Statistical Analysis of Functional Data

Emily Helfer[a], Bennett Smith[b], Rana Haber[c], and Adrian M. Peter[d]

[a]Department of Statistics, Carnegie Mellon University, Pittsburgh, Pennsylvania; ehelfer@andrew.cmu.edu
[b]Department of Mathematics and Physics, Illinois College, Jacksonville, Illinois; smith.bennett@mail.ic.edu
[c]Department of Mathematical Science, Florida Institute of Technology, Melbourne, Florida; rhaber2010@my.fit.edu
[d]Department of Engineering Systems, Florida Institute of Technology, Melbourne, Florida; apeter@fit.edu

ABSTRACT

In this paper, we explore several different statistical techniques for functional data analysis. Initially, we explored non-parametric density estimation for (infinite dimensional) functional data. The theoretical development differs slightly from standard multivariate statistics due to the technical issues that arise when working in infinite dimensional function spaces; for example, care must be taken during operations such as integration. We discuss using wavelets and Hermite polynomials as bases for orthogonal series density estimation of finite dimensional data and their extensions to the function space setting. However, due to the exhaustive computations needed to apply orthogonal series density estimation to functional data, we decided to explore using kernel density estimation instead. The downside to this is that the functional data must be spatially dependent to use kernel density estimation. Finally, we study classification of functional data using distribution fields. We focus on using several function-specific features of the data and develop a novel metric learning framework that optimally combines the features and classifies the data. The practicality of this classification technique is demonstrated on several functional datasets.

Keywords: Classification, Density Estimation, Distribution Fields, Functional Data, Functional Data Analysis

1. INTRODUCTION

Functional data is important because it is present in many forms. Functional data can be curves, surfaces, or images. However, all functional data must have the distinctive aspect of being continuously defined and having an underlying continuum. In contrast to finite dimensional data, functional data is the entire curve or surface, not just a point in space. This assumes that data is measured over a continuum, or that it is smoothly varying. Since functional data is so applicable, it is prevalent in many fields of study. Simple examples of functional data are curves representing temperature changes, stock market fluctuations, or people's gaits as they walk. Some examples of functional data that are more complex are data of a persons veins shown from a CT scan or even the image of the CT scan itself. Due to these properties of functional data, Functional Data Analysis (FDA) is becoming a growing field in machine learning.

One area of FDA that we explored is density estimation. Density estimation is an important area of analysis because it allows us to provide a statistical description of the data and how they are distributed. Density estimation is needed to do further statistical analysis on data, such as taking expected values or creating confidence intervals. We began by implementing one dimensional density estimation using Orthogonal Series Density Estimation (OSDE). This density estimation used wavelet scaling functions as the basis. Then, we implemented OSDE again, but using Hermite polynomials instead of wavelets to form the basis. Using both these techniques, we were able to achieve high estimation accuracy. We then explored extensions of the OSDE technique to the functional data setting. This requires the development of orthogonal bases for infinite dimensional function spaces and the use of an appropriately defined measure to carry out integration (the standard Lebesgue measure of finite dimensional spaces is not defined in this infinite setting). However, due to the expensive computations of implementing OSDE to functional data, we decided to use Kernel Density Estimation (KDE) to create the

AMALTHEA REU Technical Report No. 2015–5; available at www.amalthea-reu.org. Please send your correspondence to Georgios C. Anagnostopoulos.

density estimation for functional data. We implemented KDE for both finite dimensional data and spatially dependent functional data.

The ultimate objective of exploring density estimation was to use it in the development of classification algorithms that can better categorize functional data from multiple different classes. Classification of functional data allows us to do things such as determining whether or not people are injured based on curves representing their gaits or detecting whether a person suffers from a heart anomaly based on data from an EKG machine. We explored a novel supervised classification method for functional data which uses Distribution Fields (DFs), an idea previously used for image tracking.[23] The DFs take advantage of the curves being continuous because they are created based on features of the curves, such as amplitudes of the data or its derivatives. The DFs based on the features are combined using optimal weights and these combined DFs are used to classify the data by the means of a learned metric. The optimal weights and metric are found using a novel optimization method based on concave-convex optimization.[33] Finally, we have implemented our classification method on several functional datasets. Not only is our classification method competitive with state of the art classification methods for functional data, but it also has the benefit that it is able to use the functional nature of the data.

