SCHOOL OF ECONOMICS AND MANAGEMENT

Analysing Functional Data of Two-Dimensional Arguments using Tensor Spline Orthonormal Bases

Freja Wikstrand

Supervised by: Krzysztof Podgórski

STAN40: First Year Master Thesis in Statistics
15 ECTS

Lund University
School of Economics and Management
Department of Statistics
Spring 2024

Abstract

This thesis establishes a theoretical framework for constructing orthonormal tensor spline bases, to transform two-dimensional functions into a coherent set of coefficients, for easier analysis of functional data. Through empirical testing on image datasets, the method showcases promising results, underscoring its utility in pattern recognition tasks. This research lays a foundation for further exploration into advanced data analysis techniques, with implications extending to multidimensional functional representations and computational modelling.

Contents

1 Introduction

Functional data is a fascinating field of study that delves into data consisting of entire functions rather than individual data points. This unique approach is believed to enable a richer understanding of complex, dynamic systems, making it particularly powerful in capturing and analysing variability over continuous domains. Functional data analysis provides a versatile framework for addressing real-world problems in diverse fields such as biology, economics, and engineering, offering insights that traditional data analysis methods might overlook.

In recent years, the emergence of deep learning has revolutionised the field of functional data analysis, offering powerful tools for extracting insights from complex and high-dimensional functional datasets. Deep learning techniques, such as neural networks, have shown promise in handling the intricate relationships and patterns present in functional data. However, despite their effectiveness, deep learning models often face challenges in interpretability, generalisation to new datasets, and robustness to noise and outliers.

Another method of working with functional data is to use splines, however when dealing with two-dimensional functions, the use of splines encounters challenges due to increased complexity and potential limitations. Splines, while effective in representing one-dimensional curves, can become computationally demanding and less intuitive when extended to two dimensions. Issues may arise in terms of defining appropriate knots, managing increased data dimensionality, and ensuring smoothness across both dimensions.

Balancing accuracy and computational efficiency becomes a delicate challenge, motivating the exploration of alternative methods for representing and analysing two-dimensional functions. This thesis aims to explore numerical approximation, evaluation, and analysis methods for splines applied to two-dimensional functions, while also considering the potential role of deep learning in augmenting these methods and overcoming their limitations.

2 Functional Data

Functional data refers to data where the observations are functions, as opposed to individual data points. Instead of having a set of numbers or vectors as your data, you have functions. Functional data analysis (FDA) is a statistical framework designed to analyse and make inferences about such functional data.

Let $F(t)$ be a random function representing our functional data. We can think of this as a function that takes time t as an input and gives us some observed value. In functional data analysis, we often decompose $F(t)$ into a mean function $\mu(t)$ and some fluctuation around the mean $\epsilon(t)$:

$$F(t) = \mu(t) + \epsilon(t) \tag{2.1}$$

Here, $\mu(t)$ represents the average behaviour of the functional data, and $\epsilon(t)$ captures the variability or randomness. Statistical methods in functional data analysis aim to estimate the mean function $\mu(t)$, model the variability $\epsilon(t)$, and make inferences about the underlying processes generating the functional data.

It is important to note that functional data analysis involves concepts from functional analysis and often uses tools like functional principal component analysis (FPCA) to analyse the variability in the data. Multidimensional functional data has more than one input, in this report the focus will be on two-dimensional data, defined as

$$F(x, y) = \mu(x_1, x_2) + \epsilon(x_1, x_2) \tag{2.2}$$

2.1 Hilbert Space of Square Integrable Functions

Our main objects are functions, as such, we need to study their properties. Functions can be viewed as elements in the Hilbert space of the square integrable functions. For a set $\mathcal{D} \subseteq \mathbb{R}^d$, where $\mathbb{R}^d$ is the d-dimensional space of real numbers, the Hilbert Space is denoted by

$$L^2 = \left\{ f : \mathcal{D} \to \mathbb{R}; \int_{\mathcal{D}} |f(x)|^2 dx < \infty \right\} \tag{2.3}$$

For the purpose of this thesis $d \in \{1, 2\}$, where $\mathcal{D} = [a, b]$, and $\mathcal{D} = [a, b] \times [c, d]$, respectively. The square integrable functions can be viewed as vectors in L^2, the norms and inner products for these functions is denoted

$$||f|| = \sqrt{\int_{\mathcal{D}} |f(x)|^2 dx} \tag{2.4}$$

$$< f, g >= \int_{\mathcal{D}} f(x)g(x)dx \tag{2.5}$$

From here we further state that f is orthogonal to g (hereafter denoted $f \perp g$), if $< f, g >= 0$, and f is normalised if $||f|| = 1$.

2.2 Orthonormal Basis Functions

In order to analyse functional data, we want to project it onto a subspace of L^2, spanned by a basis of functions. Let $\{e_i\}_{i=1}^{\infty} \subseteq L^2$ such that $\sum_{i=1}^{\infty} \alpha_i e_i \in L^2$, where $\alpha_i \in \mathbb{R}$. Then we have an infinite linear combination of functions. Using this, we can find a function

$$f \in \mathcal{H} = \left\{ \sum_{i=1}^{\infty} \alpha_i e_i, \quad \alpha_i \in \mathbb{R} \right\} \tag{2.6}$$

Here e_i is called a basis in $\mathcal{H}$, and any function f in that space can be found using a linear combination of the basis functions.

