**ALMA MATER STUDIORUM - UNIVERSITÀ DI BOLOGNA**

**DOUBLE DEGREE WITH THE UNIVERSITY OF GLASGOW**

DEPARTMENT OF STATISTICAL SCIENCES

"PAOLO FORTUNATI"

Bachelor in Statistical Sciences

Curriculum Stats&Maths

# Dimension reduction: PCA and GPLVM for Simulated Cylinder and Real Handwritten Digits

**(Multivariate Analysis)**

**Presented by:**

Ivan Carnevali

0001020167

**Supervisors:**

Mu Niu

Angela Montanari

SESSION I

ACADEMIC YEAR 2023/2024

# Acknowledgments

*Ringrazio il Professore Mu Niu, che mi ha accompagnato in questo percorso, facendomi da guida. Allo stesso modo ringrazio la Professoressa Angela Montanari: questa tesi non sarebbe stata scritta senza il suo prezioso contributo, nè avrei avuto modo di affrontare l'esperienza di crescita che è stata Glasgow.*

*Ringrazio Marco Palma, che con il suo bagaglio di esperienza e con la pazienza che lo contraddistingue è stato il fratello maggiore che tutti meriteremmo di avere nell'affrontare il percorso accademico, che nasconde insidie e momenti di difficoltà.*

*Ringrazio mia madre, che mi sostiene da ormai vent'anni in ogni circonstanza, e che è in grado di aiutarmi a capire quale sia la scelta migliore senza mai intromettersi nel definire il mio percorso di vita. Ringrazio mio padre che se non mi avesse parlato fin da neonato di cose complicate che senza ombra di dubbio non comprendevo, passando per pazzo agli occhi della gente, chissà se oggi saremmo qui.*

*Come loro, ringrazio tutta la mia famiglia*[1]*, che è stata di supporto in questi anni assicurandomi un ambiente sereno e protetto, indispensabile per riuscire nel percorso che ho intrapreso.*

*Un ringraziamento speciale a tutte le persone che hanno reso Glasgow un posto meno freddo. A Giorgio e ad Alessandro, che senza loro e le risate insieme, le serate d'inverno avrebbero la meglio.*

*Grazie a Nunzia, che per (s)piacevoli coincidenze ha condiviso con me solo metà di una triennale, anche se in realtà siamo sempre rimasti vicini, e continueremo ad esserlo ancora.*

*Grazie a chi, da Roma, dopo tre anni, è ancora presente, che sia da Ibba, o*

[1] Argo è ovviamente incluso in quest'ultima

*improvvisando viaggi in un'altra regione d'Italia.*
*Merita un ringraziamento il Terzo Piano, che lì è impossibile sentirsi soli, e che nei momenti di sconforto è in grado di essere vicino come una famiglia.*
*Grazie ad Andrea, Lorenzo, Simone, Silvia e Greta, perché non saprei come immaginare la mia vita oggi se non vi avessi incontrati tre anni fa. Ci siete sempre, nei momenti di gioia, come oggi, e nei momenti più difficili, che ne abbiamo passati. Ci siete stati e ho la certezza che ci sarete, e più di tutto vi ringrazio per questo. Grazie per avermi fatto sentire amato e coccolato, indipendentemente da dove ci trovassimo, sparsi in Europa.*
*Grazie Agnese per avermi voluto al tuo fianco quando l'idea di godersi la propria autonomia era allettante, e grazie per essere stata al mio fianco, insegnandomi che la più bella autonomia è quella che abbiamo, e avremo, insieme.*

# Contents

# Chapter 1

# Project introduction

## 1.1 Abstract and Aim of the project

The main objective of this project is to perform dimensionality reduction firstly on a simulated dataset, and then on a real dataset of handwritten digits, obtaining a bi-dimensional latent space for the datasets, in order to provide an easier visualisation for the dataset by summarising it and keeping the majority of the information in a reduced space.
For the dimensionality reduction two methodologies will be compared: Principal Component Analysis (PCA) and Gaussian Process Latent Variable Models (GPLVM).
The final aim of the project is to understand how the two different techniques behave in the case of the simulated dataset, and whether the difference in performance between the two remains constant or varies when considering the real dataset.
It will also be of interest to analyse how the theoretical properties of the two techniques will impact the analysis in practice, trying to understand which characteristics are best suited to the given context.

## 1.2 Background

Since most of the structures that can be found are represented in a more complex space than the one which is intrinsic to them, some of the information in the dataset is redundant.
By exploiting this redundancy, it is possible to simplify the structure of the dataset, keeping only non-superfluous information, and finding a lower dimensionality representation of the same structure [Simon (1996)]. This is due to the fact that rather than representing the dimensionality of the data, the representation is a reflection of the dimensionality in the data collection process [Ek and Lawrence (2009)]. Regarding the above, dealing with high dimensional structures could lead to some issues such as high computational cost to perform learning and model over-fitting. Dimension reduction allows these problems to be solved providing an improved generalisation of the model and mitigating the collinearity. It also offers an easier interpretation and visualisation of the dataset and an increased storage and computational efficiency.
For all of these reasons dimensionality reduction is widely used in computer vision tasks such as object recognition, image classification and video analysis, or in domains such as Speech Recognition and Noise Reduction.

# Chapter 2

# Methodology

## 2.1 Principal Component Analysis (PCA)

### 2.1.1 Introduction

Principal component analysis is one of the most popular multivariate statistical methods. It was firstly introduced by Pearson [Pearson (1901)] and later on developed according to a very different perspective by Hotelling [Hotelling (1933)]. The two formulations result to be equivalent, and since the Hotelling's formulation is more often used, this one will be presented in this project.
The idea of PCA is to find a small number of linear combinations of the observed variables which explain most of the variation in the observed data, or, namely, to find a dimension along which the observations are maximally separated. The first principal component is the linear combination with maximal variance, the second principal component is the linear combination with maximal variance in a direction orthogonal to the first principal component, and so on.
The entire process will therefore consist of a rotation of the original reference system such that the variance of the projections along the new axes is maximised. In this way, the axes for which the rotation will have a variance that can be considered noise, will be discarded, thus preserving much of the original variability of the data and performing a dimensionality reduction.

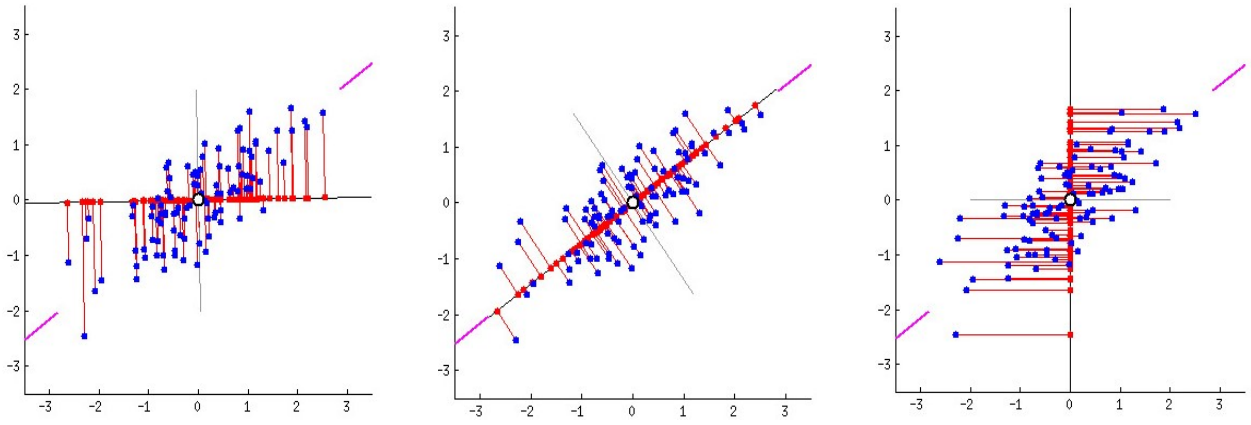


*Figure 2.1: Different rotations of the axes system [Dholakiya (2023)].*

### 2.1.2 Mathematical formulation

Let $\mathbf{x}$ be a $p$-dimensional random vector with expected value $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$, and define a linear combination $y_1 = \mathbf{a}_1^T\mathbf{x}$. The aim of PCA is to find the vector $\mathbf{a}_1$ such that the variance of $y_1$, $Var(y_1) = \mathbf{a}_1^T\boldsymbol{\Sigma}\mathbf{a}_1$, is maximised. The function for the variance of $y_1$ doesn't admit a finite maximum, since it results to be a positive semi-definite quadratic form, given that $\boldsymbol{\Sigma}$ is the covariance of $\mathbf{x}$. Since there is no interest in maximising the variance by increasing the norm of the vector, and given that a constraint must be imposed in order to find a finite solution to the problem presented here, the following condition will be imposed: $\mathbf{a}_1$ will be chosen to be a norm 1 vector, namely $\mathbf{a}_1^T\mathbf{a}_1 = 1$. More formally, a vector $a_1$ is being looked for such that:

$$\max_{a_1} \mathbf{a}_1^T\boldsymbol{\Sigma}\mathbf{a}_1 \text{ subject to } \mathbf{a}_1^T\mathbf{a}_1 = 1.$$

The problem can be restated in the constrained optimization framework based on Lagrange multipliers by defining the function

$$\phi = \mathbf{a}_1^T\boldsymbol{\Sigma}\mathbf{a}_1 - \lambda_1(\mathbf{a}_1^T\mathbf{a}_1 - 1)$$

and by looking for the vector $a_1$ that maximizes it. To address this problem it is necessary to differentiate with respect to $a_1$ and to equate to 0 all the partial derivatives:

$$\frac{\partial\phi}{\partial\mathbf{a}_1} = 2\boldsymbol{\Sigma}\mathbf{a}_1 - 2\lambda_1\mathbf{a}_1 = 0$$

leading up to

$$\boldsymbol{\Sigma}\mathbf{a}_1 = \lambda_1\mathbf{a}_1. \tag{2.1}$$

The relationship between eigenvalues and eigenvectors of the covariance matrix $\boldsymbol{\Sigma}$ can be easily recognised: $\lambda_1$ turns out to be an eigenvalue of $\boldsymbol{\Sigma}$, with $\mathbf{a}_1$ being the corresponding eigenvector.
By premultiplying both sides of Equation 2.1 by $\mathbf{a}_1^T$, because of the unit norm constraint, the following is obtained:

$$\mathbf{a}_1^T\boldsymbol{\Sigma}\mathbf{a}_1 = \mathbf{a}_1^T\lambda_1\mathbf{a}_1 = \lambda_1.$$

But the left hand side of the equation corresponds to the quantity that has to be maximised, since $Var(y_1) = \mathbf{a}_1^T\boldsymbol{\Sigma}\mathbf{a}_1$.
Therefore, in order to derive the linear combination having the largest variance, the largest eigenvalue of $\boldsymbol{\Sigma}$ will be picked, and the corresponding eigenvector will be the first principal component, namely the direction with maximum variance.
The same process can be repeated in order to obtain the second principal component, that is the linear combination with maximal variance in a direction orthogonal to the first principal component. The mathematical formulation of this problem is the following:

$$\phi = \mathbf{a}_2^T\boldsymbol{\Sigma}\mathbf{a}_2 - \lambda_2(\mathbf{a}_2^T\mathbf{a}_2 - 1) - \lambda_3\mathbf{a}_2^T\mathbf{a}_1,$$

where $\lambda_2(\mathbf{a}_2^T\mathbf{a}_2 - 1)$ is the term for the unit norm constraint of the vector $\mathbf{a}_2$, and $\lambda_3\mathbf{a}_2^T\mathbf{a}_1$ is the term for the orthogonality constraint between $\mathbf{a}_1$ and $\mathbf{a}_2$. After differentiating with respect to $\mathbf{a}_2$ and equating the derivatives to 0, the following equation is obtained:

$$2\mathbf{\Sigma}\mathbf{a}_2 - 2\lambda_2\mathbf{a}_2 - \lambda_3\mathbf{a}_1 = 0. \tag{2.2}$$

By premultiplying both sides of the Equation 2.2 by $\mathbf{a}_1^T$, the following is obtained:

$$2\mathbf{a}_1^T\mathbf{\Sigma}\mathbf{a}_2 - 2\lambda_2\mathbf{a}_1^T\mathbf{a}_2 - \lambda_3\mathbf{a}_1^T\mathbf{a}_1 = 0. \tag{2.3}$$

Remembering that $\mathbf{a}_1^T$ is an eigenvector of $\mathbf{\Sigma}$, and thus $\mathbf{a}_1^T\mathbf{\Sigma} = \lambda_1\mathbf{a}_1^T$, while keeping in mind that $\mathbf{a}_2^T\mathbf{a}_2 = 1$ (unit norm constraint) and that $\mathbf{a}_1^T\mathbf{a}_2 = 0$ (orthogonality constraint), $\lambda_3 = 0$ is obtained. Equation 2.3 therefore results in:

$$\mathbf{\Sigma}\mathbf{a}_2 = \lambda_2\mathbf{a}_2.$$

Thus $\lambda_2$ is the second largest eigenvalue of $\mathbf{\Sigma}$ and the corresponding eigenvector $\mathbf{a}_2$ is the direction, orthogonal to the first principal component $\mathbf{a}_1$, accounting for the largest variance. The above process can be repeated for all principal components. It is indeed possible to derive as many principal components as there are observed variables. In general, the $k$-th principal component of $\mathbf{x}$ is $y_k = \mathbf{a}_k^T\mathbf{x}$ and $V(\mathbf{a}_k^T\mathbf{x}) = \lambda_k$, where $\lambda_k$ is the $k$-th largest eigenvalue of $\mathbf{\Sigma}$, and $\mathbf{a}_k$ is the corresponding eigenvector.