The layout of this paper is as follows. Section 2 discusses relevant literature about functional data, FDA, and DFs. Section 3 explains what functional data is, the goals of FDA, and some basic terms about FDA. Section 4 explores Non-Parametric Density Estimation (NPDE) through OSDE with both wavelets and Hermite polynomials. Section 5 and Section 6 describe the process of applying NPDE to one-dimensional data and spatially dependent functional data through the use of kernels. Section 7 gives background information on both DFs and the process to use them for functional data classification. Section 8 explains the process for combining DFs based on different features of functional data in order to optimize our classification method. Section 9 details the classification experiments we performed on several functional datasets; focusing mainly on the setup and results/comparisons. Section 10 discusses the usefulness of our classification method and further improvements that can be made. Appendix A, Appendix B and Appendix C contain background information about measure theory (applicable to Functional Density Estimation (FDE)), the L^2 space and integration measures (both applicable to functional data in general).

2. LITERATURE REVIEW

There have been many explorations in the field of functional data and FDA. Ferraty and Vieu[6] describe a functional datum to be an observation of a random variable that takes values in an infinite dimensional space. Horváth and Kokoszka,[11] along with many others, present functional data as an observation dependent on some bounded time interval, such as curves. Lazar[16] makes sure to point out that functional data does not have to be a function of time, and furthermore, does no have to be one dimensional. Thus, we can think of many things as functional data besides curves, such as images. Zhang and Müller[34] give many different examples of why functional data analysis is of importance. One such example is density and hazard functions of age-at-death, pertinent in biomedical studies, which focuses on analyzing the age-at-death data in order to create trajectories of mortality. This would allow things such as life expectancy predictions to be made.

As noted by Cuevas,[2] FDA is a fairly recent topic, only dating back to the 1990's and only gaining popularity beginning in 1997. In particular, it was Ramsay and Silverman who jump started the advancements in FDA. In their book,[20] there are a multitude of FDA techniques such as functional principal component analysis and functional discriminant analysis. In addition to Ramsay and Silverman, Ferraty and Vieu[6] are two others who pushed forward the field of FDA. They focus mainly on two specific areas of FDA in their book: prediction and classification. For classification, they discuss using the k-Nearest Neighbors (kNN) classification algorithm, which we have implemented to gauge the usefulness of our distribution field classification algorithm. Similar to the classification aspect, Cuesta-Albertos and Fraiman[1] developed a way to cluster functional data based on the concept of impartial trimming. In a more recent book,[5] Ferraty discusses many other different functional data analysis techniques, such as regression, dimensionality reduction, density estimation, and clustering. However, for the density estimation and clustering, the theory may only be applied to spatial functional data.

The most recent advances in classification of functional data include several different approaches that base the classifiers on different aspects of the data. Goia and Bongiorno[8] proposed an unsupervised classification algorithm

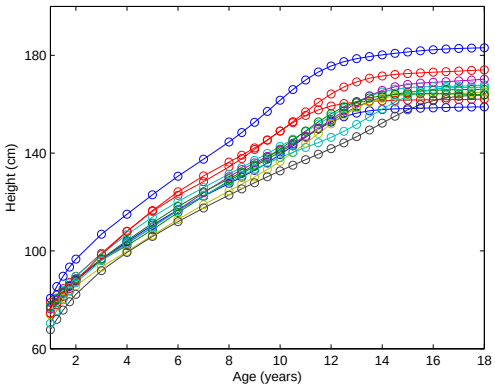


Figure 3.1: The height of 10 children as functions of time. This would be 10 functional data observations, one for every curve representing a child's height over time.[20]

for functional data that extends a density based multivariate cluster approach using small-ball probabilities as the starting points for clusters. However, they only use their algorithm for one simulated dataset, while the classifier in this paper is used on several real datasets. Fuchs et al.[7] proposed the use of semi-metrics based on features extracted from curves combined with kNN to classify functional data. Their paper only uses kNN as a classifier, while this paper explores the use of a different classifiers besides kNN. Hubert et al.[12] propose a classification technique called classification in distance space, which aims to provide a fully non-parametric, robust, supervised classifier that works with possibly skewed data. The classifier is based on depth and distance measures of the curves. Varughese et al.[28] propose a classifier that uses wavelet coefficients to characterize and classify the functional data based on light curve measurements. Although it is stated in their paper that the classifier could be used on curves other than light curves, they only tested it on one astronomical dataset, while the classifier proposed in this paper was tested on several datasets and could be used on any set of functional data. Górecki et al.[9] proposed classifying functional data based on adapted multivariate regression models and techniques. Their classifier only focuses on binary classification, while the classifier in this paper is able to perform multiclass classification as well.