A good basis for inner product spaces is orthonormal (see [Debnath and Mikusinski, 1999, Chapter 3] for a more in-debt study into the benefits of an orthonormal basis when working with inner-product subspaces). That is

$$e_i \perp e_j, \quad i \neq j \tag{2.7}$$

$$||e_i|| = 1 \tag{2.8}$$

With an orthonormal basis, if $f \in \mathcal{H}$, then $f = \sum_{i=1}^{\infty} < f, e_i > e_i$ and $||f||^2 = \sum_{i=1}^{\infty} < f, e_i >^2$.

From here on, we assume all $\mathcal{H} \subseteq L^2$ to be finite dimensional, meaning there is a finite orthonormal basis in $\mathcal{H}$; $\{e_i\}_{i=1}^d$. Then any function in L^2 can be projected onto $\mathcal{H}$ by using the formula

$$P_{\mathcal{H}} f = \sum_{i=1}^{d} < f, e_i > e_i \quad \in \mathcal{H} \tag{2.9}$$

$P_{\mathcal{H}} f$ will henceforth be referred to as $\hat{f}$. In one dimension, we see that an orthonormal basis would yield

$$\hat{f} =< f, e > e = \int_a^b f(x)e(x)dx \cdot e \tag{2.10}$$

In two dimensions we want to choose an orthonormal basis e_1, e_2:

$$\begin{array}{ll} e_1(x_1) & e_2(x_2) \\ x_1 \in [a,b] & x_2 \in [c,d] \\ ||e_1|| = 1 & ||e_2|| = 1 \end{array}$$

$$e_1 \perp e_2$$

Then we get that

$$\hat{f} = \sum_{i=1}^{2} < f, e_i > e_i =< f, e_1 > e_1 + < f, e_2 > e_2 \tag{2.11}$$

$$= \int_c^d \int_a^b f(x_1, x_2)e_1(x_1)dx_1dx_2 \cdot e_1 + \int_c^d \int_a^b f(x_1, x_2)e_2(x_2)dx_1dx_2 \cdot e_2 \tag{2.12}$$

For the purpose of e_i outside the integral, we can treat it as a constant B_i, which is later substituted back into their original functional form.

$$= \int_c^d \int_a^b f(x_1, x_2)e_1(x_1)dx_1dx_2 \cdot B_1 + \int_c^d \int_a^b f(x_1, x_2)e_2(x_2)dx_1dx_2 \cdot B_2 \tag{2.13}$$

$$= \int_c^d \int_a^b (B_1e_1(x_1) + B_2e_2(x_2))f(x_1, x_2)dx_1dx_2 =< (B_1e_1 + B_2e_2), f > \tag{2.14}$$

3 Splines

A spline is a piece wise-defined curve that is formed by combining several simpler curves. It is like connecting dots with a smooth curve rather than a jagged line.

Suppose you have a set of data points (x, y), and you want to create a smooth curve $S(x)$ that passes through these points. Using n basis functions, which are points of references for finding the spline, the spline can be defined as:

$$S(x) = \sum_{i=1}^{n} C_i \cdot B_i(x) \tag{3.1}$$

Here C_i are coefficients and $B_i(x)$ are basis functions. These basis functions are typically chosen to be smooth and defined over small intervals. The coefficients C_i are determined such that the spline passes through the given points.

Using the theory from chapter 2, we can use our orthonormal basis such that $C_i =< f, e_i >$, and $B_i = e_i$. Many functions in functional data may be too complex to describe with one singular curve, and as such, using splines allows for finding changes in the function.

3.1 Tensor Space of Splines in $\mathbb{R}^2$

We assume we have two sets of orthonormal basis functions

$$OB_{i1}(x_1), \qquad x_1 \in [a, b], \quad i = 1, 2, ..., n_1 \tag{3.2}$$

$$OB_{j2}(x_2), \qquad x_2 \in [c, d], \quad j = 1, 2, ..., n_2 \tag{3.3}$$

such that

$$||OB_{i1}|| = 1 \quad \forall i, \qquad OB_{i1} \underset{i \neq i'}{\perp} OB_{i'1} \tag{3.4}$$

$$||OB_{j2}|| = 1 \quad \forall j, \qquad OB_{j2} \underset{j \neq j'}{\perp} OB_{j'2} \tag{3.5}$$

Now we define a space containing all the linear combinations of the first set, as well as one for the second set.

$$\mathcal{H}_1 = \left\{ \sum_{i=1}^{n_1} \alpha_i OB_{i1}, \quad \alpha_i \in \mathbb{R} \right\}, \quad \mathcal{H}_2 = \left\{ \sum_{j=1}^{n_2} \beta_j OB_{j2}, \quad \beta_j \in \mathbb{R} \right\} \tag{3.6}$$

In order to map two-dimensional data to one dimension, we also need to define the tensor product (denoted $\otimes$):

$$OB_{i1} \otimes OB_{j2} : [a,b] \times [c,d] \to \mathbb{R} \tag{3.7}$$

$$(OB_{i1} \otimes OB_{j2})(x_1, x_2) \stackrel{def}{=} OB_{i1}(x_1) OB_{j2}(x_2) \tag{3.8}$$

The new basis basis functions will then be a tensor product between the two dimensions, which are also orthonormal to eachother. This can be proven by finding the norm and inner products of the functions.