### 2.1.3 Properties of PCA

Applying PCA, starting from the $p$-dimensional random vector $\mathbf{x}$, a new $p$-dimensional random vector $\mathbf{y}$ has been achieved, such that $\mathbf{y} = \mathbf{A}^T\mathbf{x}$ where $\mathbf{A}$ is an orthonormal matrix whose $k$-th column is the eigenvector of corresponding to the $k$-th largest eigenvalue. Thus, the PCs are defined by an orthonormal linear transformation of $\mathbf{x}$. Directly from this derivation it is possible to obtain $\mathbf{\Sigma}\mathbf{A} = \mathbf{A}\mathbf{\Lambda}$, where $\mathbf{\Lambda}$ is the diagonal matrix whose $k$-th diagonal element is $\lambda_k$, the $k$-th eigenvalue of $\mathbf{\Sigma}$, and $\lambda_k = V(\mathbf{a}_k^T\mathbf{x}) = V(\mathbf{y}_k)$. Two alternative ways of expressing $\mathbf{\Sigma}\mathbf{A} = \mathbf{A}\mathbf{\Lambda}$ that derive from $\mathbf{A}$ being orthogonal are

$$\mathbf{A}^T\mathbf{\Sigma}\mathbf{A} = \mathbf{\Lambda},$$

meaning that the covariance matrix of the principal components is diagonal, and therefore the principal components are uncorrelated, and

$$\mathbf{\Sigma} = \mathbf{A}\mathbf{\Lambda}\mathbf{A}^T,$$

with $\mathbf{\Sigma}$ and $\mathbf{\Lambda}$ being similar matrices, implying that they have the same trace $\mathbf{tr}(\mathbf{\Sigma}) = \mathbf{tr}(\mathbf{\Lambda})$. This means that the Principal component transformation leaves the total variance unchanged, distributing it differently between the various dimensions.

## 2.2 Gaussian Process Latent Variable Model (GPLVM)

### 2.2.1 Introduction to Gaussian Processes

Gaussian processes are an excellent tool because they manage to combine two fundamentally important traits: a complex and consistent mathematical treatment, and computational

tractability. These two properties together make it a very useful modelling tool used both in statistics and in machine learning. Gaussian processes are very suitable to model functions and make predictions, and can be considered as as a generalisation of the Multivariate Gaussian distribution. Specifically, it is possible to think at the limit case in which the random variables that appear as vectors in the Multivariate Gaussian distribution increase their dimensionality more and more, until they become a continuous function.
A Gaussian process can therefore be thought of as a collection of random variables, any finite number of which have a joint Gaussian distribution. It defines a distribution over functions, and it is fully characterized up to a mean and a covariance function.

### 2.2.2 Multivariate Gaussian distribution

As mentioned in the subsection 2.2.1, Gaussian processes have their foundation on the Multivariate Gaussian distribution. It is therefore useful to have a clear understanding of what the latter is.
The probability density function of a Multivariate Gaussian distribution of dimension $\boldsymbol{n}$ with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$ is given by the following equation:

$$f(\mathbf{x}) = \frac{1}{(2\pi)^{n/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right) \tag{2.4}$$

where

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad \boldsymbol{\mu} = \begin{bmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_n \end{bmatrix}, \quad \boldsymbol{\Sigma} = \begin{bmatrix} \sigma_1^2 & \sigma_{12} & \dots & \sigma_{1n} \\ \sigma_{21} & \sigma_2^2 & \dots & \sigma_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{n1} & \sigma_{n2} & \dots & \sigma_n^2 \end{bmatrix}.$$

#### Properties

Many of the properties of the Multivariate Gaussian distribution also apply to Gaussian processes. The main ones are listed here.

- The **marginal distribution** of any subset of dimensions is a univariate Gaussian distribution. In particular, for the bivariate case:

$$p_{X,Y}([x,y]) \sim \mathcal{N}\left(\begin{bmatrix}\mu_X \\ \mu_Y\end{bmatrix}, \begin{bmatrix}\Sigma_{XX} & \Sigma_{XY} \\ \Sigma_{YX} & \Sigma_{YY}\end{bmatrix}\right)$$

$$X \sim \mathcal{N}(\boldsymbol{\mu}_X, \boldsymbol{\Sigma}_{XX}), \quad Y \sim \mathcal{N}(\boldsymbol{\mu}_Y, \boldsymbol{\Sigma}_{YY})$$

and is possible to get the marginal distribution of $X$ by integrating out the $Y$:

$$p_X(x) = \int p_{X,Y}(x,y)\,dy = \int p_{X|Y}(x \mid y)\,p_Y(y)\,dy$$

- The **conditional distribution** of any subset of dimensions given the values of the remaining dimensions is also a Multivariate Gaussian distribution. In particular, for the bivariate case, is possible to obtain the following formula for the conditional distribution of $X$ given $Y$.

$$X \mid Y \sim \mathcal{N}(\mu_X + \Sigma_{XY}\Sigma_{YY}^{-1}(Y - \mu_Y), \Sigma_{XX} - \Sigma_{XY}\Sigma_{YY}^{-1}\Sigma_{YX})$$

- Any **linear combination** of the components of a Multivariate Gaussian distribution is a univariate Gaussian distribution.

- $Y_i$ and $Y_j$ are **independent** if and only if $\Sigma_{ij} = 0$. Namely, if the covariance between two variables is equal to 0, they are independently distributed.

### 2.2.3 Gaussian Processes

#### Definition

A Gaussian process is a stochastic process with Gaussian finite-dimension distribution and it defines a distribution over functions. Is possible to think of it as an infinite-dimensional generalization of a multivariate Gaussian distribution. It is fully specified by a mean function $\mu(x)$, that represents the expected value of the function at each point, and a covariance function $k(x, x')$ called kernel, which describes how the values of the function at different points are correlated. The Gaussian process can be denoted as

$$f(x) \sim \mathcal{GP}(\mu(x), k(x, x')). \tag{2.5}$$

For any finite collection of random variables $X_1, X_2, \ldots, X_n \subset X$, the joint distribution of the corresponding function values $f(X_1), f(X_2), \ldots, f(X_n)$ is a Multivariate Gaussian distribution:

$$\begin{bmatrix} f(x_1) \\ \vdots \\ f(x_n) \end{bmatrix} \sim \mathcal{N}\left( \begin{bmatrix} \mu(x_1) \\ \vdots \\ \mu(x_n) \end{bmatrix}, \begin{bmatrix} k(x_1, x_1) & \ldots & k(x_1, x_n) \\ \vdots & \ddots & \vdots \\ k(x_n, x_1) & \ldots & k(x_n, x_n) \end{bmatrix} \right) \tag{2.6}$$

#### Kernels

The kernel function $k : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ takes two points $x, x' \in \mathbb{R}^n$ as input and returns a similarity measure in the form of a scalar: $k(x, x') : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$. The kernel, namely the covariance matrix $\Sigma$ between two sets of points $X$ and $X'$, is defined as $\Sigma = \mathrm{Cov}(X, X') = k(x, x')$. Different similarity measures can be chosen in order to define the covariance matrix, and depending on which one is chosen a specific kernel type will be defined for the Gaussian process. The choice of which one to implement is relevant, since it directly influences many aspects of the stochastic process, including the shape, the differentiability and the variance of the function.

### Types of kernels

Kernels can be divided into two major categories: **stationary kernels**, which depend exclusively on the Euclidean distance of the two points considered, and not on their absolute values, being therefore invariant to translations, and **non-stationary kernels**, which also depend on the specific values of the two points on which similarity is measured. The most commonly used ones are:

- **Radial Basis Function Kernel (RBF)**: a stationary kernel with covariance function

$$k(x, x') = \exp\left(-\frac{\|x - x'\|^2}{2\ell^2}\right). \tag{2.7}$$

  The similarity function, as defined, gives higher weight to points that are close to each other. The covariance between two points decreases as their distance increases. Moreover, using an RBF kernel allows to generate smooth functions that result to be infinitely differentiable: this is an useful property in order to capture complex patterns. It should also be considered that this kernel has a particularly simple form for the covariance matrix, since the inverse of the covariance matrix is a diagonal matrix. This is an helpful property in order to simplify calculations.

- **Matérn**: a stationary kernel with covariance function

$$k(x, x') = \frac{2^{1-\nu}}{\Gamma(\nu)}\left(\frac{\sqrt{2\nu}}{\ell}\|x - x'\|\right)^{\nu} K_\nu\left(\frac{\sqrt{2\nu}}{\ell}\|x - x'\|\right), \tag{2.8}$$

  where $\Gamma(\nu)$ is the gamma function and $K_\nu$ is a modified Bessel function.
  This kernel is a generalisation of the RBF kernel. It has an additional parameter $\boldsymbol{\nu}$ which controls the smoothness of the covariance function. Popular choices for this parameter are $\nu = 3/2$ or $\nu = 5/2$, resulting in functions that are respectively once or twice differentiable.

- **Exponential**: a stationary kernel with covariance function

$$k(x, x') = \exp\left(-\ell\|x - x'\|\right). \tag{2.9}$$

  The result of this kernel is similar to that of the RBF kernel, but here the covariance decreases exponentially with the distance between the two points, allowing the capturing of long-range correlations more strongly than the RBF kernel.

Figure 2.2 shows the differences among the three discussed kernels.

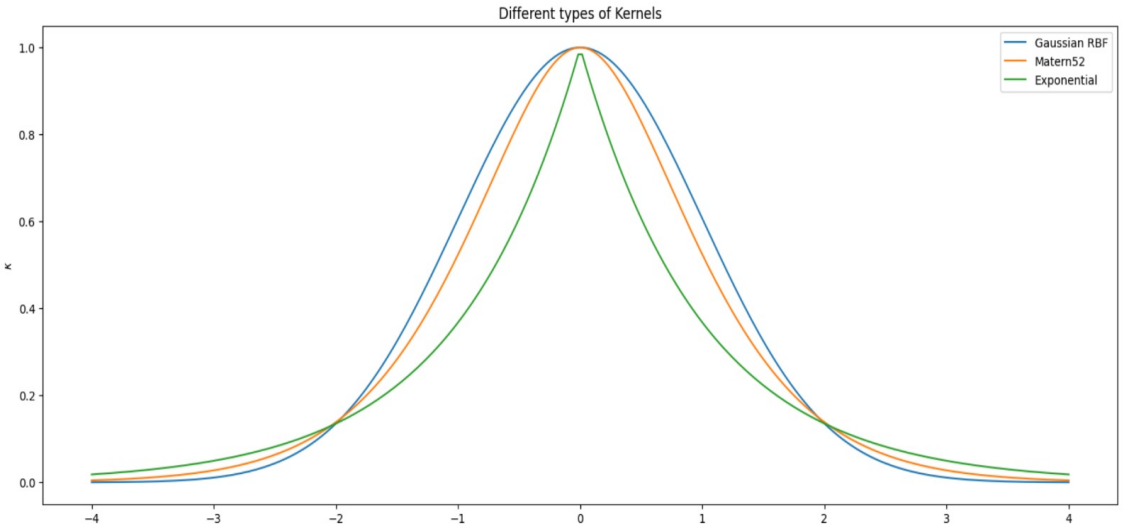



*Figure 2.2: Kernel function of the three discussed kernels*

In the previous formulae $l$, $p$ and $\nu$ are hyperparameters. They will be discussed in the following subsection.
It is possible to adapt the covariance function to model all the major stochastic processes, such as Brownian motion and White noise. Other widely used kernels are the one in the family of Periodic Kernels, but these will not be used in this project as the considered dataset does not display periodicity.

### Hyperparameters

As seen, the kernel functions depend on certain parameters such as the ***length-scales*** ($l$) the ***variance*** ($\sigma^2$) and the ***smoothness*** ($\nu$).
Changing the **length-scales** is equivalent to rescaling our points by $\frac{1}{l}$ before computing the covariance. Choosing a high value for the length-scale will imply that the two points after rescaling will be mapped to a narrower range of values, and this will imply a higher correlation between them. As a consequence the described functions will be more similar to a linear function. If one instead selects a low value for the length-scale, the two points after rescaling will be mapped to a wider range of values, and this will imply a lower correlation between them. In summary, high values for the length-scale lead to long-range correlation, while lower values length-scales lead to short-range correlation.
The **variance** parameter controls the amplitude of the covariance function. The choice of variance can have a significant impact on the behavior of the GP model and on the quality of the fit to the data.
The **smoothness** determines how many times the covariance function is differentiable. In particular the covariance function will be $\nu - 1$ times differentiable. Popular choices for the smoothness parameter are $\nu = \frac{3}{2}$ and $\nu = \frac{5}{2}$, while for $\nu \to \infty$ and for $\nu = \frac{1}{2}$ the RBF and the exponential kernel are respectively obtained [Rasmussen and Williams (2006)].
How the covariance function varies according to changes in the hyperparameters discussed above can be observed in Figure 2.4, Figure 2.3 and Figure 2.5, applied to the three kernels shown in Figure 2.2.