Since their debut, DFs have primarily been used for image processing, such as tracking portions of a photo or an item in a video. Sevilla-Lara and Learned-Miller[23] proposed the idea of using DFs to track images. Ning et al.[19] used DFs to create a weighted-geometric-mean multiple instance learning classifier, again for image tracking. Both of these papers only used DFs on images and did not implement them on any other functional data as will be discussed in this paper.

3. FUNCTIONAL DATA

3.1 Basics

As mentioned in Section 2, functional data is an observation of a random variable that takes values in an infinite dimensional space.[6] Functional data can arise from independent replications, a single long record, or input/output pairs. The simplest example of functional data is time series data. One such example would be curves representing the height of children over time, as seen in Figure 3.1. In an example such as this, it is important to note that the functional data are the curves themselves, not the individual points that make up the curves. Although functional data have an underlying smooth continuous function, it is discretely recorded observations over some continuum, such as time or position. Techniques such as interpolation can help define values at unknown positions. If we did not consider the underlying functional data to be smooth, there would be no advantage to treating the discrete observations as functional as opposed to simply multivariate.[20]

In the example of the height of children over time, we were dealing with one dimensional functional data. However, functional data can also be multidimensional. This occurs when we have functional observations that are distributed over multidimensional domains. This would include images, which have intensity and color composition as a function of spatial location.[20]

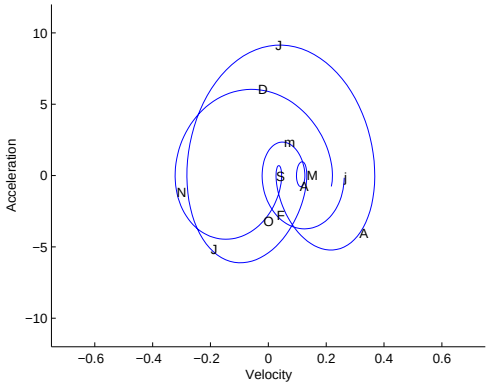


Figure 3.2: A phase-plane plot of velocity and acceleration for the non-durable goods.[20]

3.2 Functional Data Analysis

The goals of Functional Data Analysis (FDA) is to represent the data in ways that aid further analysis. Further analysis could be anything from being able to take derivatives or comparing different datasets against each other. Another goal of FDA is to display the data in such a way as to highlight various characteristics. The features that are selected and how they are represented can provide new insights into the physical significance of the data. For example, plotting derivatives of curves against each other is a way to display data to bring attention to specific characteristics of the data. A third goal of FDA is to study important patterns or variations. Finding patterns can help make predictions and develop hypotheses about how the data would look outside of the observed scope. A fourth goal is to explain variation in an outcome or dependent variable by using input or independent variable information. This is essentially experimentation, since different values of the independent variable are used as input and the output changes are observed. A final goal of FDA is to compare two or more sets of data with respect to certain types of variation. The two sets of data can contain different sets of replicates, such as different children from whom the height and weight data was taken from in Figure 3.1, of the same functions, or different functions for a common set of replicates.

The first step of FDA is to take the discrete data points and create a smooth underlying function. If the discrete data is completely error-less, then the process of creating the underlying function is called interpolation. If there is error or noise that affects the data, the process is called smoothing. One way of providing smoothing is to identify some of the known constraints of the function, such as knowing that a child's height cannot be negative, and imposing those constraints on the function. Another problem that is prevalent when using FDA is that functional data is lent from independent replications so that the first "step" can be located arbitrarily in time.To make the data more comprehendable, time shifts must be done to line up the data.