$$||OB_{i1} \otimes OB_{j2}|| = \int_c^d \int_a^b OB_{i1}(x_1) OB_{j2}(x_2) dx_1 dx_2 \tag{3.9}$$

$$= \int_a^b OB_{i1}(x_1) dx_1 \cdot \int_c^d OB_{j2}(x_2) dx_2 = ||OB_{i1}|| \cdot ||OB_{j2}|| = 1 \tag{3.10}$$

$$< OB_{i1} \otimes OB_{j2}, OB_{i'1} \otimes OB_{j'2} > \qquad (i,j) \neq (i',j') \tag{3.11}$$

$$= \int_c^d \int_a^b OB_{i1}(x_1) OB_{j2}(x_2) OB_{i'1}(x_1) OB_{j'2}(x_2) dx_1 dx_2 \tag{3.12}$$

$$= \int_a^b OB_{i1}(x_1) OB_{i'1}(x_1) dx_1 \cdot \int_c^d OB_{j2}(x_2) OB_{j'2}(x_2) dx_2 \tag{3.13}$$

$$=< OB_{i1}, OB_{i'1} > \cdot < OB_{j2}, OB_{j'2} >= 0 \tag{3.14}$$

Since at least one of the inner functions above is 0 due to orthogonality between bases and $(i,j) \neq (i',j')$, we can see that the inner product indeed becomes 0. We can then conclude that the resulting set

$$\mathcal{H}_1 \otimes \mathcal{H}_2 = lin\{OB_{i1} \otimes OB_{j2}, i = 1, ..., n_1, j = 1, ..., n_2\} \tag{3.15}$$

contains linear combinations of orthonormal square integrable functions, and is a subspace of L^2; the Tensor Space of Splines in $\mathbb{R}$. Then the projection of f onto $\mathcal{H}_1 \otimes \mathcal{H}_2$ becomes

$$\hat{f} = \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} < f, OB_{i1} \otimes OB_{j2} > OB_{i1} \otimes OB_{j2} \tag{3.16}$$

3.2 R-Package: Splinets

In the R-package splinets, splines are represented efficiently by expanding them using Taylor series at specific points called knots. This representation considers the areas of support, making it suitable for handling sparse functional data.

The goal is to find the best functional fit, in the least square sense, using splinets for data that consists of sampled values of functions or splines built over another set of knots. This optimal fit is then used for functional data analysis. The package can be used to find splines for one-dimensional data using an orthonormal basis. [Liu et al., 2023]

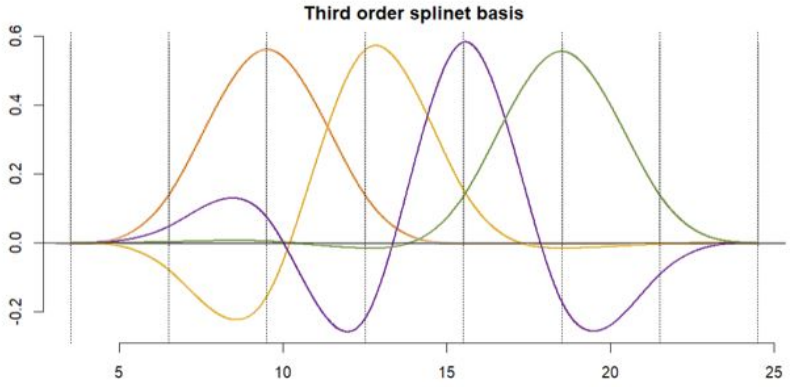

Figure 3.1: The figure presents an illustrative example of four orthonormal basis splines generated by Splinet in one dimension.

3.2.1 Splinet for Two Dimensions

Using the package, one-dimensional orthogonal basis splines can be found, as well as the coefficients when projection a function. It can also evaluate the spline for a new set of x_1 and x_2. However, Splinet at its current form can not do this for two dimensions.

$$\hat{f} = \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} < f, OB_{i1} \otimes OB_{j2} > OB_{i1} \otimes OB_{j2} \tag{3.17}$$

Looking at the above expression we start by finding a way to approximate $< f, OB_{i1} \otimes OB_{j2} >$. Since this is a double integral, it can be approximated

using double sums.

$$< f, OB_{i1} \otimes OB_{j2} > = \int_c^d \int_a^b f(x_1, x_2) OB_{i1}(x_1) OB_{j2}(x_2) dx_1 dx_2 \tag{3.18}$$

$$\approx \sum_{l=1}^{N_2} \sum_{k=1}^{N_1} f(x_{k1}, x_{l2}) OB_{i1}(x_{k1}) OB_{j2}(x_{l2}) \nabla x_1 \nabla x_2 \tag{3.19}$$

where $(a \leq x_{11} < x_{21} < ... < x_{N_1 1} \leq b)$, $(c \leq x_{12} < x_{22} < ... < x_{N_2 2} \leq d)$ and $\nabla x_1 = x_{(k+1)1} - x_{k1} \quad \forall k$, $\nabla x_2 = x_{(l+1)2} - x_{l2} \quad \forall l$.