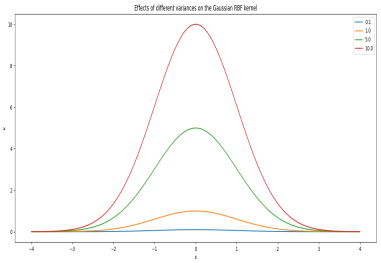


*Figure 2.3: Effects of different variances on the Gaussian RBF kernel.*

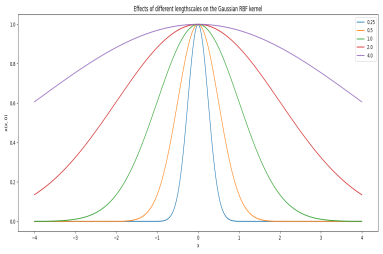


*Figure 2.4: Effects of different length-scales on the Gaussian RBF kernel.*

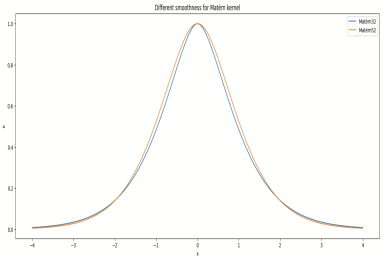


*Figure 2.5: Effects of different smoothness on the Matérn kernel.*

### Combination of different kernels

A great advantage of Gaussian processes is that kernels can be combined together, creating a new, more customised kernel. Combining different kernels allows in fact for greater flexibility and an improved fit of the Gaussian process to the data, in order to introduce domain knowledge into the process and obtain a powerful tool.
Two common ways of combining kernels are addition and multiplication, but more advanced combinations could also be used, such as exponentiation [Rasmussen and Williams (2006)]. In particular the sum kernel is defined via $k_{sum}(X,Y) = k_1(X,Y) + k_2(X,Y)$, while the product kernel is defined via $k_{product}(X,Y) = k_1(X,Y) \times k_2(X,Y)$.
Figure 2.6 shows how the new kernel inherits the properties of its two constituent kernels, both in the case of addition and multiplication. In this case the constituent kernels are RBF and Periodic, but the same operation can be made for all types of kernel.

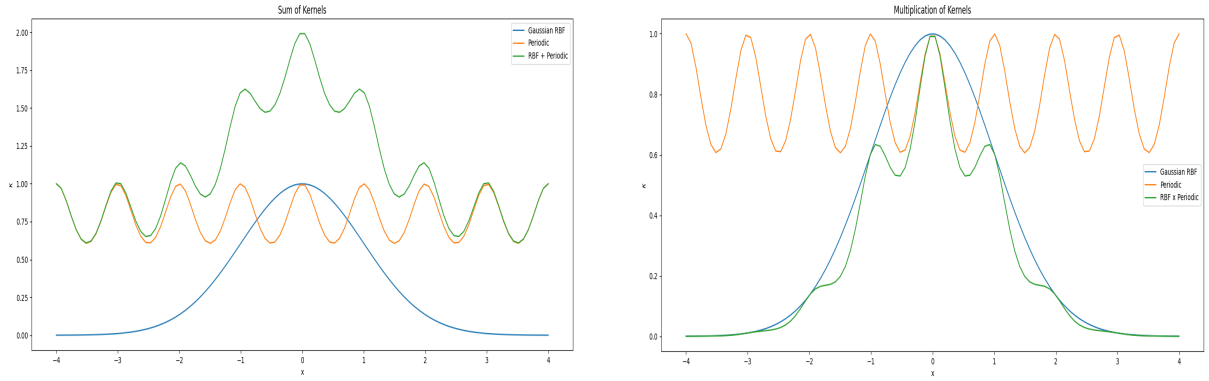



*Figure 2.6: Combination of Gaussian RBF kernel and Periodic kernel. Results of the combinations can be seen in green.*

## 2.2.4 Definition of Gaussian Process Latent Variable Model

The objective of the Gaussian Process Latent Variable Model is to perform dimensionality reduction. In order to work, GPLVM assumes that the observed data, lying in the high dimensional space, is generated from a lower-dimensional data $X$. The model will therefore

learn the lower dimensional representation $X \in \mathbb{R}^{N \times Q}$ of the dataset, that can be presented in the higher dimensional space as a matrix $Y \in \mathbb{R}^{N \times D}$, where $N$ stands for the number of units in our training sample, and $D$ represents the dimensionality of the training sample, having $Q \ll D$.
The mapping function allows points to be projected from the observation space $Y$ to the low-dimensional latent space in the following way:

$$y_i = f(x_i) + \varepsilon,$$

where $y_i$ is a sample point in $Y$, $x_i$ is the latent variable projected in $X$, $\varepsilon$ is the noise with a Gaussian distribution $\varepsilon \sim \mathcal{N}(0, \sigma^2)$ and $f$ is the mapping function with Gaussian Process (GP) prior $f \sim \mathcal{GP}(0, K)$. Given that GPLVM is a probabilistic model, it can be represented by a directed graph, as shown in Figure 2.7.

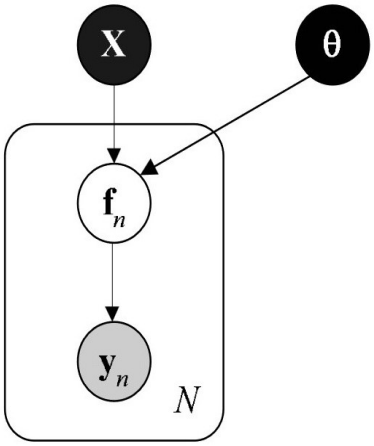



*Figure 2.7: The Gaussian Process Latent Variable Model. $Y$ is the observational space, $X$ is the latent space and $\theta$ are the hyperparameters of the mapping function $f$ [Lawrence and Hyvärinen (2005)].*

Indeed, in Lawrence's formulation of GPLVM [Lawrence and Hyvärinen (2005)], the prior is specified on the mapping and not over the latent locations, and the marginal likelihood is specified over the formulated mapping. The marginal likelihood $p(Y|X, \theta)$ can be obtained by combining the likelihood of the data with the prior, and integrating over $f$:

$$\begin{aligned} p(Y|X, \theta) &= \prod_{j=1}^{D} \frac{1}{(2\pi)^{\frac{1}{2}} |K|^{\frac{1}{2}}} \exp\left(-\frac{1}{2} y_{:,j}^T K^{-1} y_{:,j}\right) \\ &= \int p(Y|f) p(f|X, \theta)\, df \end{aligned}$$

where $\theta$ denotes the hyperparameters of the kernel function and $y_{:,j}$ denotes the $j$-th column of matrix $Y$. Thus, it is possible to maximize the marginal likelihood with respect to $X$ and the hyperparameter $\theta$:

$$\begin{aligned} \{\hat{\mathbf{X}}, \hat{\theta}\} &= \arg\max_{X,\theta} p(Y|X, \theta) \\ &= \arg\max_{X,\theta} \int p(Y|X, f, \theta) p(f)\, df \end{aligned}$$

Learning in the GPLVM framework consists of maximising the likelihood of the data with respect to the locations of the latent variables $\mathbf{X}$ and the hyperparameter $\theta$ of the process, in order to obtain the optimal latent space $\hat{\mathbf{X}}$ and the optimal set of hyperparameters $\hat{\theta}$ [Ek and Lawrence (2009)], [Li and Chen (2016)], [Gopalan (Gopalan)].

### Initialisation of the model

In the definition of the model it is important to initialise the position of the points in latent space. This is necessary because otherwise there is the risk that, during the optimisation, the model will get trapped in a local optimum that does not recover the true embedded space. PCA could be used for this purpose. However, for certain datasets, PCA can provide a poor initialisation. For this reason, it is common to use the Isomap algorithm for inizializing the model close to the global optima (more details on the functioning of the Isomap algorithm are presented in Appendix B). In this manner it is possible to recover the underlying structure and then provide a probabilistic description of the data through the GPLVM.
Using this approach it is possible to combine the strengths of the Isomap algorithm, which can provide a unique solution, with those of the GPLVM, which provides a simple way to calculate the mapping function, and which, because of its probabilistic nature, allows for comparison of different models through their Log-Likelihood [Ek and Lawrence (2009)], [Lawrence and Hyvärinen (2005)], [Gopalan (Gopalan)].

### Automatic Relevance Determination (ARD)

In order to provide even more flexibility to the model, it is possible to use Automatic Relevance Determination (ARD). This technique allows a different length-scale parameter for every input dimension. The optimal value for each of these will be obtained throughout the optimization of the model. Determining the relevance of each parameter automatically allows the model to assess the importance of the latent dimensions, treating the length-scale as a scaling parameter and identifying its value for each dimension. This also allows for setting to zero the weights of unnecessary dimensions and enabling the number of latent dimensions required to be determined in a systematic way [De Boi et al. (2023)], [Gopalan (Gopalan)]. This, represented in Figure 2.8, is often referred to as "switching off" dimensions in the latent space.
All the kernels presented in subsubsection 2.2.3 can implement the functionality of the ARD. For example, it is possible to achieve an ARD RBF kernel with covariance function defined as follows:

$$k_{\text{ARD-RBF}}(x, x') = \sigma^2 \exp\left(-\frac{1}{2}\|x - x'\|^\top \mathbf{L} \|x - x'\|\right), \tag{2.10}$$

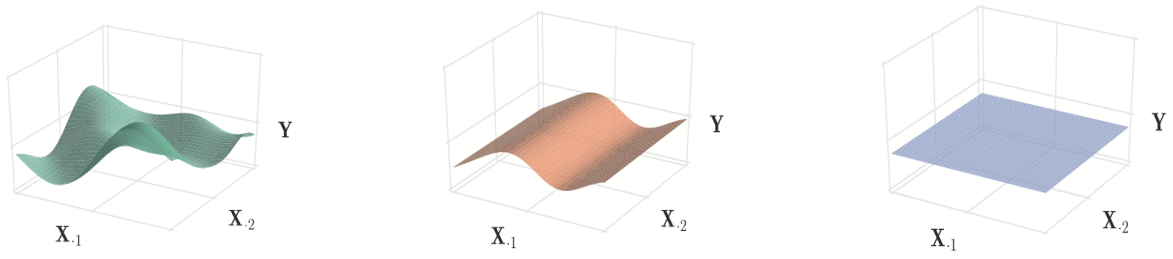
where $\mathbf{L}$ is a diagonal matrix containing the length-scale parameter of each dimension. Each element $\ell_\mathbf{i}$ of the diagonal corresponds to the length-scale of the i-th input dimension. A large $\ell_\mathbf{i}$ value means that the i-th dimension is relevant, while for a lower $\ell_\mathbf{i}$ value the relevance of the i-th dimension is inferior. It is also possible to get the ARD verison of Matérn kernel,

defined as follows:

$$k_{\text{ARD-Matérn}}(x, x') = \sigma^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left( \mathbf{L}\sqrt{2\nu}\|x - x'\| \right)^{\nu} K_{\nu} \left( \mathbf{L}\sqrt{2\nu}\|x - x'\| \right), \tag{2.11}$$

where $\mathbf{L}$ is defined as above. The same holds for the Exponential kernel, that in the ARD version has covariance function:

$$k_{\text{ARD-Exponential}}(x, x') = \exp\left(-\mathbf{L}\|x - x'\|\right). \tag{2.12}$$




*Both $X_1$ and $X_2$ are "switched on"* | *Only $X_2$ has been "switched off"* | *Both $X_1$ and $X_2$ are "switched off"*

*Figure 2.8: Three examples of a Gaussian process with ARD RBF kernel to show the effect of the scaling parameter on the input dimension. The scaling parameters where chosen to be 1 for "on" and $10^{-10}$ for "off" [Zwiessele (2017)].*

### Optimisation of the model

Having defined the model and having initialised the latent space with one of the proposed techniques, it is necessary to find the optimal values for the latent variables, the kernel hyperparameters, and the noise variance. It is possible to use different algorithms for the optimization. In particular, [GPy (2012)], the python package used in this project, offers the possibility of choosing between Scaled Conjugate Gradient (SCG), Truncated Newton Conjugate-Gradient (TNC) and Limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS-B). For this project the last of the three will be used. More details on the functioning of the L-BFGS-B algorithm are provided in Appendix C.

## 2.3 Comparison between PCA and GPLVM

Summarising, PCA is a linear dimensionality reduction technique that aims to preserve as much variance as possible, providing an interpretable lower-dimensional space, while GPLVM is a non-linear technique that catches much more complex relationships, but which may result in less interpretable latent representations, and it is at risk of being computationally expensive. Furthermore, a key advantage of the GPLVM is that it is incorporated in a probabilistic framework, meaning that there is a probability associated with the training data, allowing for comparison of Likelihood values.
In conclusion, it can be said that GPLVM is a non-linear probabilistic extension of PCA [Lawrence and Hyvärinen (2005)].

# Chapter 3

# Datasets used

As mentioned in section 1.1 the analysis will be first performed on a simulated dataset (section 3.1) and then on a real one (section 3.3). A test dataset will be used in order to evaluate the performance of the dimensionality reduction achieved on the simulated dataset (section 3.2).

## 3.1 Simulated Cylinder dataset

As shown in Figure 3.1, the Simulated dataset consists of a 3-dimensional Cylinder made of 20 equidistant points per circumference, with 15 equidistant circumferences. This results in a total of 300 points. Each column of points is represented with a different colour in order to have a better understanding of the mapping performed between the lower dimensional space and the observed space. In this way is possible to "keep track" of how each point is transformed between the two spaces.

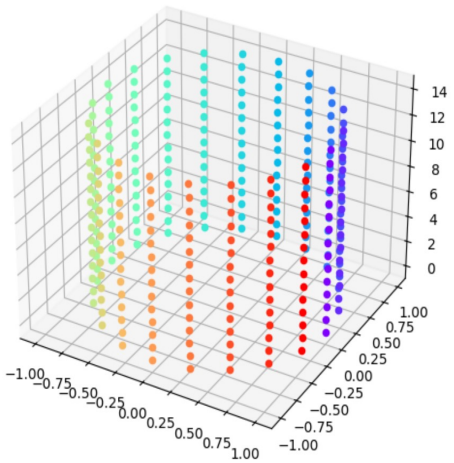



*Figure 3.1: The Simulated Cylinder dataset in the observed 3-dimensional space.*

## 3.2 Test dataset for the Cylinder: Helix dataset

In order to visualise how good the reconstruction of the simulated Cylinder is, a test dataset will be used. As shown in Figure 3.2 this set consists of 21 points creating an Helix that completes a full revolution around the original Cylinder. The distance between the points was chosen so that each point lies on a column of points of the cylinder. Also in this case

each point is represented with a different colour according to on which column of points in the cylinder the helix point originally lies.

