação**: Métrica de Fisher. 2013. 70 pp. Master Thesis – Universidade Estadual de Campinas, Campinas.

RADUCANU, Bogdan; DORNAIKA, Fadi. A supervised non-linear dimensionality reduction approach for manifold learning. **Pattern Recognition**, Elsevier, v. 45, p. 062432–062444, 2012.

RIBEIRO, Bernardete; VIEIRA, Armando; NEVES, João Carvalho das. Supervised Isomap with Dissimilarity Measures in Embedding Learning. In: IBEROAMERICAN CONGRESS ON PATTERN RECOGNITION, 2008, Havana. **Progress in Pattern Recognition, Image Analysis and Applications**. Ed. by José Ruiz-Shulcloper and Walter G. Kropatsch. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008. P. 389–396.

RIDDER, Dick de; DUIN, Robert P.W. **Locally Linear Embedding for Classification**. 2002.

RIDDER, Dick de; KOUROPTEVA, Olga, et al. Supervised Locally Linear Embedding. In: INTERNATIONAL CONFERENCE ON NEURAL INFORMATION PROCESSING, 2003, Istanbul. **Artificial Neural Networks and Neural Information Processing — ICANN/ICONIP 2003**. Ed. by Okyay Kaynak. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003. P. 333–341.

RIEMANN, Bernhard. **Grundlagen für eine allgemeine Theorie der Functionen einer veränderlichen complexen Grösse**. 1851. Ph.D. Thesis – Georg-August-Universität Göttingen, Göttingen.

ROSENBLATT, Murray. Remarks on Some Nonparametric Estimates of a Density Function. **Annals of Mathematical Statistics**, The Institute of Mathematical Statistics, v. 27, n. 3, p. 832–837, 1956.

ROUSSEEUW, Peter J. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. **Journal of Comp. and Appl. Math.**, v. 20, p. 53–65, 1987.

ROWEIS, Sam T.; SAUL, Lawrence K. Nonlinear Dimensionality Reduction by Locally Linear Embedding. **Science**, American Association for the Advancement of Science, v. 290, p. 2323–2326, 2000.

SAUL, Lawrence K.; ROWEIS, Sam T. **An Introduction to Locally Linear Embedding**. 2000.

_____. Think Globally, Fit Locally: Unsupervised Learning of Low Dimensional Manifolds. **Journal of Machine Learning Research**, v. 4, p. 119–155, 2003.

SCHÖLKOPF, Bernhard; SMOLA, Alexander; MÜLLER, Klaus Robert. Kernel principal component analysis. In: ADVANCES in Kernel Methods — Support Vector Learning. MIT Press, 1999. P. 327–352.

_____. Nonlinear component analysis as a kernel eigenvalue problem. **Neural Computation**, v. 10, n. 5, p. 1299–1319, 1998.

SCOTT, David W. On optimal and data-based histograms. **Biometrika**, v. 66, n. 3, p. 605–610, 1979.

SHALIZI, Cosma. **Nonlinear Dimensionality Reduction I: Local Linear Embedding**. 2009. Available from:
<http://www.stat.cmu.edu/~cshalizi/350/lectures/14/lecture-14.pdf>.

SHANNON, Claude Elwood; WEAVER, Warren. **The Mathematical Theory of Communication**. Urbana: The University of Illinois Press, 1964. 125 pp.

SHIRKHORSHIDI, A.S.; AGHABOZORGI, S.; WAH, T.Y. A Comparison Study on Similarity and Dissimilarity Measures in Clustering Continuous Data. **PLOS ONE**, 2015.

SILVERMAN, B.W. **Density Estimation for Statistics and Data Analysis**. New York: Chapman & Hall/CRC, 1986.

SPUREK, Przemysław; PAŁKA, Wiesław. Clustering of Gaussian distributions. In: IEEE. 2016 International Joint Conference on Neural Networks (IJCNN). 2016. P. 3346–3353.

TENENBAUM, Joshua B.; SILVA, Vin de; LANGFORD, John C. A Global Geometric Framework for Nonlinear Dimensionality Reduction. **Science**, American Association for the Advancement of Science, v. 290, p. 2319–2323, 2000.

THEODORIDIS, Sergios; KOUTROUMBAS, Konstantinos. **Pattern Recognition**. 4. ed.: Academic Press, 2008. P. 984.

TIKHONOV, Andreï Nikolaevich; ARSENIN, V. I. A. K. **Solutions of ill-posed problems**. 1977.

VAPNIK, Vladimir N. **The Nature of Statistical Learning Theory**. Springer-Verlag, 1993.

WANG, Fei; SUN, Jimeng. Survey on Distance Metric Learning and Dimensionality Reduction in Data Mining. **Data Min. Knowl. Discov.**, Kluwer Academic Publishers, Hingham, MA, USA, v. 29, n. 2, p. 534–564, Mar. 2015. ISSN 1384-5810. DOI: 10.1007/s10618-014-0356-z. Available from: <http://dx.doi.org/10.1007/s10618-014-0356-z>.

WEBB, Andrew R.; COPSEY, Keith D. **Statistical Pattern Recognition**. 3. ed.: John Wiley & Sons, 2011.

XIAOFEI, He; NIYOGI, Partha. Locality Preserving Projections. In_____. **Advances in Neural Information Processing Systems**. MIT Press, 2003. v. 16. Available from: <https://proceedings.neurips.cc/paper_files/paper/2003/file/d69116f8b0140cdeb1f99a4d5096ffe4-Paper.pdf>.

YOUNG, T. Y.; CALVERT, T. W. **Classification, Estimation and Pattern Recognition**. Elsevier, 1974.