To better calculate this, we can create matrices such that:

$$F_{N_1,N_2} = \begin{pmatrix} f(x_{11}, x_{12}) \nabla x_1 \nabla x_2 & \cdots & f(x_{11}, x_{N_2 2}) \nabla x_1 \nabla x_2 \\ \vdots & \ddots & \vdots \\ f(x_{N_1 1}, x_{12}) \nabla x_1 \nabla x_2 & \cdots & f(x_{N_1 1}, x_{N_2 2}) \nabla x_1 \nabla x_2 \end{pmatrix} \tag{3.20}$$

(3.21)

$$(OB_1)_{N_1,n_1} = \begin{pmatrix} OB_{11}(x_{11}) & OB_{11}(x_{21}) & \cdots & OB_{11}(x_{N_1 1}) \\ OB_{21}(x_{11}) & OB_{21}(x_{21}) & \cdots & OB_{21}(x_{N_1 1}) \\ \vdots & \vdots & \ddots & \vdots \\ OB_{n_1 1}(x_{11}) & OB_{n_1 1}(x_{21}) & \cdots & OB_{n_1 1}(x_{N_1 1}) \end{pmatrix} \tag{3.22}$$

(3.23)

$$(OB_2)_{N_2,n_2} = \begin{pmatrix} OB_{12}(x_{12}) & OB_{12}(x_{22}) & \cdots & OB_{12}(x_{N_2 2}) \\ OB_{22}(x_{12}) & OB_{22}(x_{22}) & \cdots & OB_{22}(x_{N_2 2}) \\ \vdots & \vdots & \ddots & \vdots \\ OB_{n_2 2}(x_{12}) & OB_{n_2 2}(x_{22}) & \cdots & OB_{n_2 2}(x_{N_2 2}) \end{pmatrix} \tag{3.24}$$

Then we can find our coefficients via matrix multiplication.

$$C_{n_1,n_2} = (OB_1)_{N_1,n_1} F_{N_1,N_2} (OB_2)^T_{N_2,n_2} \tag{3.25}$$

$$\approx \begin{pmatrix} < f, OB_{11} \otimes OB_{12} > & \cdots & < f, OB_{11} \otimes OB_{n_2 2} > \\ \vdots & \ddots & \vdots \\ < f, OB_{n_1 1} \otimes OB_{12} > & \cdots & < f, OB_{n_1 1} \otimes OB_{n_2 2} > \end{pmatrix} \tag{3.26}$$

In order to evaluate our spline for a new set x_1 and x_2, where $(a \leq x_{11} < x_{21} < ... < x_{M_1 1} \leq b)$, $(c \leq x_{12} < x_{22} < ... < x_{M_2 2} \leq d)$ we can once again use matrix multiplication.

$$S_{M_1,M_2} = (OB_1)^T_{M_1,n_1} C_{n_1,n_2} (OB_2)_{M_2,n_2} \tag{3.27}$$

such that $\hat{f}(x_{k1}, x_{l2}) = S_{M_1,M_2}(k, l)$. By using the above calculations, together with two sets of orthonormal basis-splines from Splinets, we can calculate splines for two-dimensional functions.

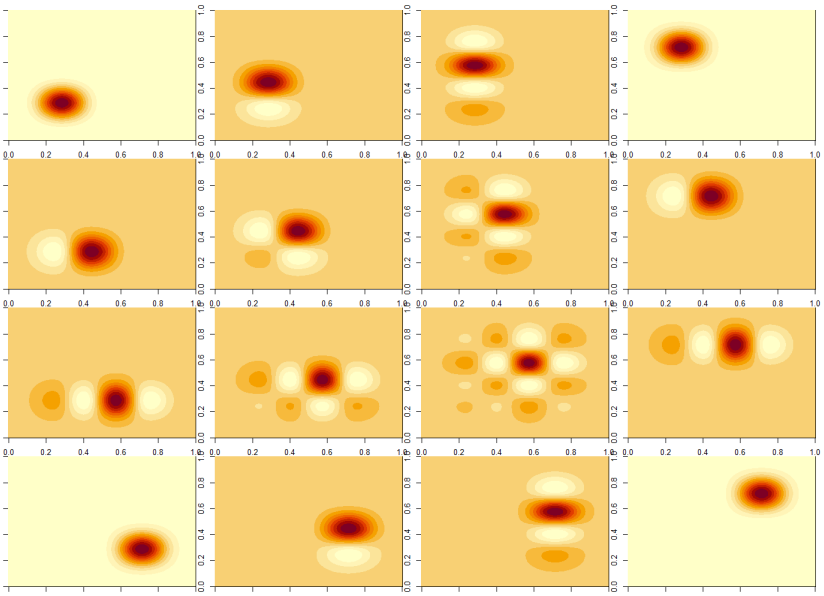

Figure 3.2: The figure depicts sixteen basis splines generated by combining the basis splines from Figure 3.1 along one axis with the same basis along the second axis. In the images, darker points represent higher values, while brighter points indicate lower values.

3.3 Analysis of Functional Data in Two Dimensions

Our method condenses each observation into a matrix of coefficients. Additionally, the utilisation of spline basis functions inherently captures the directional relationship between these coefficients. Consequently, the matrix representation of coefficients becomes redundant, as the splines already encode the directional information within the data.

Therefore, we can streamline our analysis by focusing directly on the individual coefficients. Each coefficient can be treated as a distinct variable, eliminating the need to maintain them in matrix form. By projecting additional observations onto the same subspace using identical basis splines, each observation consists of $n_1 \times n_2$ variables.

For further analysis, we can employ classical statistical methods, including hypothesis testing, regression analysis, and various classification techniques.

4 Experiment: Clothing Classification

The classification of clothing items from images is a fundamental task in computer vision with numerous practical applications, such as online retail, image retrieval, and fashion recommendation systems.

The objective of this experiment is twofold: first, to validate the performance of using two dimensional spline when classifying the images in our data, and see how it compares to previous methods within deep learning. Secondly, a brief study into how the choice of knots, and the order of smoothness for the splines will affect the performance.