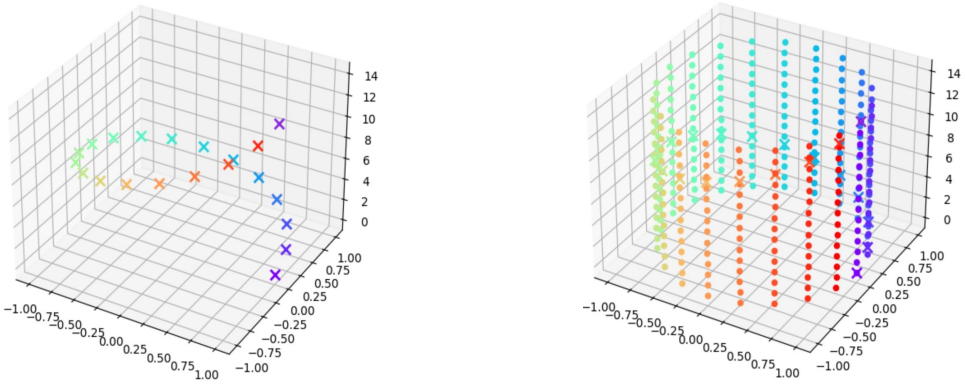


*Figure 3.2: On the left the representation of the Helix test dataset, on the right the same dataset together with the simulated Cylinder.*

## 3.3 Real Handwritten Digits dataset (MNIST)

The Modified National Institute of Standards and Technology database [LeCun, Cortes, and Burges (LeCun et al.)] is a database of black-and-white digits from 0 to 9 written by high school students and employees of the United States Census Bureau. Examples of the digits and the distribution of them in the dataset are shown in Figure 3.3.

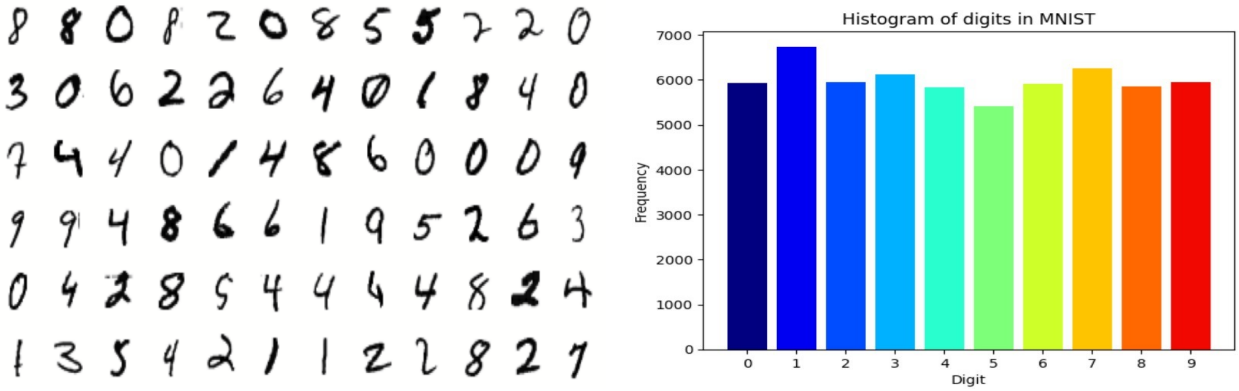



*Figure 3.3: Sample images and distribution of the digits in the MNIST train dataset.*

It consists of 60000 training images and 10000 test images, along with the corresponding labels indicating which digit is represented. The digits have been size-normalized and centered in a fixed-size image so that each of them can be represented by a 28x28 matrix, where each entry corresponds to a pixel, and has a value from 0 to 255 according to the pixel's colouring, where 0 is white, 255 is black, and all other values are shades of grey.
To perform the analysis, a new training set of 5000 images has been created by randomly sampling from the original training set, as the digits are more or less equally present in the original data set, as shown in Figure 3.3. The same procedure was repeated for the test dataset in order to obtain a smaller test dataset consisting of 500 images.

# Chapter 4

# Results for the Cylinder dataset

In this section the results of the analysis performed on the simulated cylinder dataset will be presented, starting with those obtained using Principal Component Analysis (section 4.1) and following with those obtained using Gaussian Process Latent Variable Model (section 4.2). A comparative evaluation of the differences and performance of the two techniques will consequently be carried out (section 4.3).

## 4.1 Outcomes achieved through PCA

Dimensionality reduction using PCA was performed on the Simulated Cylinder dataset presented in section 3.1. The three principal components explained the same proportion of total variability present in the dataset, as shown in Figure 4.1.

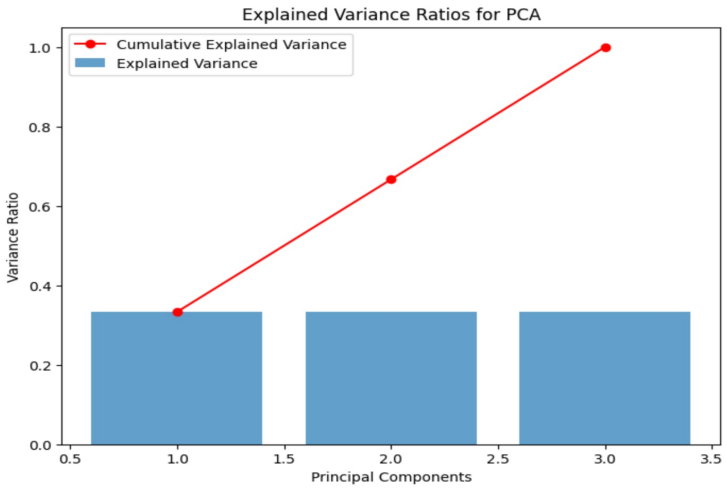
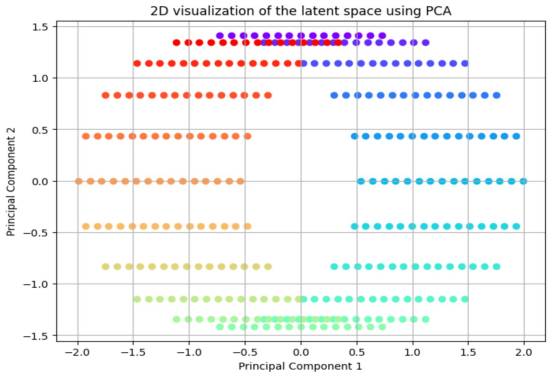
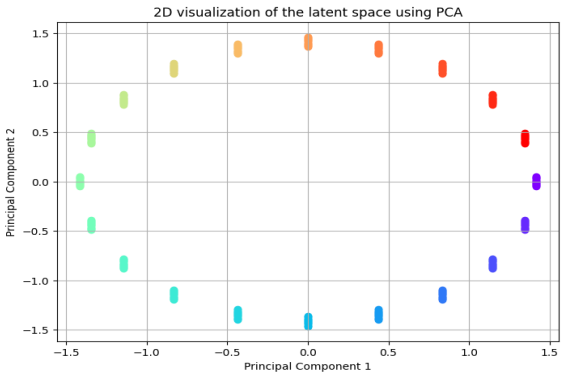



*Figure 4.1: Amount of total variance explained by each Principal Component and Cumulative Explained Variance.*

This means that choosing to keep just the first two principal components, resulting in a latent space of 2 dimensions, will lead to a loss of a third of the total variability. Furthermore, since all of the principal components explain the same amount of variability, there is no need to use the first two of them, but any pair will be a valid choice. Because of the great loss of variability during the dimensionality reduction, the representation of the latent space obtained through PCA will be poor. This can be better understood looking at Figure 4.2, where two possible representation of the latent space of the Cylinder are presented.



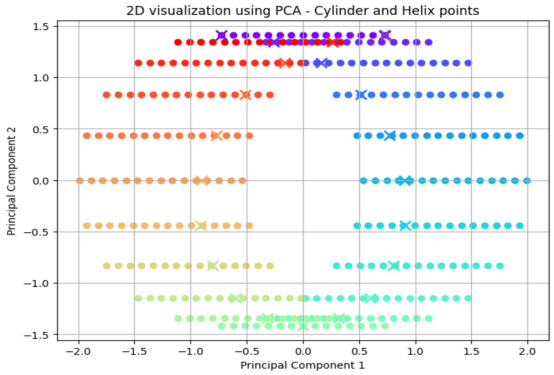

*Latent Space A*



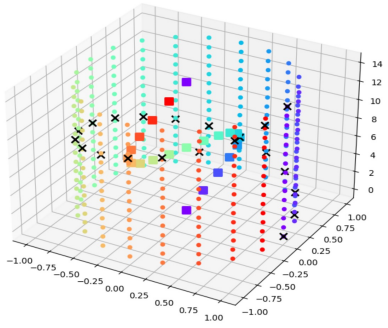

*Latent Space B*

*Figure 4.2: Two equivalent representations of the latent space of the Cylinder obtained using PCA, depending on which two principal components are used.*

In the Latent Space A there is a great amount of confusion: at the top and bottom of the graph there is an overlap of points despite the fact that they have different colours, so they should be far apart from each other. In the Latent Space B the height dimension of the original Cylinder is totally lost, and the points are mapped into the latent space as if they were being observed from above. In order to perform a deeper and more precise assessment, the Helix test dataset discussed in section 3.2 is used.
The transformation that allowed the mapping to the lower dimensional space was also applied to the Helix test set. In order to evaluate the performance of the dimensionality reduction is then sufficient to see whether each helix point in the latent space is close to the points of the Cylinder, lying in the latent space as well, and having the same colour. If instead the evaluation is intended to be carried out after the reconstruction from the latent space to the original space, it is sufficient to examine the distance between the original points of the helix test dataset and the reconstructed ones. This can be better understood by analysing Figure 4.3.




*Latent Space of the Cylinder with transformed test Helix points.*



*Helix test points reconstructed in the original space with the Cylinder. Original Helix points in black.*

*Figure 4.3: Useful plots in order to evaluate the performances of the PCA in the process of dimensionality reduction, using the Helix test dataset presented in section 3.2.*

From the plot on the left in Figure 4.3 we can see that the Helix points are mapped consistently in the latent space of the Cylinder, lying very close to the cylinder points with the same colour. Nevertheless it is difficult to understand what is going on in the upper and lower part of the plot, where all the points are very close one to another, causing confusion. This results in an extremely poor reconstruction of the Helix in the original space. In fact, as can be seen looking at the right plot in Figure 4.3, the reconstructed Helix points plotted in coloured "■" marker are very far apart compared to the original Helix points plotted in black "**x**" marker. In conclusion, in order to compare the result obtained using PCA with the ones obtained using GPLVM will be useful to calculate the Mean Squared Error (MSE) for the Helix test dataset, that in this case is $MSE = 0.1746032$, as shown in Table 4.1. This value has no meaning when taken individually, but it will be useful as a criterion for comparing the performance of the different methods used.

| | **MSE** |
|---|---|
| **PCA** | 0.1746032 |

Table 4.1: MSE of the Helix test dataset after reconstruction using PCA.

The reasoning that was conducted leads to the conclusion that PCA is unsuitable for performing dimensionality reduction of the analysed dataset.

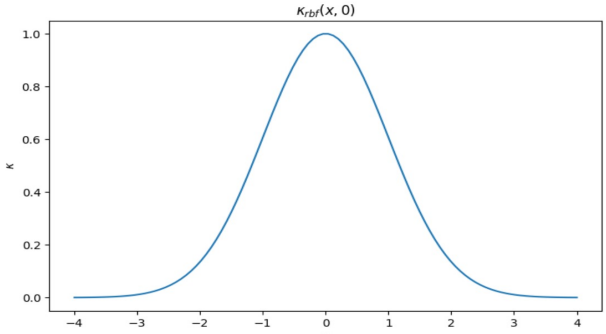
## 4.2 Outcomes achieved through GPLVM

To perform dimensionality reduction using GPLVM the first thing to be done is to specify the features of the mapping function, namely the characteristics of the Gaussian Process.
As described in chapter 2, different kernels (or combination of multiple kernels) will be used for adapting the Gaussian Process to the dataset which is analysed, for it to best reflect the properties of the data. In order to speed up the optimisation process for finding the optimal latent space $\hat{\mathbf{X}}$, the latter will be initialised using other dimensionality reduction techniques such as Isomap or even the PCA itself.
Both the choice of the kernel and the type of initialisation used will be based on trial and error. Automatic Relevance Determination will additionally be exploited, so that the choice of the most suitable length-scale will be carried out automatically. Also in this case, as was done when reducing the dimensionality using PCA in section 4.1, the Helix test dataset will be used for testing the performance of the technique, with the help of MSE's calculation. Since the GPLVM is inserted in a probabilistic framework, also comparison of Log-Likelihood will be carried out in order to evaluate the performance of the different models.

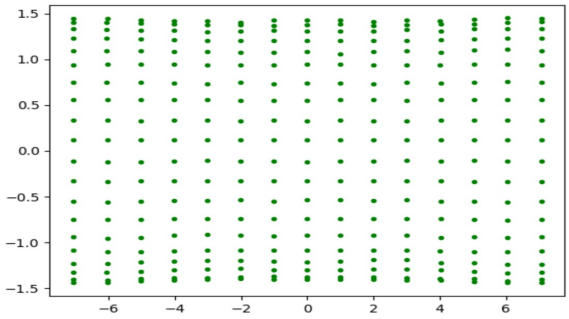
### 4.2.1 Best outcome obtained through GPLVM

Since the choices for the refinement of the model were made by trial and error, multiple results were obtained. This section will present the best result obtained, achieved using the RBF kernel shown in Figure 4.4a) and initialising the latent space $\mathbf{X}$ using the Isomap algorithm with 20 neighbours (shown in Figure 4.4b). The value of 20 is the best choice for the number

of neighbours in the Isomap algorithm because the Cylinder consists of circumferences with 20 points each. ARD was used allowing for a different length-scale parameter for every input dimension. This could also be used to automatically determine the number of dimensions to be retained in the latent space, but in this case two dimensions will be used anyway, so that the lower dimensional space can be properly visualised. Using these settings leads to the latent space presented in Figure 4.5a, with the transformed Helix points plotted with the "**x**" marker and the latent space of the Cylinder plotted with the "•" marker. This lower dimensional space is very clear and makes it simple to understand how the mapping function is working: it is as if the three-dimensional Cylinder was vertically cut and unfolded on the two-dimensional space. The Helix points are also mapped correctly, leading to the highest value obtained for the Log-Likelihood during the analysis, as presented in Table 4.2.