Displaying results from FDA can be difficult since there are many different ways to represent data and some forms may give a better representation than others. For example, the standard plot of a function against time may not be very informative. Instead, the plots of derivatives versus time could prove more interesting or the relationships between derivatives, phase-plane plots, can provide information not easily seen in the original function. One example of a phase-plane plot is given in Figure 3.2

3.3 Definitions

FDA uses many key concepts. Some of the concepts are listed below where x_i is a curve, N is the total number of curves, and t is where the function is being evaluated.

- Mean - The average value of a function
  - $\bar{x}(t) = N^{-1} \sum x_i(t)$
- Variance - Measures how far a set of numbers is spread out

- $var_X(t) = (N-1)^{-1} \sum x_i(t) - \bar{x}(t)]^2$.

- Covariance - How two random variables change together
  - $cov_X(t_1, t_2) = (N-1)^{-1} \sum \{x_i(t_1) - \bar{x}(t_1)\}\{x_i(t_2) - \bar{x}(t_2)\}$.
- Correlation - The statistical relationship of dependence
  - $corr_X(t_1,t_2) = \frac{cov_X(t_1,t_2)}{\sqrt{var_X(t_1)var_X(t_2)}}$.
- Cross-covariance - The covariance between (X, Y)
  - $cov_{X,Y}(t_1, t_2) = (N-1)^{-1} \sum \{x_i(t_1) - \bar{x}(t_1)\}\{y_i(t_2) - \bar{y}(t_2)\}$.
- Cross-correlation - The correlation between (X, Y)
  - $corr_{X,Y}(t_1, t_2) = \frac{cov_{X,Y}(t_1,t_2)}{\sqrt{var_X(t_1)var_Y(t_2)}}$.
- Dimensionality - The sum across functional features or the number of pieces of information that are required to define each feature or event.
- Practical dimensionality - The total amount of information required to define it to some required level.
- Resolution - The location and number of observations taken.
- Smooth - The close data values are unlikely to be too different and the underlying function possesses one or more derivatives.

4. ORTHOGONAL SERIES DENSITY ESTIMATION

4.1 Wavelets

We begin our non-parametric density estimation by working with Orthogonal Series Density Estimation (OSDE) for one dimensional data. Non-parametric density estimation does not assume an underlying distribution for the data. OSDE uses an orthogonal basis for the estimation. The bases being used in this estimation are different compact wavelet basis functions. Using an expansion of the chosen orthogonal basis with scalars that are estimated from the orthogonal basis and the sample data, a density function can be created in the L^2 space. The density function is given by Eq. (4.1),[21] where $f \in L^2(\mathbb{R})$ and $\phi(x)$ is the basis set of functions. In this case, $\phi(x)$ is the basis for the specific wavelet function chosen in relation to the sample data and $\phi_j(x)$ is an individual component of the basis.

$$f(x) = \sum_{j=1}^{\infty} \alpha_j \phi_j(x) \tag{4.1}$$

Since ϕ is an orthogonal basis, each coefficient of the density function, α_j, corresponds to a specific component of the basis function, ϕ_j, and each coefficient can be calculated by finding the inner product of the density function, $f(x)$, and the corresponding component, $\phi_j(x)$, shown in Eq. (4.2) .

$$\alpha_j = < f, \phi_j > \tag{4.2}$$

The definition of the inner product on functions is the integral of both functions, so when the inner product from Eq. (4.2) is expanded out to the integral between the functions f and ϕ_j , Eq. (4.3) is formed. By the definition of the expected value of a variable, the coefficients of the density function are equal to the expected value of the corresponding basis function given the sample data, shown in Eq. (4.4).[27]

$$\alpha_j = \int f(x)\phi_j(x)dx \tag{4.3}$$

$$\alpha_j = E[\phi_j(x)] \tag{4.4}$$

The expected value of a variable is approximately equal to the mean of the variable, which can be applied to the corresponding basis function, ϕ_j, to calculate the estimated coefficient, $\hat{\alpha}_j$, as

$$\hat{\alpha}_j = \frac{1}{N}\sum_{i=1}^{N}\phi_j(x_i). \tag{4.5}$$

The estimated density function based on the chosen basis functions and the sample data, Eq. (4.6), is created when the estimated coefficients are substituted back into Eq. (4.1)[27]

$$\hat{f}(x) = \sum_j \hat{\alpha}_j\phi_j(x). \tag{4.6}$$

When wavelet functions are used as the orthogonal basis for the density estimation, each wavelet has its own base function, $\phi(x)$. An example of a base function, is the base Haar wavelet function,[21] which is defined as

$$\phi(x) = \begin{cases} 1, & 0 \le x < 1 \\ 0, & otherwise \end{cases}. \tag{4.7}$$