Hopefully this method will be able to facilitate the development of more efficient and versatile applications in the domain of computer vision.

4.1 Data Presentation

The dataset used in this experiment consists of 70,000 images of clothing items, each resized to a resolution of 28×28 pixels. These images are categorised into ten distinct classes, representing various types of apparel: T-shirt, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, and Boot.

Each image is labelled with its corresponding class, facilitating supervised learning tasks such as classification and evaluation. The balanced distribution of samples across categories ensures that the model encounters an equitable representation of each class during training, minimising the risk of class imbalance biases.

To ensure an unbiased evaluation of our classification method, 10,000 images are set aside as a separate test dataset. This test data remains unseen by the model during training and validation stages, serving as an independent benchmark for assessing the generalisation performance of our approach.

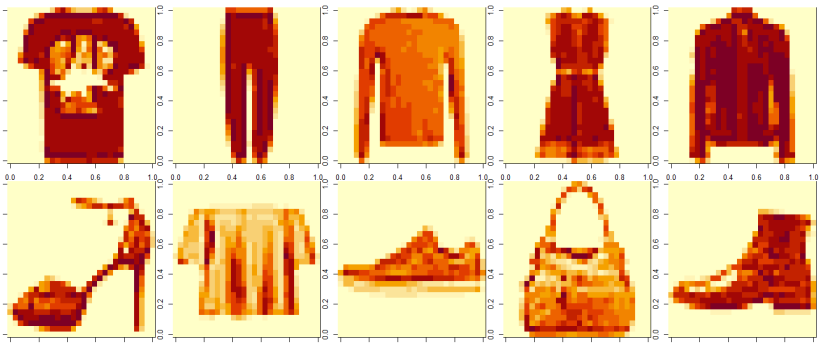

Figure 4.1: Each image above represents a sample from one of the classes in the dataset, showcasing the diversity of clothing items included in the experiment.

4.2 Method

This section outlines the methodology employed to develop an algorithm utilising Splines for the classification of images within the dataset. Leveraging the versatility and expressive power of Splines, we present a systematic approach to extracting discriminative features from image data, which hopefully facilitates more accurate and efficient classification.

4.2.1 Creating the Splines

Given the inherent characteristics of Splines, particularly their sensitivity to details at image edges, a preprocessing step is employed to mitigate potential information loss. To address this, each image undergoes a resizing procedure aimed at preserving edge details. Specifically, 3 pixels are added along the borders of the images, effectively expanding them to a size of 34×34 pixels. Consequently, the first and last 3 rows, as well as the first and last 3 columns, are padded with zeros. This resizing strategy ensures that crucial edge information is retained, thereby enhancing the overall fidelity of the image data and facilitating more accurate feature extraction and classification.

In the process of generating basis splines, two critical hyper-parameters demand consideration: the order of smoothness and the number of knots. These parameters play pivotal roles in shaping the characteristics of the resulting splines and consequently influence the overall flexibility and complexity of the spline basis. Notably, adjusting the order of smoothness governs the degree of curvature accommodated by the splines, while varying the number of knots dictates the granularity of the spline representation across the data range. While typically in two dimensions, this would entail four hyper-parameters, as each axis would have its own set, in this experiment, we will utilise the same basis for both axes for simplicity.

Finding the Coefficients

Once these hyper-parameters are judiciously determined based on the specific requirements for the task at hand, the next step involves the construction of orthonormal basis splines. Leveraging established methodologies outlined in Chapter 3, these basis splines forms a foundational framework. Subsequently, the constructed basis splines serve as the building blocks for representing each image within the dataset. Employing the prescribed methodology, the corresponding splines for every picture are derived, encapsulating essential structural and contextual information inherent to the image, into a set of 784 variables.

4.2.2 Principal Component Analysis

Principal Component Analysis (PCA) is a widely-used dimensionality reduction technique employed to transform high-dimensional data into a lower-dimensional representation while preserving the essential structure and variance of the original data. Given the multitude of variables obtained from the splines, identifying key features becomes paramount.

The fundamental concept behind PCA is to identify a set of orthogonal axes, known as principal components, along which the data exhibits the maximum variance. By projecting the data onto these principal components, PCA effectively captures the most significant sources of variation within the dataset, facilitating compact and informative representations.

Mathematically, given a dataset $\mathbf{X}$ with n samples and p features, PCA involves the following steps:

1. Standardise the dataset by centering the data around the mean and scaling it to unit variance:

$$\mathbf{X}_{\text{standardised}} = \frac{\mathbf{X} - \mu}{\sigma}$$

where μ is the mean vector and σ is the standard deviation vector.

2. **Covariance Matrix Computation:** Compute the covariance matrix $\mathbf{C}$ of the standardised data:

$$\mathbf{C} = \frac{1}{n-1}\mathbf{X}_{\text{standardised}}^{\top}\mathbf{X}_{\text{standardised}}$$

3. **Eigen Decomposition:** Perform eigen decomposition on the covariance matrix to obtain eigenvalues and eigenvectors:

$$\mathbf{C}\mathbf{v}_i = \lambda_i \mathbf{v}_i$$

where λ_i and $\mathbf{v}_i$ are the eigenvalues and eigenvectors, respectively.

4. **Principal Components Selection:** Select the top k eigenvectors corresponding to the k largest eigenvalues to form the transformation matrix $\mathbf{V}_k$.