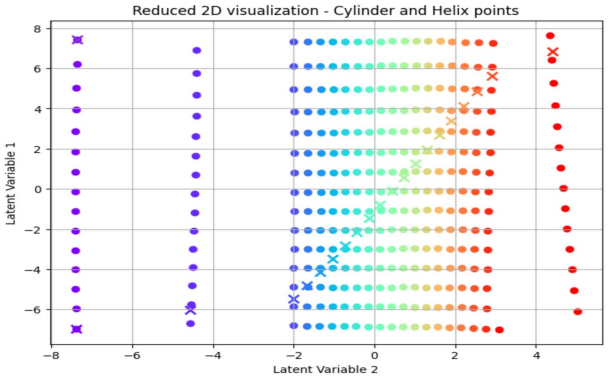



*(a) Covariance function of the RBF kernel used.*

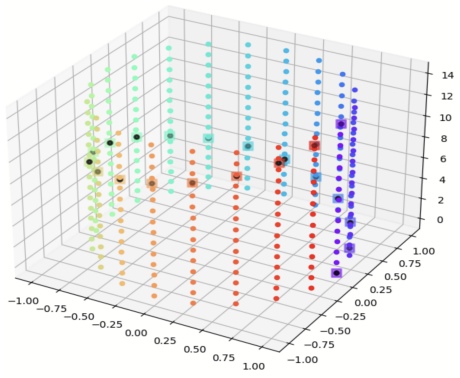


*(b) Initialised latent space obtained using Isomap algorithm with 20 neighbours.*

*Figure 4.4: Best setting for GPLVM*




*(a) Latent Space of the Cylinder with transformed test Helix points.*



*(b) Helix test points reconstructed in the original space with the Cylinder.*

*Figure 4.5: Best model, Log-Likelihood = 6475*

The reconstruction of the Helix points in three-dimensional Cylinder space is almost perfect, presenting a value for the MSE = 0.0000404, and as visible in Figure 4.5b, since the reconstructed Helix points plotted in "■" marker are exactly overlapping with the original Helix points plotted in black with the "•" marker.

### 4.2.2 Comparison of results obtained with different settings

Various results were obtained from the several attempts conducted. In this section some of them will be presented in order to show how, as the GPLVM settings change, the latent space obtained differs. For each model used, on the left will be presented the covariance function of its kernel, in the center the initialisation of its latent space, and on the right the latent space obtained after the optimisation.
In Figure 4.7 and in Figure 4.8 it is possible to see how, if two different kernels are used while keeping everything else fixed, the results can be totally diverse. In fact, by using a Matérn 5/2 kernel instead of an Exponential kernel, it is possible to obtain a much clearer latent space, leading to an higher value for the Log-Likelihood (see Figure 4.7 and Table 4.2). However in the latent space obtained in the subsection 4.2.1 and presented in Figure 4.5a, a more precise mapping can be noticed, while in Figure 4.7 some columns of points are still mapped in a confusing way, inverting the order of colours.
The latent space shown in Figure 4.8 is instead completely inappropriate: it is difficult to understand the mapping between the two spaces, and confusion prevails, leading to an incredibly low value for the Log-Likelihood (see Table 4.2).
For the latent space represented in Figure 4.9, the same initialisation was used, namely with Isomap algorithm and 20 neighbours, but for the kernel an RBF multiplied by a Periodic kernel was chosen, resulting in the covariance function presented on the left in Figure 4.9. Using these settings a high value for the Log-Likelihood is obtained (see Table 4.2), but the respective latent space results to be unclear, with columns of points that do not respect the colour structure of the cylinder.
If, on the other hand, starting from the setting that led to the best results (see Figure 4.5a) PCA is used instead of Isomap algorithm for initialisation, a final latent space is obtained which still carries some features typical of the latent space obtained with PCA, such as the circular shape of the mapped points, as represented in Figure 4.10. In this case, although the value of the Log-Likelihood is very good (see Table 4.2), the representation of the latent space is not particularly clear, which leads to a preference for other models with a similar value for the Log-Likelihood and the MSE. Automatic Relevance Determination has always been used because it has been seen to greatly improve model performance.

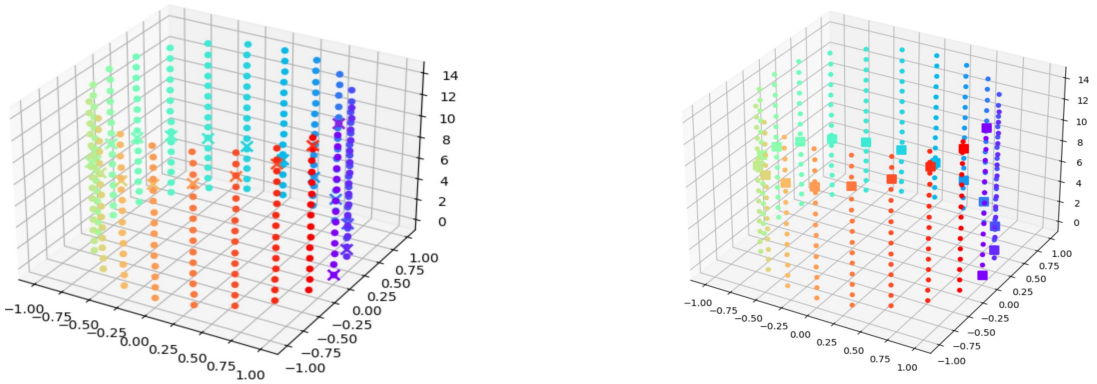
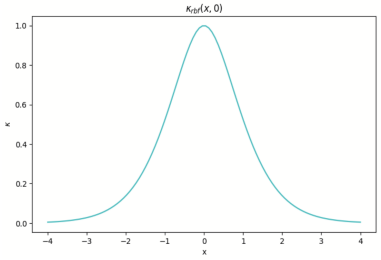
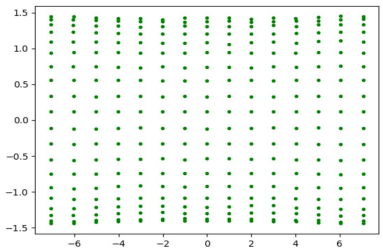


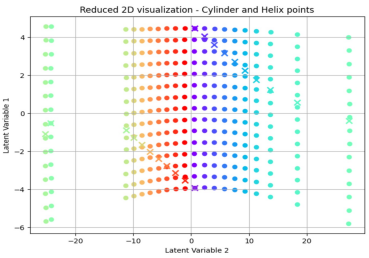
*Figure 4.6: Original Helix test dataset (left) and its reconstruction with GPLVMs (right).*




*Matérn 5/2 kernel*

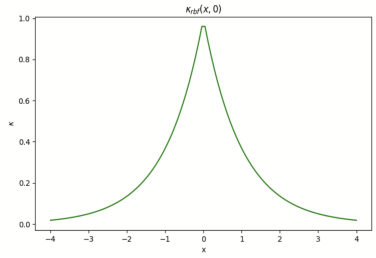


*Isomap with 20 neighbours*

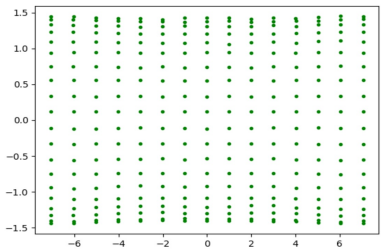



*Latent space obtained*

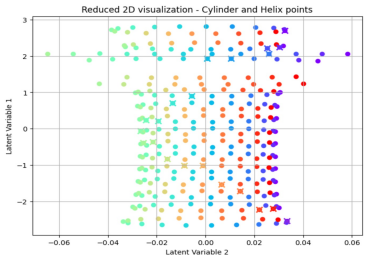
*Figure 4.7: Model A*




*Exponential kernel*

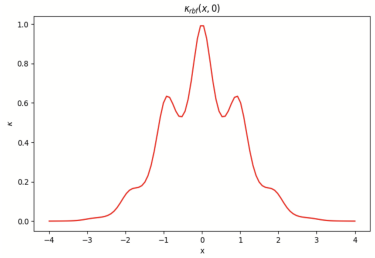


*Isomap with 20 neighbours*

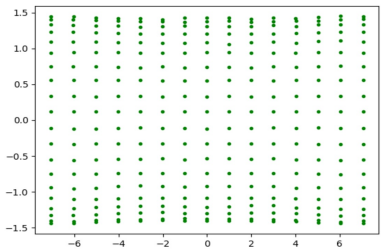



*Latent space obtained*

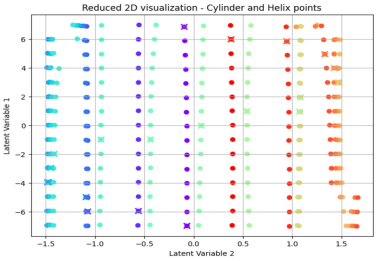
*Figure 4.8: Model B*




*RBF × Periodic kernel*

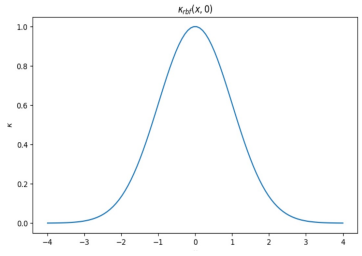


*Isomap with 20 neighbours*

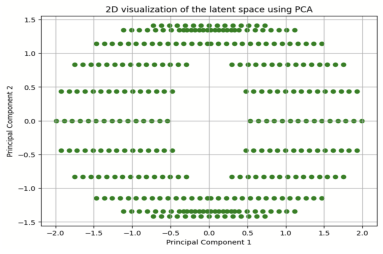



*Latent space obtained*

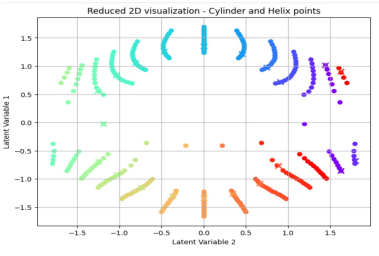
*Figure 4.9: Model C*




*RBF kernel*




*PCA*




*Latent space obtained*

*Figure 4.10: Model D*

| | MSE | Log-Likelihood |
|---|---|---|
| **Best model** | 0.0000404 | 6475 |
| **Model A** | 0.0010639 | 5954 |
| **Model B** | 0.0269841 | 467 |
| **Model C** | 0.0269844 | 5424 |
| **Model D** | 0.0000666 | 5638 |

Table 4.2: MSE of Helix test dataset after reconstruction and Log-Likelihood using different settings of GPLVM. Green values are preferable to yellow values.

In conclusion, it is interesting to note that, regardless of the latent spaces, Likelihood and MSE values obtained with the different models, when the Helix points are mapped back in the three-dimensional space of the Cylinder, all the models can achieve a correct reconstruction of the Helix test dataset, as shown in Figure 4.6. Calculating the MSE, however, it is possible to see that the reconstructions differ in very small details, which may not be visible to the human eye. Also in this case, as presented in Table 4.2, the best reconstruction, characterized by the lowest MSE, is the one obtained with the Best Model, presented in Figure 4.4.
However, in this context it is of interest to obtain a clearly interpretable latent space, in order to be able to understand how the mapping function operates without confusion. For this reason, in choosing the best model, more than the value of the log likelihood or the quality of the reconstruction given by the MSE, the latent space obtained played a fundamental role.

## 4.3 Comparison of results: PCA and GPLVM

The theoretical differences between PCA and GPLVM stated in section 2.3 are reflected in the different results obtained with the two techniques. The result obtained with PCA is unsatisfying and incredibly poor compared to those achieved through GPLVM which are much more flexible, also because they do not have the linearity constraint for the mapping function, and since they can be tailored to the specific properties of the analysed dataset. This is also emphasized by the different values of the MSE for the two techniques, as shown in Table 4.3.

| | MSE |
|---|---|
| **PCA** | 0.1746032 |
| **Best model** | 0.0000404 |

Table 4.3: MSE of Helix test dataset after reconstruction using PCA and the best settings for GPLVM. Green values are preferable to yellow values.

However, it must be emphasised that $10^5$ iterations were performed for the optimisation of the GPLVMs. In some cases the model reached convergence of the parameters before these $10^5$ iterations, resulting in a saving of time, but in many other cases the model optimisations took up to one hour. The same waiting time was not necessary to obtain results through PCA: using this method the results were provided instantly, since it is enough to find the solution of an eigenvalue problem.

# Chapter 5

# Results for the MNIST dataset

In this section the results of the analysis performed on the Real Handwritten Digits dataset (MNIST) will be presented, starting with those obtained using Principal Component Analysis (section 5.1) and following with those obtained using Gaussian Process Latent Variable Model (section 5.2). A comparative evaluation of the differences and performance of the two techniques will consequently be carried out (section 5.3).

## 5.1 Outcomes achieved through PCA

Dimensionality reduction using PCA was performed on the MNIST dataset presented in section 3.3. In Figure 5.1 the amount of variance explained by the first 20 Principal Components is presented. It is possible to see that the first two PCs account for a really small portion of the total variability (namely 10.3%). In order to obtain decent results for the reconstruction (usually the aim is to obtain a latent space that accounts for 90% of the original variability) it would be necessary to use more than just two PCs, but this cannot be done in this context since the aim is to get a bi-dimensional latent space.