Transformations such as translations and dilations need to be applied to the base wavelet function to create a new basis for the estimation and to achieve a density function that correctly models data on a domain different than the original domain of the wavelet function. The translations of the base wavelet function can be found by multiplying a scaling coefficient, 2^j where j is the resolution of the density estimate, to all of the sample data values and subtracting every translation value, k, needed for the entire domain to be covered by the function. The values of k are determined by

$$\lceil a2^j \rceil - 2N + 1 \le k \le \lfloor b2^j \rfloor \tag{4.8}$$

where a and b are the upper and lower bounds, respectively, of the new sample domain to be estimated over and j is still the resolution of the density estimate.[27] This equation is based on the support of the chosen wavelet type. In the Haar wavelet example, since it is the same as Daubechies 1 wavelet type, the range of k would be from $\lceil a2^j \rceil - 1 \le k \le \lfloor b2^j \rfloor$.

Finally, all of the values for the new basis function are found by evaluating the translated base function, $\phi(2^j x - k)$, at each sample data point, and multiplying the resulting value by a normalizing factor, $2^{j/2}$, to ensure that the final estimated density function integrates to 1.

$$\phi_{j,k}(x) = 2^{j/2}\phi(2^j x - k) \tag{4.9}$$

shows the transformed basis function, $\phi_{j,k}$, where j is the resolution of the density estimate and k is the integer value of the translation.[21] Going back to the Haar wavelet example, once every sample point is transformed by multiplying by the scaling coefficient and subtracting every translation value, every new calculated value is evaluated in the base equation, Eq. (4.7). These values are then multiplied by the normalizing factor to get the new basis which consists of only the normalizing factor and zeros, which is used in Eq. (4.5) to calculate the estimated coefficients. The final estimated density function,[27]

$$\hat{f}(x) = \sum_k \hat{\alpha}_{j,k}\phi_{j,k}(x), \tag{4.10}$$

contains the estimated coefficients calculated from Eq. (4.5) and the values of the new basis functions calculated in Eq. (4.9).

4.2 Hermite Polynomials

4.2.1 Basics

There are two types of Hermite polynomials, Probabilists' Hermite polynomials and Physicists' Hermite polynomials. In this paper we will use Probabilists' Hermite polynomials, which we will refer to as just Hermite polynomials, for which the unnormalized equation is defined as

$$P_n(x) = (-1)^n e^{x^2/2}\frac{d^n}{dx^n}\left(e^{-x^2/2}\right). \tag{4.11}$$

In $\mathrm{L}^2(\mathbb{R}, \mu)$, the space of all finite integrable functions as defined in Eq. (B.13), Hermite polynomials are orthogonal,

$$\langle P_n, P_m \rangle_\mu = E\left[P_n(\xi) P_m(\xi)\right] = n! \delta_{n,m}. \tag{4.12}$$

where $\delta_{n,m}$ is the Kronecker delta,[17] which allows for them to be used for OSDE. For an orthonormal Hermite polynomials, $H_n(x)$, $P_n(x)$ in Eq. (4.11) are divided by the normalizing constant $\sqrt{n!}$,

$$H_n(x) = \frac{P_n(x)}{\sqrt{n!}} = (n!)^{-1/2} (-1)^n e^{x^2/2} \frac{d^n}{dx^n}\left(e^{-x^2/2}\right). \tag{4.13}$$

Hermite polynomials are only defined for $n = 0, 1, 2...$, however $P_{-1}(x) = H_{-1}(x) = 0$ and $P_0(x) = H_0(x) = 1$. The Hermite polynomials also have a generating function,[17] defined as

$$\psi(x, z) = e^{-x^2/2 + xz}, \tag{4.14}$$

which is also used in OSDE.