The hyperparameter k in PCA denotes the number of principal components retained in the reduced-dimensional space. Selecting an appropriate value for k is crucial, as it directly influences the amount of variance preserved in the

reduced representation. A higher value of k retains more variance but may lead to overfitting and increased computational complexity, while a lower value of k sacrifices some variance for simplicity and computational efficiency.

In this experiment, PCA is applied to each of the ten classes in the dataset, resulting in a unique transformation matrix for each class. By following these steps, we ensure that the PCA transformation for each class captures the most relevant features, allowing for effective dimensionality reduction and improved classification performance.

4.2.3 Classification

To classify a new vector of variables, the vector undergoes transformations specific to each class. The steps are as follows:

1. **Remove Zero Variance Variables:** Zero variance variables, which were identified and removed during the training phase for each class, are also removed from the new vector.

2. **Scaling:** The remaining variables in the new vector are scaled using the mean and standard deviation values saved from the training data for each class. Let $\mathbf{x}$ be the new vector and μ_{class} and σ_{class} be the mean and standard deviation vectors for the class. The scaled vector $\mathbf{x}_{\text{scaled}}$ is computed as:

$$\mathbf{x}_{\text{scaled}} = \frac{\mathbf{x} - \mu_{\text{class}}}{\sigma_{\text{class}}}$$

3. **Projection onto Principal Components:** The scaled vector is then projected onto the subspace spanned by the top k principal components of the class. If $\mathbf{V}_k$ is the matrix of the first k eigenvectors (principal components) for the class, the projection $\mathbf{x}_{\text{projected}}$ is given by:

$$\mathbf{x}_{\text{projected}} = \mathbf{V}_k \mathbf{V}_k^\top \mathbf{x}_{\text{scaled}}$$

4. **Calculate Distance:** The distance between the original scaled vector $\mathbf{x}_{\text{scaled}}$ and its projection $\mathbf{x}_{\text{projected}}$ is calculated using the Euclidean distance:

$$d = \|\mathbf{x}_{\text{scaled}} - \mathbf{x}_{\text{projected}}\|_2$$

These steps are repeated for each class, resulting in a distance value for each class. The class with the smallest distance value is selected as the predicted class for the new vector.

In summary, the class whose projection has the shortest distance from the original scaled vector is chosen as the classification result.

4.3 Results

In this section, we present the findings of our experiments. We first discuss the selection of the order of smoothness and the number of knots for the spline approximations. Subsequently, we describe the process of choosing the optimal number of principal components for the Principal Component Analysis (PCA). The accuracy of our classification model, based on these parameters, is also evaluated and compared to previous approaches.

Choosing Order of Smoothness and Number of Knots

Figure 4.2: The figure juxtaposes original images, each measuring 34×34 pixels, with their corresponding projections onto splines, visualized at a resolution of 270×270 pixels. The basis splines are created with 31×31 knots and an order of smoothness equal to 2.

To determine the optimal order of smoothness and the number of knots, we ensure that they are dyadic, as this configuration provides the best efficiency for the `Splinet` package [Liu et al., 2023, page 46]. Ensuring dyadic form involves choosing an order of smoothness (k) and an integer (N) greater than 1, such that the number of knots along one axis is $2^N k - 1$. Additionally, the number of knots should not exceed the number of pixels along the axis (34), as exceeding this number tends to introduce extraneous details into the image, leading to overfitting.

As we plan to use Principal Component Analysis (PCA) later in the process (as mentioned in Section 4.2.2), it is crucial to choose hyperparameters that allow the approximated images to closely match the original images, while still adhering to the dyadic form.

After testing various combinations of k and N on the test data and calculating the mean absolute error, we found that using 31 knots and an order of smoothness of 2 provided the closest approximation to the original images. This configuration will be used in subsequent steps. Notably, an order of smoothness of 4 also performed comparably well with 31 knots.

Choosing k

Cross-validation was employed on the training data to evaluate the accuracy of the algorithm for different values of k, the number of principal components. The training data was divided into ten subsets. Each subset was sequentially set aside as test data while the remaining subsets were used for training. The algorithm's performance was evaluated on the test subset, and this process was repeated for all ten subsets. The mean ratio of correctly classified images was used to measure accuracy for each value of k.

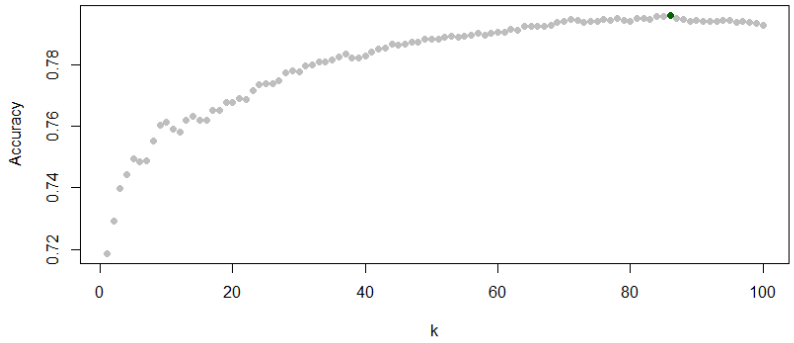

Figure 4.3: The figure illustrates the accuracy for different values of k between 1 and 100, where $k = 86$ is marked as green.

Starting with low values of k, we incrementally increased k until we observed a decline in accuracy. The optimal value, $k = 86$, provided the best performance, yielding an accuracy of 79.69% during cross-validation.