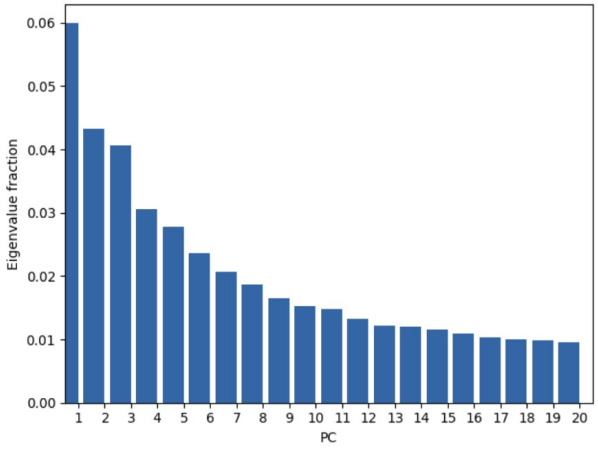



*Figure 5.1: Amount of total variance explained by each PC.*

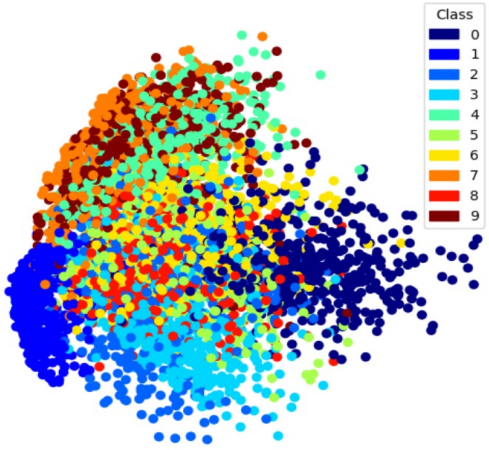



*Figure 5.2: Latent space obtained using PCA.*

This will result in a loss of almost 90% of the variability, leading to a very poor performance of the PCA in the given context. In fact, in the obtained latent space which is presented in Figure 5.2, the different digits are close and often overlapping to one another, proving the difficulty in separating them in the latent space by creating clusters of matching digits.

However, this does not turn out to be true for the digit 0, which is concentrated in a small area of the latent space. The transformation that allows the mapping to the lower dimensional space is now also applied to the test set presented in section 3.3. In order to evaluate the performance of the dimensionality reduction one random digit in the test dataset will be mapped in the latent space and then reconstructed in the original space. The latent space with the mapped test digit is presented in Figure 5.3, with the "**x**" marker with the color corresponding to the given digit. It can be seen that the test digit, despite appearing to be a 5, is extremely close to different digits. This will make the process of reconstruction of the image representing the test digit much more complex, leading to unsatisfactory results, as shown in Figure 5.4.

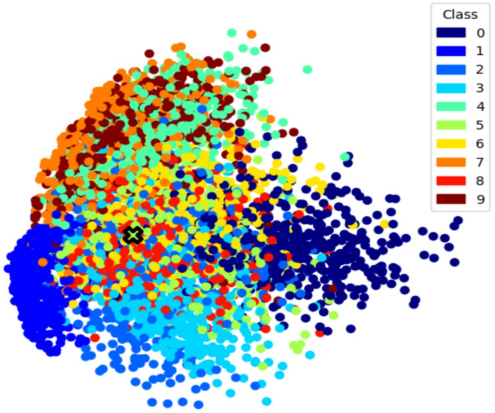



*Figure 5.3: Latent space with the mapped test digit represented with the "**x**" marker in green, corresponding to the digit 5.*

*Figure 5.4: Original (left) and reconstructed (right) test image.*

In conclusion, as was done for the simulated dataset in section 4.1, in order to compare the result obtained using PCA with the ones obtained using GPLVM, the Mean Squared Error for all the 500 images of the test dataset has been calculated, resulting to be $MSE = 3599.8274$, as shown in Table 5.1. It must be remembered that also in this case this value has no meaning when taken individually, but only when related to the values of the different techniques used.

| | **MSE** |
|---|---|
| **PCA** | 3599.8274 |

Table 5.1: MSE of the test dataset after reconstruction using PCA.

The analysis conducted in this section leads to the conclusion that PCA is unsuitable for performing dimensionality even in this scenario.

## 5.2 Outcomes achieved through GPLVM

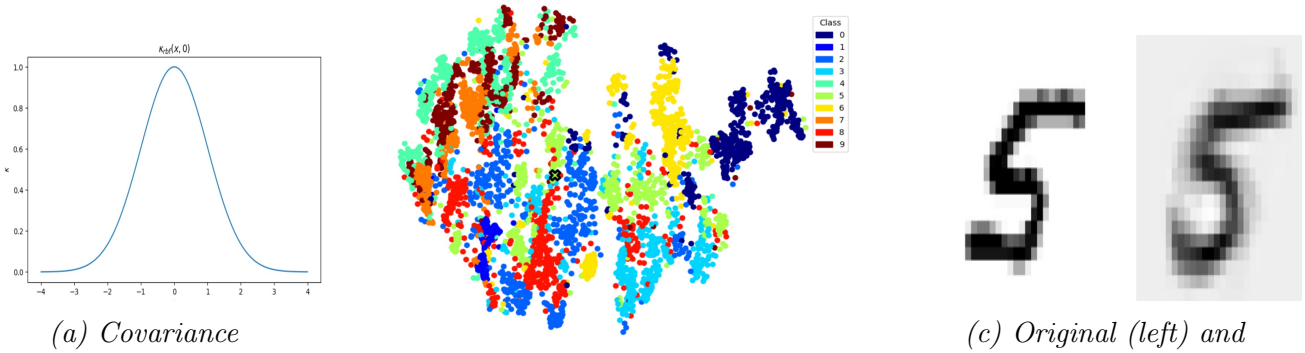
The structure of the analysis of this dataset will be very similar to that designed for the simulated dataset in section 4.2. Different kernels will be used to perform the dimensionality

reduction of this dataset, in order to adapt the Gaussian Process to the features of the dataset, and by initialising the optimal latent space $\hat{\mathbf{X}}$ using other dimensionality reduction techniques such as Isomap or even the PCA itself in order to speed up the optimisation of the model. Automatic Relevance Determination will always be used since it was observed to considerably improve the performance of the model (see subsection 4.2.2). The test dataset will then be used for testing the performance of the technique by mapping one digit in the latent space and reconstructing the corresponding image in the original space, implementing the calculation of the MSE and comparing the Log-Likelihood values for the different models.

### 5.2.1 Models defined

Also in this case the choices for the refinement of the model were made by trial and error, resulting in multiple outcomes. In this section they will be presented and analysed, with the purpose of discerning the optimal outcome. For training the different models the same sub-dataset of 5000 units was used, with an optimisation process consisting of 5000 iterations. For each model will be presented the kernel covariance function, the latent space of digits with a random digit from the test dataset mapped, and the corresponding original and reconstructed test image.

- **Model A** (see Figure 5.5) was obtained using the settings that resulted to be most effective in the Simulated Cylinder framework, namely an RBF kernel (see Figure 5.5a) with Isomap initialisation. In this case the best choice for the number of neighbours in the Isomap algorithm resulted to be 10, since there are 10 distinct digits in the dataset.
- **Model B** (see Figure 5.6) is defined starting from Model A and using PCA for the initialisation instead of the Isomap algorithm. Everything else remains unchanged.
- **Model C** (see Figure 5.7) is also defined starting from Model A, but a Matérn 5/2 kernel was used instead of the RBF kernel.
- **Model D** (see Figure 5.8), like model C, differs from model A only in the type of kernel used. In this case, an Exponential kernel was chosen.

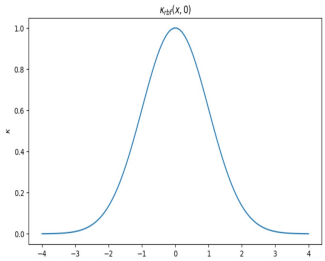



*(a) Covariance function of the RBF kernel used.*

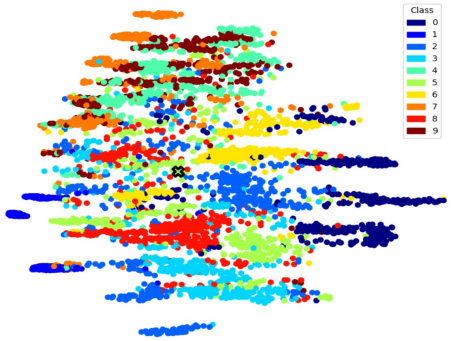
*(b) Latent Space with a random digit from the test dataset mapped.*

*(c) Original (left) and reconstructed (right) test image.*

*Figure 5.5: Model A*



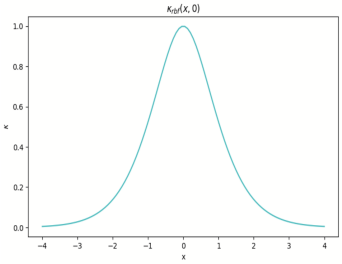

*(a) Covariance function of the RBF kernel used.*



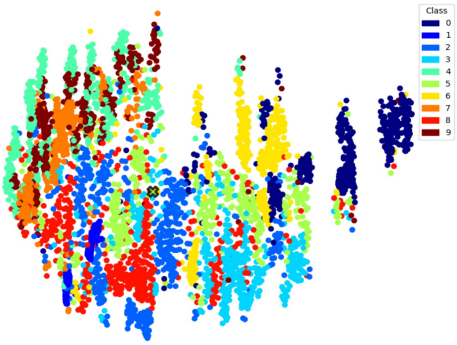

*(b) Latent Space with a random digit from the test dataset mapped.*

*(c) Original (left) and reconstructed (right) test image.*

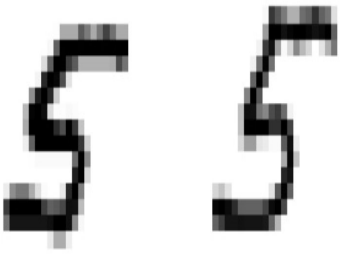
*Figure 5.6: Model B*




*(a) Covariance function of the Matérn 5/2 kernel used.*

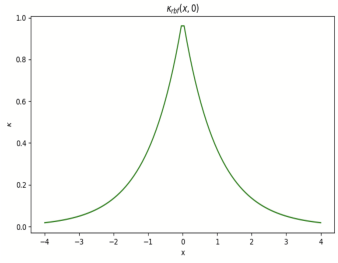



*(b) Latent Space with a random digit from the test dataset mapped.*

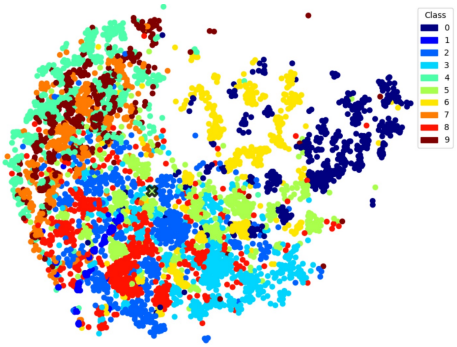


*(c) Original (left) and reconstructed (right) test image.*

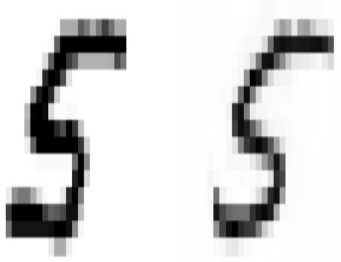
*Figure 5.7: Model C*




*(a) Covariance function of the Exponential kernel used.*




*(b) Latent Space with a random digit from the test dataset mapped.*



*(c) Original (left) and reconstructed (right) test image.*

*Figure 5.8: Model D*

### 5.2.2 Comparison of results obtained with different models

With regard to **Model A**, it is possible to see that the mapping function is able to separate the different digits in the latent space, creating clusters, as shown in Figure 5.5b. These clusters are not exhaustive, meaning that for the same digit multiple clusters are present in the latent space. This may be due to the fact that there are multiple "styles" for writing the same digit. However, some of these groups are still very close to each other and not always extremely tightly concentrated. This may lead to confusion in the reconstruction, which could turn out not to be completely clear. In Figure 5.5c it is indeed possible to notice that the background is not entirely free from noise, blending in with the digit.
The same reasoning applies for **Model B**. In Figure 5.6b it is possible to see that the resulting low-dimensional space is organised differently with respect to the previous one, due to the use of a different technique for the initialisation of the latent space. Despite this difference, the reconstruction turns out to be very similar to the previous one, presenting some background confusion and not extremely precise borders for the digit (see Figure 5.6c). However, unlike what happened with the Simulated Cylinder dataset, initializing the latent space with PCA instead of the Isomap algorithm did not result in significant derangement.
The issues highlighted for the first two models are mitigated in **Model C**, where the Matérn 5/2 kernel was used. The different clusters in the latent space are more separated from each other and more dense within themselves, as shown in Figure 5.7b. This results in a reconstruction without background noise and with contours that are much more clearly defined, as shown in Figure 5.7c.
A good reconstruction, presented in Figure 5.8c, is also obtained using **Model D**. This is achieved through the use of the Exponential kernel shown in Figure 5.8a, which results in the definition of a latent space that is, however, not as good as could be expected from the quality of the reconstruction performed. This in fact displays less clustered groups of different digits. The points are mapped in a slightly more confusing manner compared to what happens using Model C.