4.2.2 Recursive Relationship

We are able to transform equations to substitute back into the differentiation to achieve a recursive relationship between degrees of Hermite polynomials, $P_n(x)$. We begin by differentiating the unnormalized Hermite polynomial in Eq. (4.11), to get an equation that relates different degree Hermite polynomials,

$$P_n'(x) = x P_n(x) - P_{n+1}(x). \tag{4.15}$$

The Hermite generating function, Eq. (4.14) is then expanded into a Taylor series and simplified to

$$\psi(x, z) = e^{x^2/2} e^{-(z-x)^2/2} = e^{x^2/2} \sum_{n=0}^{\infty} \frac{(-1)^n}{n!} \frac{d^n}{dx^n}\left(e^{-x^2/2}\right) z^n = \sum_{n=0}^{\infty} \frac{P_n(x)}{n!} z^n. \tag{4.16}$$

Once the partial derivative with respect to x is calculated

$$\frac{\partial}{\partial x}\psi(x, z) = z\psi(x, z) = \sum_{n=0}^{\infty} \frac{P_n(x)}{n!} z^{n+1} = \sum_{n=0}^{\infty} \frac{P_n'(x)}{n!} z^n, \tag{4.17}$$

the summation index can be shifted to get another equation that relates to the degrees of polynomials,

$$P_n'(x) = n P_{n-1}(x). \tag{4.18}$$

The equation calculated from the partial derivative of the expanded generating function, Eq. (4.18) can be substituted into the derivative for the unnormalized Hermite polynomial, Eq. (4.15) and rearranged to get the recursive relationship for Hermite polynomials,

$$P_n(x) = x P_{n-1}(x) - n P_{n-2}(x). \tag{4.19}$$

Similarly, for normalized Hermite polynomials, $H_n(x)$, the recursive relationship is described as

$$H_n(x) = \frac{1}{\sqrt{n}}\left(x H_{n-1}(x) - \sqrt{n-1} H_{n-1}(x)\right). \tag{4.20}$$

These recursions are only valid for $n = 0, 1, 2, \ldots$, just as Hermite polynomials are.[17]

4.2.3 Orthonormal Basis

The normalized Hermite polynomials, $\{H_n(x) : n = 0, 1, 2, \ldots\}$ form a complete orthonormal basis in $\mathrm{L}^2(\mathbb{R}, \mu)$.[17] Therefore, for any $f(x) \in \mathrm{L}^2(\mathbb{R}, \mu)$, there exists a Fourier-Hermite expansion using the normalized Hermite polynomials, $H_n(x)$,

$$f(x) = \sum_{n=0}^{\infty} f_n H_n(x), \tag{4.21}$$

where the inner product of the expansion is defined as

$$f_n = \langle f(x), H_n(x) \rangle = \int f(x) H_n(x) \mu(dx). \tag{4.22}$$

If we consider $f \in \mathrm{L}^2(\mathbb{R}, \mu)$ as a function of a unit Gaussian random variable ξ, using the definition of the inner product of ξ, Eq. (B.15), we can rewrite the Fourier-Hermite expansion

$$f(\xi) = \sum_{n=0}^{\infty} f_n H_n(\xi) \tag{4.23}$$

and the inner product as

$$f_n = \langle f(\xi), H_n(\xi) \rangle = E[f(\xi) H_n(\xi)]. \tag{4.24}$$

. The density distribution of ξ, $f_N(\xi)$, can be estimated by truncating the Fourier-Hermite expansion at N, shown in Eq. (4.25). If the density distribution of ξ is smooth, f_N will converge fast and can be estimated accurately with a smaller N.[17]

$$f_N(\xi) = \sum_{n=0}^{N} f_n H_n(\xi) \tag{4.25}$$

Let the Hermite functions, $\psi_n(x)$, be represented by

$$\psi_n(x) = \rho^{1/2}(x) H_n(x), \tag{4.26}$$

where $\rho(x)$ is the Gaussian distribution defined in Eq. (B.12) and $H_n(x)$ are the normalized Hermite polynomials. These Hermite functions form an orthonormal basis for $L^2(\mathbb{R})$, and thus any $g(x) \in \mathrm{L}^2(\mathbb{R})$ can be written as a linear combination of the orthonormal basis, $\psi_n(x)$,

$$g(x) = \sum_{n} g_n \psi_n(x) \tag{4.27}$$

and the inner product, g_n, can be written as

$$g_n = \langle g(x), \psi(x) \rangle = E[g(x), \psi(x)]. \tag{4.28}$$

Just as in Eq. (4.25), the Fourier-Hermite expansion with the Gaussian distribution, $g(x)$, can be truncated to create a density estimation.