Final Result

Using the parameters attained above, the model was tested on the test data, achieving an accuracy of 79.23%. This can be compared to the results in [Basna et al., 2023a], where the images were treated as one-dimensional splines. There, the highest accuracy attained was 78.6%, indicating there is an improvement when using tensor products to attain two-dimensional orthonormal bases.

5 Discussion

In this thesis, we have theoretically explored and empirically tested a method for using tensor splines to handle two-dimensional functions. While the results are not extraordinary, they show promise, indicating that with further refinements, this method could become an efficient approach. Several areas have been identified for potential improvement.

The Issue of Hyperparameters

The selection of hyperparameters, particularly the number of knots and the order of smoothness, significantly impacts the accuracy of image evaluation. Adhering to the dyadic formula, as discussed in Section 4.2.1, limits the possible choices for the number of knots, potentially excluding the optimal combination of knots and smoothness order.

In this thesis, we assumed that the configuration producing approximations closest to the original images is the best choice, particularly in the context of PCA. However, this assumption may not hold true, especially since the comparison ignores areas between pixels, which can bias the results towards configurations with a higher number of knots. Here, the only upper limitation considered was the original number of pixels, whereas other constraints might also be relevant.

Additionally, it is reasonable to believe that the number of knots has a more substantial effect on accuracy than the order of smoothness. Therefore, maintaining the dyadic relationship between these parameters might not yield the most accurate image representations.

Future research could focus on developing a more sophisticated mathematical framework for selecting hyperparameters for two-dimensional splines. This could involve optimising the trade-off between the number of knots and the order of smoothness to improve the overall performance and accuracy of the spline approximations.

Choice of Model

The choice of model is crucial in high-dimensional data analysis. In our example, we used Principal Component Analysis to identify subspaces that best described each class and then classified a data point based on the closest subspace. However, exploring other models for classification might yield better results. Investigating alternative approaches could enhance the performance and robustness of the classification system.

The Possibilities in Two-Dimensional Splines

An important issue in the given dataset is that some classes are very similar, which can confuse the model. For instance, shirts and coats have a higher tendency to be misclassified (see figure 5.1). Coats and shirts are often classified as pullovers, while shirts and t-shirt are frequently mistaken for each other. Knowing these issues, one potential solution is to group these similar classes and then focus on specific parts of the image to differentiate within these groups.

Working in two dimensions allows for a more intuitive approach to focus on specific parts of the function compared to treating it as a one-dimensional function. This could improve the model's ability to accurately classify similar classes by emphasising distinguishing features.

Overall, while the current results indicate room for improvement, the approach shows potential. Further studies and refinements could lead to significant advancements in the application of tensor splines for two-dimensional functions.

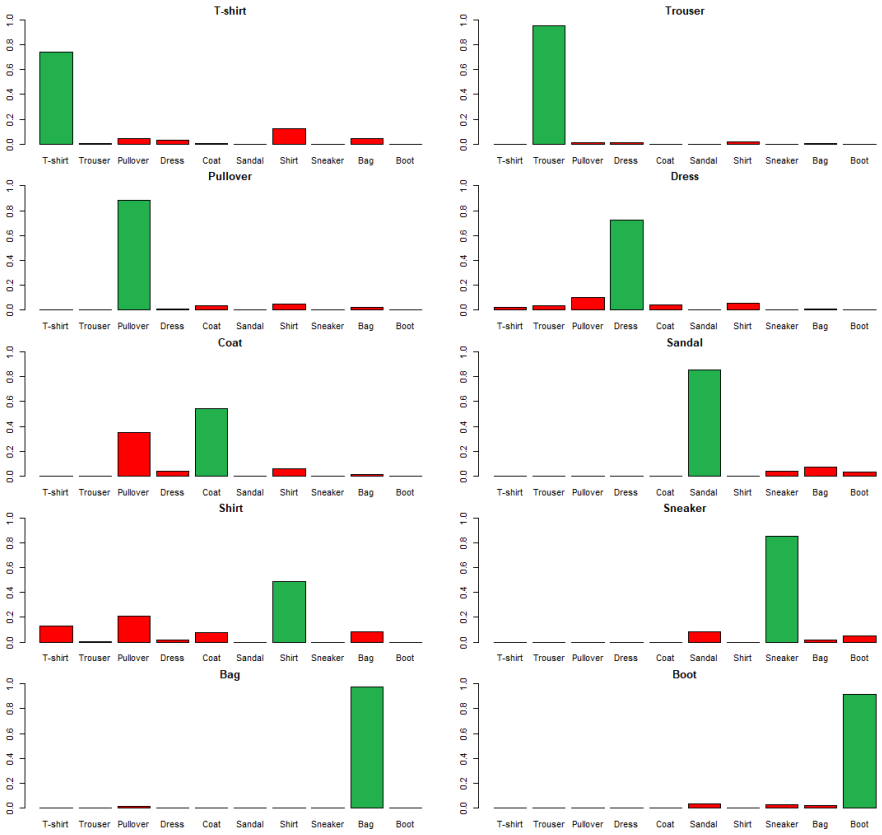

Figure 5.1: The figure illustrates the classification results for each class using the model from the experiment. Each subplot corresponds to a true class, where bars indicate the classification outcomes. Green bars denote correct classifications, while red bars indicate incorrect classifications.