**Conclusions**

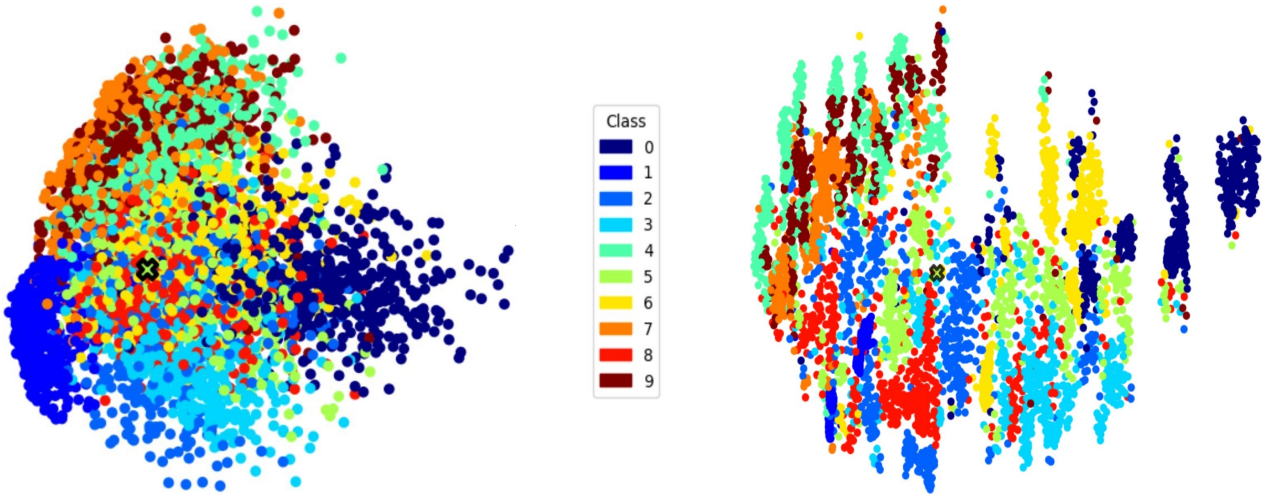
After this analysis and with the help of Table 5.2, where the values for MSE and Log-Likelihood of all models are presented, it is possible to conclude that the preferred model should be Model C. In fact, this model not only leads to the most accurate reconstruction and presents the most clustered latent space of all the models, it also provides the lowest MSE value and the highest value for the Log-Likelihood.

| | MSE | Log-Likelihood |
|---|---|---|
| **Model A** | 1926.76547 | -20453463.773 |
| **Model B** | 1942.27933 | -20490061.009 |
| **Model C** | 1876.58679 | -20416410.033 |
| **Model D** | 2064.22316 | -20445616.083 |

Table 5.2: MSEs of the test dataset after reconstruction and Log-Likelihood values using different GPLVM settings. Green values are preferable to yellow values.

## 5.3 Comparison of results: PCA and GPLVM

Even in the context of the Real Handwritten Digits dataset, the theoretical differences between PCA and GPLVM stated in section 2.3 are evident in the dissimilar results obtained with the two techniques. In Figure 5.9 it is evident that the latent space obtained with GPLVM is much better organised, presenting multiple clusters for the different digits, while in the latent space obtained using PCA the different digits are overlapping each other. The test digit in the lower-dimensional space obtained through PCA is very close to different digits, and may be confused under reconstruction in the original space. The same does not happen when using GPLVM.




*(a) Latent space obtained through PCA.*

*(b) Latent space obtained through the best GPLVM setting.*

*Figure 5.9: Comparison of the Latent Space obtained with the two techniques. The test digit mapped is represented with the "x" marker.*

*(a) Test digit.*

*(b) Test digit reconstructed with PCA.*

*(c) Test digit reconstructed with GPLVM.*

*Figure 5.10: Test digit and its reconstruction with the two techniques.*

The difference in performance of the two techniques is even more evident when analysing Figure 5.10, in which the test digit, the reconstruction performed by PCA, and that obtained with GPLVM are presented side by side. In Figure 5.10b the background is very confused, making it difficult to understand which digit is shown. It is up to interpretation, and could be argued that a number 9 is depicted, while the true number represented is a 5, as can be immediately perceived by looking at Figure 5.10c. In this image the digit does indeed stand out with well-defined contours and a clear background.
This also emerges when looking at Table 5.3, where the MSE value after the reconstruction of the test dataset is presented for both techniques. Using PCA, the MSE value is in fact almost double that obtained using GPLVM.

| | **MSE** |
|---|---|
| **PCA** | 3599.8274 |
| **Model C** | 1876.5868 |

Table 5.3: MSEs of the test dataset after reconstruction using PCA and the best GPLVM setting. Green values are preferable to yellow values.

Additionally, it should be noted that each optimisation needed to obtain the optimal latent space using GPLVM required waiting times of more than 24 hours, sometimes without even achieving convergence. Indeed, as mentioned in [Lawrence and Hyvärinen (2005)], "GPLVM is a viable alternative to other non-linear visualisation approaches for small data sets". The time required for optimisation, indeed, increases with the size of the training dataset. This is why a sub-sample of 5000 units was used.
The same amount of time is not necessary when using PCA, which requires incredibly less computing power.

# Chapter 6

# Conclusions

After having performed dimensionality reduction using PCA and GPLVM on both the Simulated Cylinder dataset presented in section 3.1 and the Real Handwritten Digits dataset (MNIST) presented in section 3.3, the superiority of GPLVM for performing aforesaid tasks is evident.
The incredible flexibility when defining the model makes it possible to create a tool that adapts to the characteristics of the analysed dataset and the context-specific requirements. This results in an excellent performance not only with regard to dimensionality reduction, where the model manages to retain an important part of the original variability without losing useful information, but also when reconstructing the test dataset from the low-dimensional space to its original dimensionality.
PCA, on the other hand, is a completely rigid technique, which does not allow for adaptation to the context. Its strengths are the very fast time it takes to perform dimensionality reduction and its easier mathematical formulation, which allows it to be used with knowledge even by less experienced users. However, if the objective is to obtain a latent space in which the functioning of the mapping function is visibly evident, and which is capable of retaining the vast majority of the original variability even with the use of only a few dimensions, this is definitely not the best technique to use. Similarly, if it is of interest to perform classification in a latent space with few dimensions, or if it is desired to obtain a reconstruction of test datasets in the original space that are very close to the truth, GPLVM is to be preferred to PCA.

## 6.1 Limitations

One of the major limitations of this project is certainly the limited computing power of the devices used. Particularly in the case of the analysis of the Real Handwritten Digits dataset (MNIST), the already significant optimization times have made it impossible to use the entire training dataset, leading to the choice of using a sub-sample of 5000 out of the 60000 units available. It can be imagined that if the training dataset was used in its entirety, it would be possible to obtain a model with more accurate and stable performances, increasing further the difference in effectiveness between the two techniques used. Another limitation is the challenging objectivity in selecting the best model, as there are multiple features to evaluate, which could sometimes lead to conflicting conclusions in identifying the optimal model. Indeed, it would be would be simplistic to choose the model with the highest Log-Likelihood and the lowest MSE, as the organisation of the latent space also plays an important role, and the same MSE can sometimes be misleading, particularly for the Real Handwritten Digits dataset. This is because the MSE is calculated on the matrix in which

the image has been encoded, and small differences in color, insignificant and imperceptible to the human eye, or a shift to the right of the reconstructed image, could have a significant impact on the MSE value, while having no effect on the quality of the image reconstruction.

## 6.2 Future work

Future work that could be done is to use techniques that, building on the two methods discussed in this project, develop them further, such as Probabilistic Principal Component Analysis [Tipping and Bishop (1999)], Discriminative GPLVM (see [Urtasun and Darrell (2007)]) or Supervised GPLVM (see [Jiang et al. (2011)]). Furthermore, it could be possible to proceed by testing the various methodologies on datasets other than those examined in this project, with the aim of observing whether the difference in performance remains constant irrespective of the dataset analysed or varies as the latter changes.
Finally, the simplest and most obvious way to continue developing this work would be to dedicate more time to optimization, using a sub-sample composed of more units.

# Bibliography


De Boi, I., C. H. Ek, and R. Penne (2023). Surface approximation by means of gaussian process latent variable models and line element geometry. *Mathematics 11*(2).

Dholakiya, P. (2023). Principal component analysis.

Ek, C. H. and P. Lawrence (2009). *Shared Gaussian process latent variable models*. Ph. D. thesis, Citeseer.

Gopalan, N. Gaussian process latent variable models for dimensionality reduction and time series modeling.

GPy (since 2012). GPy: A gaussian process framework in python. `http://github.com/SheffieldML/GPy`.

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

Jiang, X., J. Gao, T. Wang, and L. Zheng (2011). Supervised latent linear gaussian process latent variable model for dimensionality reduction. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 41*(2), 425–434.

Lawrence, N. and A. Hyvärinen (2005). Probabilistic non-linear principal component analysis with gaussian process latent variable models. *Journal of machine learning research 6*(11).

LeCun, Y., C. Cortes, and C. J. Burges. The mnist database of handwritten digits.

Li, P. and S. Chen (2016). A review on gaussian process latent variable models. *CAAI Transactions on Intelligence Technology 1*(4), 366–376.

Pearson, K. (1901). Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin philosophical magazine and journal of science 2*(11), 559–572.

Rasmussen, C. E. and C. K. I. Williams (2006). *Gaussian Processes for Machine Learning*. MIT Press.

Saxena, A., A. Gupta, and A. Mukerjee (2004). Non-linear dimensionality reduction by locally linear isomaps. In *International Conference on Neural Information Processing*, pp. 1038–1043. Springer.

Simon, H. A. (1996). *The sciences of the artificial*. MIT press.

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

Tipping, M. E. and C. M. Bishop (1999). Probabilistic principal component analysis. *Journal of the Royal Statistical Society Series B: Statistical Methodology 61*(3), 611–622.

Urtasun, R. and T. Darrell (2007). Discriminative gaussian process latent variable model for classification. In *Proceedings of the 24th international conference on Machine learning*, pp. 927–934.

Zhu, C., R. H. Byrd, P. Lu, and J. Nocedal (1997). Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on mathematical software (TOMS) 23*(4), 550–560.

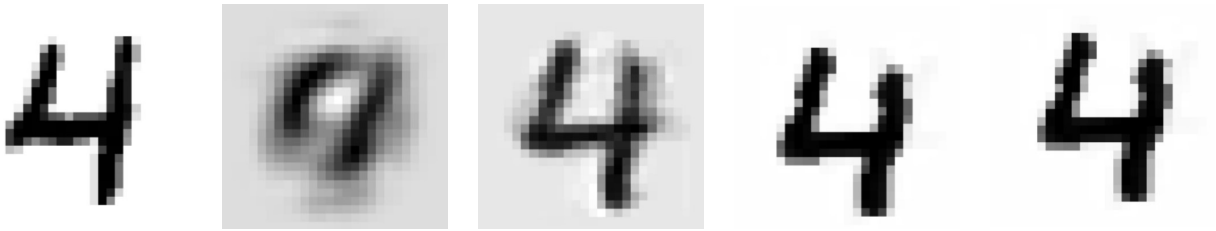
Zwiessele, M. (2017). *Bringing models to the domain: Deploying gaussian processes in the biological sciences.* Ph. D. thesis, University of Sheffield.

# Appendices

# Appendix A

## More results

### Test digits and their reconstructions with different techniques.



*Original* *PCA* *Model A* *Model B* *Model C*

*Original* *PCA* *Model A* *Model B* *Model C*

*Original* *PCA* *Model A* *Model B* *Model C*

*Form this Figures is possible to see once again that the reconstruction performances of the GPLVM are excellent compared to those obtained with PCA.*

# Appendix B

## Isometric Feature Mapping Algorithm (Isomap)

Isomap is a nonlinear dimensionality reduction technique that aims to preserve the intrinsic geometric structure of high-dimensionality data in a low-dimensionality space.
The idea behind this technique is to measure the distance between two separate points on the manifold (called the Geodesic Distance), and by preserving these distances in the lower-dimensional embedding, to unfold the underlying manifold structure of the data. The functioning of the algorithm can be decomposed into the following steps:

- Neighbors of each point are determined as points which are within the $\epsilon$ distance (or using K-nearest neighbor approach). These neighborhood relations are represented as a weighted graph G over the data points, with edges of weight $d_X(i, j)$ between neighboring points.

- The Geodesic Distances $d_M(i, j)$ is estimated for all pairs of points on the manifold M by computing their shortest path distances $d_G(i, j)$ (using Floyd-Warshall's algorithm or Dijasktra algorithm) in the graph G, capturing the intrinsic manifold structure of the data.

- Reduce the dimensionality of the data by constructing a Low-Dimensional Embedding using the Multidimensional scaling algorithm (MDS) on the computed shortest path distance matrix, preserving the pairwise Geodesic Distances as closely as possible.

The value for the residual error of the MDS algorithm is used to determine the quality of the performance of the Isomap algorithm. The lower-dimensional embedding is selected if the residual error is lower than a certain threshold [Tenenbaum et al. (2000)], [Saxena et al. (2004)].

# Appendix C

## L-BFGS-B Algorithm

The L-BFGS-B algorithm extends L-BFGS to minimize a nonlinear function of $n$ variables, $minf(x)$ subject to $l \leq x \leq u$, where the vectors $l$ and $u$ represent lower and upper bounds on the variables. Tthis extension of the previous algorithm is also appropriate and efficient for solving unconstrained problems. For the algorithm to work the gradient $g$ must be provided, but no knowledge of the Hessian matrix of f is necessary, making the algorithm very useful in cases where the Hessian is difficult to compute or is dense.
The algorithm operates as follows: At each iteration a limited-memory BFGS approximation to the Hessian is updated. This limited-memory matrix is used to define a quadratic model of the objective function f. A search direction is then computed using a two-stage approach: first, the gradient projection method is used to identify a set of active variables, then the quadratic model is approximately minimized with respect to the free variables. The search direction is defined to be the vector leading from the current iterate to this approximate minimiser. Finally a line search is performed along the search direction. It is remarkable that the limited-memory BFGS matrices are represented in a compact form that is efficient for bound-constrained problems [Zhu et al. (1997)].