5. KERNEL DENSITY ESTIMATION

Another form of non-parametric density estimation is Kernel Density Estimation (KDE), which relies on a specified kernel function. Its advantages over histograms are that the kernel density estimator is continuous and that inherits properties of the kernel function, such as differentiability. The kernel estimator is defined as

$$\hat{f}(x) = \frac{1}{nh} \sum_{i=1}^{N} K\Big(\frac{x - x_i}{h}\Big), \tag{5.1}$$

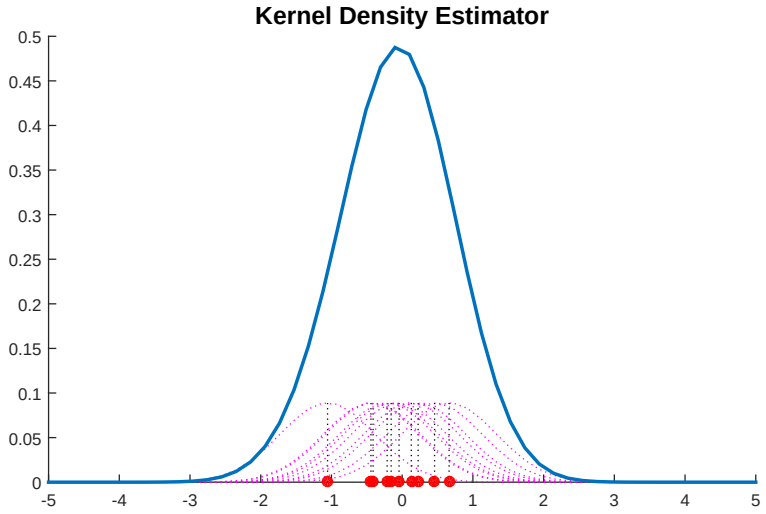


Figure 5.1: Kernel Density Estimator

where h is the bandwidth, also called the smoothing parameter or window width, and K is the kernel function. The bandwidth is what controls the width of the kernel functions, while the kernel function controls the shape. The kernel function must satisfy the property

$$\int_{-\infty}^{\infty} K(x)dx = 1 \tag{5.2}$$

so that the density estimator integrates to 1. The density estimator can also be rewritten using $K_h(t) = K(t/h)/h$ as

$$\hat{f}(x) = \frac{1}{n}\sum_{i=1}^{N} K_h(x - x_i). \tag{5.3}$$

Conceptually, KDE can be thought of having a kernel function sitting on top of each data point, and the density estimate can be calculated by taking the sum of every individual kernel function. An example of a kernel density estimator with the underlying kernels displayed is shown in Figure 5.1.

The most common kernel functions are symmetric probability densities, such as the standard Gaussian distribution, however other kernels such as the triangle, Epanechnikov, biweight, and triweight are also often used. The equations for these kernels can be seen in Table 5.1.

Table 5.1: Kernel functions

Kernel Name	Equation	Support
Triangle	$K(t) = (1 - \|t\|)$	$-1 \leq t \leq 1$
Epanechnikov	$K(t) = \frac{3}{4}(1 - t^2)$	$-1 \leq t \leq 1$
Biweight	$K(t) = \frac{15}{16}(1 - t^2)^2$	$-1 \leq t \leq 1$
Triweight	$K(t) = \frac{35}{32}(1 - t^2)^3$	$-1 \leq t \leq 1$
Normal	$K(t) = \frac{1}{\sqrt{2\pi}} exp\{-\frac{t^2}{2}\}$	$-\infty \leq t \leq \infty$

The optimal bandwidth for the KDE is chosen depending on the kernel function. For example, the optimal bandwidth for a Gaussian kernel is $h \approx 1.06\sigma N^{-1/5}$. The optimal bandwidth can be calculated by cross validation, or any bandwidth can be chosen by the user.

The procedure for calculating the KDE is to first choose a kernel function, K, the bandwidth, h, and the domain over which to evaluate, $\hat{f}(x)$. Next, for each observation, evaluate the kernel at all x in the domain as in

$$K_i = K\Big(\frac{x - x_i}{h}\Big); i = 1...N \tag{5.4}$$

to get one kernel for each data point, x_i, to make a set of N curves. Finally, weight each curve by $1/h$ and then take the average of the weighted curves for each x.