6 Conclusion

The method of transforming a two-dimensional function into a set of coefficients using tensor products of splines proves to be a robust approach for handling functions within high-dimensional data frameworks. Exploring the properties and applications of such transformations holds promise for effectively analysing two-dimensional functional data. Moreover, laying this groundwork can pave the way for extending the methodology to functions of higher dimensions, potentially opening new avenues in data analysis and computational modelling. By leveraging the versatility and efficiency of spline-based transformations, researchers can advance their capabilities in processing and interpreting complex datasets across various domains.

Bibliography

[Basna et al., 2023a] Basna, R., Nassar, H., and Podg'orski, K. (2023a). Splinets – orthogonal splines and fda for the classification problem.

[Basna et al., 2021] Basna, R., Nassar, H., and Podgórski, K. (2021). Machine learning assisted orthonormal basis selection for functional data analysis.

[Basna et al., 2023b] Basna, R., Nassar, H., and Podgórski, K. (2023b). Empirically driven spline bases for functional principal component analysis in 2d.

[Basna et al., 2024] Basna, R., Nassar, H., and Podgórski, K. (2024). Spline based methods for functional data on multivariate domains.

[Debnath and Mikusinski, 1999] Debnath, L. and Mikusinski, P. (1999). *Introduction to Hilbert Spaces with Applications*. Academic Press Inc, 2 edition.

[Douglas Carroll et al., 2003] Douglas Carroll, J., E. Green, P., and Latting, J. (2003). *Analyzing Multivariate Data*. Thomson Learning.

[Gába et al., 2021] Gába, A., Talská, R., Machalocá, J., and Hron, K. (2021). Compositional splines for representation of density functions. *Computational Statistics*, 36:1031–1064.

[Johnson and Wichern, 2014] Johnson, R. and Wichern, D. (2014). *Applied Multivariate Statistical Analysis*. Pearson, 6 edition. New International Edition.

[Liu et al., 2023] Liu, X., Nassar, H., and Podgorski, K. (2023). Functional data analysis using splines and orthogonal spline bases.

A R-code: Functions for Two-Dimensional Splines

Here is the R-functions used for handling two-dimensional splines in this thesis. To use these one must have the R-package *Splinets* installed.

```
#####################################################################
#Function: splinet_2dim(xknots, yknots, smorder = 3)
#   The function finds and returns a list of two sets of
#   orthonormal basis spline objects.
#Inputs:
#   xknots: n+2 vector, the knots
#       (presented in the increasing order)
#       used for the x-axis
#   yknots: n+2 vector, the knots
#       (presented in the increasing order)
#       used for the y-axis
#   smorder: integer, the order of the splines,
#       the default is smorder=3.
#Output:
#   A list two orthonormal basis spline  objects;
#   OB_1 for the x-axis, and OB_2 for the y-axis
#####################################################################
splinet_2dim <- function(xknots, yknots, smorder = 3){
  OB_1 <- splinet(xknots,smorder,norm = T)
  OB_2 <- splinet(yknots,smorder,norm = T)
  return(list("OB_1"=OB_1,"OB_2"=OB_2))
}
```

```
#####################################################################
#Function: project_2dim(f, splines, x.set, y.set)
#   The function projects a function f
#   to find and return coefficients
#   for two-dimensional basis splines.
#Inputs:
#   f: A matrix of estimated values for the
#       double integral of the function
#       splines: A list of splines as
#       attained from the splinet_2dim
#   x.set: A vector of the values of x
#       on which the integral values of f
#       was observed.
#   y.set: A vector of the values of y
#       on which the integral values of f
#       was observed.
#Output:
#   The list of splines is returned with
#   an additional element "Coeff",
#   containing the coefficients for the
#   projection of f onto the basis splines.
#####################################################################
project_2dim <- function(f, splines, x.set, y.set){
  OB1x <- evspline(splines$OB_1$os,
        x=x.set)[,2:(length(splines$OB_1$os@der)+1)]
  OB2y <- evspline(splines$OB_2$os,
        x=y.set)[,2:(length(splines$OB_2$os@der)+1)]
  splines$Coeff <- t(OB1x)%*%f%*%OB2y
  return(splines)
}
```

```
#####################################################################
#Function: evsplines_2dim(object, x=NULL, y=NULL, N=250)
#   The function takes a list of splines
#   with projected coefficients as
#   returned from project_2dim, and
#   evaluates them for a set x and y.
#Inputs:
#   object:
#       A list of splines with
#       coefficients as attained from
#       project_2dim.
#   x: vector, the arguments at which
#       the splines are evaluated along
#       the x-axis; If x is NULL, then
#       the splines are evaluated over
#       regular grids per each interval
#       of the support. The default
#       value is x=NULL.
#   y: vector, the arguments at which
#       the splines are evaluated along
#       the y-axis; If y is NULL, then
#       the splines are evaluated over
#       regular grids per each interval
#       of the support. The default
#       value is y=NULL.
#   N: integer, the number of points per
#      an interval between two
#      consequitive knots at which the
#      splines are evaluated. The
#      default value is N = 250;
#Output:
#   Returns a matrix containing the
#   values of the evaluated Spline at
#   each coordinate (x,y) attained from
#   x, y or N.
#####################################################################
evsplines_2dim <- function(object, x=NULL, y=NULL, N=250){
  OB1x <- evspline(object$OB_1$os,x=x,
        N=N)[,2:(nrow(object$Coeff)+1)]
  OB2y <- evspline(object$OB_2$os,x=y,
        N=N)[,2:(ncol(object$Coeff)+1)]
  f_hat <- OB1x%*%object$Coeff%*%t(OB2y)
  return(f_hat)
}
```