# Appendix D

## Code used for the analysis

Listing D.1: Code for Best model (Simulated Cyilinder)

```
1
2  from mpl_toolkits.mplot3d import Axes3D
3  import matplotlib.pyplot as plt
4  from matplotlib import cm
5  import numpy as np
6  import math
7  import pickle
8  import string
9  import GPy
10 np.random.seed(6)
11
12 # load model from pickle file
13 model_pkl_file = "MODEL_BEST_CYL.pkl"
14 with open(model_pkl_file, 'rb') as file:
15  m_opm = pickle.load(file)
16
17  theta = np.linspace(0, 2.*np.pi, 20, endpoint=False)
18 x = np.cos(theta)
19 y = np.sin(theta)
20
21 colors = plt.cm.rainbow(np.linspace(0, 1, 20))
22
23 fig = plt.figure(figsize=(6, 6))
24 ax = fig.add_subplot(111, projection='3d')
25
26 Y = []
27
28 for z in np.linspace(0, 14, 15):
29     for i in range(20):
30         ax.scatter(x[i], y[i], z, color = colors[i])
31         Y.append([x[i], y[i], z])
32
33 plt.show()
34
35 from sklearn import manifold
36 import pandas as pd
37 iso = manifold.Isomap(n_neighbors=20, n_components=2)
38 iso.fit(np.array(Y))
39 manifold_2Da = iso.transform(np.array(Y))
40 manifold_2D = pd.DataFrame(manifold_2Da, columns=['Component 1', 'Component 2'])
41
42 latent_dim = 2
43 lvm_dim = latent_dim
44 kern = GPy.kern.RBF(lvm_dim,ARD=True)
45 m_opm = GPy.models.GPLVM(np.array(Y), input_dim = lvm_dim, kernel=kern, X=manifold_2D)
46 print(manifold_2D.shape)
47 plt.plot(manifold_2Da[:,0],manifold_2Da[:,1],'.',color='g')
48
49 m_opm.optimize(messages=1, max_iters=100000)
50 params = m_opm.param_array
51
52 model_pkl_file = "MODEL_BEST_CYL.pkl"
```

```
53
54  with open(model_pkl_file, 'wb') as file:
55      pickle.dump(m_opm, file)
56
57  m_opm.log_likelihood()
58
59  fig, ax = plt.subplots(figsize=(8, 6))
60  latent_variables = m_opm.X.param_array
61
62  for i in range(20):
63      ax.scatter(latent_variables[i::20, 1], latent_variables[i::20, 0], color=colors[i])
64
65  plt.xlabel('Latent Variable 2')
66  plt.ylabel('Latent Variable 1')
67  plt.title('Reduced 2D visualization')
68  plt.grid(True)
69
70  plt.show()
71
72  from scipy.spatial import cKDTree
73
74  tree = cKDTree(list(zip(np.ravel(x), np.ravel(y), np.ravel(np.linspace(0, 14, 20)))))
75
76  num_points = 21
77
78  theta_helix = np.linspace(0, 2.*np.pi, num_points)
79
80  x_helix = np.cos(theta_helix)
81  y_helix = np.sin(theta_helix)
82  z_helix = np.linspace(0, 14, num_points)
83
84  fig = plt.figure(figsize=(6, 6))
85  ax = fig.add_subplot(111, projection='3d')
86
87  for z in np.linspace(0, 14, 15):
88      for i in range(20):
89          ax.scatter(x[i], y[i], z, color = colors[i])
90
91  for i in range(num_points):
92      if i == num_points - 1:
93          ax.scatter(x_helix[i], y_helix[i], z_helix[i], color='blueviolet', linewidth=2,
        marker="x", alpha=1,s=70)
94      else:
95          dist, idx = tree.query([x_helix[i], y_helix[i], z_helix[i]])
96          ax.scatter(x_helix[i], y_helix[i], z_helix[i], color=colors[idx % 20], linewidth=2,
        marker="x", alpha=1,s=70)
97
98  plt.show()
99
100 Y_helix = np.array(list(zip(x_helix, y_helix, z_helix)))
101
102 latent_variables_helix = m_opm.infer_newX(Y_helix)[0]
103
104 fig, ax = plt.subplots(figsize=(8, 6))
105
106 for i in range(20):
107     ax.scatter(latent_variables[i::20, 1], latent_variables[i::20, 0], color=colors[i])
108
109 for i in range(num_points):
110     if i == num_points - 1:
111         ax.scatter(latent_variables_helix[i, 1], latent_variables_helix[i, 0], color='
        blueviolet', linewidth=2, marker="x", alpha=1,s=70)
112     else:
113         dist, idx = tree.query([x_helix[i], y_helix[i], z_helix[i]])  # Find the closest
        point in the cylinder
114         ax.scatter(latent_variables_helix[i, 1], latent_variables_helix[i, 0], color=colors[
        idx % 20], linewidth=2, marker="x", alpha=1,s=70)
115
```

```
plt.xlabel('Latent Variable 2')
plt.ylabel('Latent Variable 1')
plt.title('Reduced 2D visualization - Cylinder and Helix points')
plt.grid(True)

plt.show()

Y_helix_pred = m_opm.predict(latent_variables_helix)[0]

fig = plt.figure(figsize=(8, 8))
ax = fig.add_subplot(111, projection='3d')

for z in np.linspace(0, 14, 15):
    for i in range(20):
        ax.scatter(x[i], y[i], z, color = colors[i])

for i in range(num_points):
    ax.scatter(Y_helix_pred[i, 0], Y_helix_pred[i, 1], Y_helix_pred[i, 2], color=colors[i
    %20], linewidth=2, marker="s", alpha=0.5, s=70)

for i in range(num_points):
        ax.scatter(x_helix[i], y_helix[i], z_helix[i], color='black', linewidth=2, marker=".
    ", alpha=0.7,s=70)  # Color it Violet

plt.show()

from GPy.util import pca
import numpy as np
p = pca.PCA(np.array(Y))

latent_spacePCA = p.project(np.array(Y))

plt.figure(figsize=(8, 6))
for i in range(20):
    plt.scatter(latent_spacePCA[i::20, 0], latent_spacePCA[i::20, 1], color=colors[i])

plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('2D visualization of the latent space using PCA')
plt.grid(True)
plt.show()

latent_spacePCA_helix = p.project(Y_helix)

plt.figure(figsize=(8, 6))

for i in range(20):
    plt.scatter(latent_spacePCA[i::20, 0], latent_spacePCA[i::20, 1], color=colors[i])

for i in range(num_points):
    if i == num_points - 1:
        plt.scatter(latent_spacePCA_helix[i, 0], latent_spacePCA_helix[i, 1], color='
    blueviolet', linewidth=3, marker="x",s=100)  # Color it Violet
    else:
        dist, idx = tree.query([x_helix[i], y_helix[i], z_helix[i]])
        plt.scatter(latent_spacePCA_helix[i, 0], latent_spacePCA_helix[i, 1], color=colors[
    idx % 20], linewidth=2, marker="x",s=100)

plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('2D visualization using PCA - Cylinder and Helix points')
plt.grid(True)
plt.show()

from sklearn.decomposition import PCA

pca_sklearn = PCA(n_components=2)
pca_sklearn.fit(np.array(Y))
```

```
180
181 latent_spacePCA = pca_sklearn.transform(np.array(Y))
182 latent_spacePCA_helix = pca_sklearn.transform(Y_helix)
183
184 Y_helix_predPCA = pca_sklearn.inverse_transform(latent_spacePCA_helix)
185
186 fig = plt.figure(figsize=(8, 8))
187 ax = fig.add_subplot(111, projection='3d')
188
189 for z in np.linspace(0, 14, 15):
190     for i in range(20):
191         ax.scatter(x[i], y[i], z, color = colors[i % 20])
192
193 for i in range(num_points):
194     ax.scatter(Y_helix_predPCA[i, 0], Y_helix_predPCA[i, 1], Y_helix_predPCA[i, 2], color=
    colors[i % 20], linewidth=2, marker="s", alpha=1, s=70)
195
196 plt.show()
```

Listing D.2: Code for Model C (Real digits)

```

import numpy as np
import pandas as pd
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
from matplotlib import cm
import numpy as np
import math
from sklearn import manifold
import string
import GPy
import matplotlib
from matplotlib.colors import LinearSegmentedColormap
from sklearn.decomposition import PCA
import pickle
from sklearn.metrics import mean_squared_error
from matplotlib.colors import LinearSegmentedColormap

cmap_inverted = LinearSegmentedColormap.from_list("InvertedGray", ["white", "black"])

test = pd.read_csv("C:/.../mnist_test.csv")
train = pd.read_csv("C:/.../mnist_train.csv")

# load model from pickle file
model_pkl_file = "MODEL_C.pkl"
with open(model_pkl_file, 'rb') as file:
 m_opm = pickle.load(file)

Y_TRAIN = train['label']
X_TRAIN = train.drop('label', axis = 1)

Y_TEST = test['label']
X_TEST = test.drop('label', axis = 1)

n = 5000
a = 500
N = 19
x_train = X_TRAIN.sample(n=n, random_state=42)
y_train = Y_TRAIN.sample(n=n, random_state=42)
x_test = X_TEST.sample(n=a, random_state=42)
y_test = Y_TEST.sample(n=a, random_state=42)

Y_TRAIN_ = Y_TRAIN.values.reshape(-1, 1)
y_train_ = y_train.values.reshape(-1, 1)
Y_TEST_ = Y_TEST.values.reshape(-1, 1)
y_test_ = y_test.values.reshape(-1, 1)

from sklearn.decomposition import PCA
PCA = PCA(n_components = 2)
x_pca = PCA.fit_transform(x_train)
test_pca = PCA.fit_transform(x_test)
test_rec_pca = PCA.inverse_transform(test_pca)

import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
import matplotlib.patches as mpatches

x_pca_y = np.hstack((x_train, y_train_))


unique_classes = np.unique(y_train_)
num_classes = len(unique_classes)
cmap = plt.cm.get_cmap('jet', num_classes)
legend_handles = [mpatches.Patch(color=cmap(i), label=str(unique_classes[i])) for i in range
    (num_classes)]
```

```

plt.figure(figsize=(8, 6))
scatter = plt.scatter(x_pca[:,0], x_pca[:,1], c=y_train_, cmap=cmap)
scatter = plt.scatter(test_pca[N,0], test_pca[N,1], c=y_test_[N,0],cmap=cmap, marker='X',s
    =150,linewidths=3,edgecolors='black')

plt.legend(handles=legend_handles, title='Class', loc=(0.85,0.52) )

plt.colorbar(scatter)
plt.axis('off')
plt.show()

fig, (ax1, ax2) = plt.subplots(1,2)
ax1.imshow(x_test.iloc[N].to_numpy().reshape(28,28),cmap=cmap_inverted)
ax2.imshow(test_rec_pca[N].reshape(28, 28),cmap=cmap_inverted)
fig.suptitle('Compression and decompression test dataset')
ax1.axis('off')
ax2.axis('off')
plt.show()

iso = manifold.Isomap(n_neighbors=10, n_components=2)
iso.fit(np.array(x_train))
manifold_2Da = iso.transform(np.array(x_train))
manifold_2D = pd.DataFrame(manifold_2Da, columns=['Component 1', 'Component 2'])

latent_dim = 2
lvm_dim = latent_dim
kern = GPy.kern.Matern52(lvm_dim,ARD=True)
m_opm = GPy.models.GPLVM(np.array(x_train), input_dim = lvm_dim, kernel=kern, X=manifold_2D)
     #X=manifold_2D ,x_pca
print(manifold_2D.shape)
plt.plot(manifold_2Da[:,0],manifold_2Da[:,1],'.',color='g')

m_opm.optimize(messages=1, max_iters=5000)
params = m_opm.param_array

m_opm.log_likelihood()

model_pkl_file = "MODEL_C.pkl"

with open(model_pkl_file, 'wb') as file:
    pickle.dump(m_opm, file)

x_gplvm = m_opm.X.param_array
x_rec_gplvm = m_opm.predict(x_gplvm)[0]
x_gplvm_y = np.hstack((x_gplvm, y_train_))
x_test_gplvm = m_opm.infer_newX(np.array(x_test))[0]
test_rec_gplvm= m_opm.predict(x_test_gplvm)[0]

plt.figure(figsize=(12, 8))
plt.scatter(x_gplvm[:,0], x_gplvm[:,1], c = y_train_, cmap='jet')
plt.scatter(test_rec_gplvm[N,0], test_rec_gplvm[N,1], c=y_test_[N,0],cmap=cmap, marker='X',s
    =150,linewidths=3,edgecolors='black', alpha=0.8)
plt.colorbar()
plt.axis('off')

num_classes = len(np.unique(y_train_))

cmap = plt.cm.get_cmap('jet', num_classes)
cmap = ListedColormap(cmap(np.linspace(0, 1, num_classes)))

legend_handles = [mpatches.Patch(color=cmap(i), label=str(i)) for i in range(num_classes)]
plt.legend(handles=legend_handles, title='Class', loc=(0.97,0.6))
plt.show()

fig, (ax1, ax2) = plt.subplots(1,2)

ax1.imshow(x_test.iloc[N].to_numpy().reshape(28,28),cmap=cmap_inverted)
```

```
ax2.imshow(test_rec_gplvm[N].reshape(28, 28),cmap=cmap_inverted)
fig.suptitle('compression and decompression')
ax1.axis('off')
ax2.axis('off')
plt.show()

from GPy.util import pca
import numpy as np
p = pca.PCA(np.array(x_train))
p.plot_fracs(20)

m_opm.kern.plot_ARD()

MSE_gplvm = mean_squared_error(x_test, test_rec_gplvm)
MSE_pca = mean_squared_error(x_test, test_rec_pca)
MSE_gplvm

MSE_pca
```