6. KERNEL SPATIAL DENSITY ESTIMATION OF FUNCTIONAL DATA

6.1 Basics

Finite dimensional KDE takes data from a finite dimension space, such as $\mathbb{R}^N$, and estimates the density function. On the other hand, infinite dimensional KDE takes data from an infinite dimension space, such as $C[0,1]$, and estimates the density functional. In this section, we will focus on spatially related functional data from $C[0,1]$. Since the data is functional and lives in an infinite dimension space, a reference measure must be set since the usual Lebesgue measure no longer exists. In theory, specifying the reference measure, μ, is not necessary because the behavior of the estimator is the same for any σ-finite measure such that $0 < \mu(A) < \infty$, for any open ball $A \in \mathcal{E}$.[3] We can think of this in a one-dimensional setting as saying that the measure is finite for any open interval in the space.

The functional data used for the kernel density estimation will come from a measurable stationary spatial process $\left(X_i, \mathbf{i} \in (\mathbb{N}^*)^N\right)$, $N \geq 1$, defined on a probability space $(\Omega, \mathcal{A}, \mathbf{P})$ such that the $X_\mathbf{i}$'s have the same distribution as variable X with values in an infinite dimensional separable normed space $(\mathcal{E}, \|\cdot\|)$. The variable X is assumed to have some unknown density f with respect to some given measure, μ, as discussed previously. The functional data, $X_\mathbf{i}$'s, come from an observed region $\mathcal{O}_n$ which can often be thought of as the rectangular region,

$$\mathcal{I}_\mathbf{n} = \left\{\mathbf{i} = (i_1, \ldots, i_N) \in (\mathbb{N}^*)^N, 1 \leq i_k \leq n_k, k = 1, \ldots, N\right\}, \tag{6.1}$$

where $\mathbf{n} = (n_1, \ldots, n_N) \in (\mathbb{N}^*)^N$. This rectangular region can be thought of as a lattice in $\mathbb{R}^N$.

The sample size can then be written as $\hat{\mathbf{n}} = n_1 \times \ldots \times n_N$. This leads to the kernel spatial density estimator for the functional data

$$f_\mathbf{n}(x) = \frac{1}{\hat{\mathbf{n}} a_\mathbf{n}^x} \sum_{\mathbf{i} \in \mathcal{I}_\mathbf{n}} K_\mathbf{n}(\|X_\mathbf{i} - x\|), \; x \in \mathcal{E} \tag{6.2}$$

where $(a_\mathbf{n}^x)$ is a sequence of normalization and $K_\mathbf{n}(\cdot)$ is a function defined from $\mathbb{R}^+$ to $\mathbb{R}$ which depends on the kernel.[3] Dabo-Niang and Yao[3] go into further detail about the assumptions on the kernel, convergence and dependency conditions.

6.2 Uniform Consistency

Before we continue, we must mention that we will only consider the uniform consistency of $f_\mathbf{n}$ over a set $\mathcal{G}$. That is, we are interested in a set $\mathcal{G}$ such that $\mathcal{G} \subset \mathcal{G}_\mathbf{n} := \bigcup_{j=1}^{d_\mathbf{n}} B(x_{\mathbf{i}_j}, r_\mathbf{n})$, where $d_\mathbf{n}$ is some integer and for $j = 1, \ldots, d_\mathbf{n}$, $B(x_{\mathbf{i}_j}, r_\mathbf{n})$ is the open ball of center $x_{\mathbf{i}_j} \in \mathcal{E}$ and radius $r_\mathbf{n} > 0$. For simplicity, we will write $x_{\mathbf{i}_j}$ as x_j. The set $\mathcal{G}$ can be created by taking a finite number of curves, $x_1, \ldots, x_{d_\mathbf{n}}$, and constructing a set of open balls $B(x_j, r_\mathbf{n})$, $j = 1, \ldots, d_\mathbf{n}$ with $r_\mathbf{n}$ such that $\mathcal{G}_\mathbf{n} = \bigcup_{j=1}^{d_\mathbf{n}} B(x_j, r_\mathbf{n})$ covers the whole set of observations of interests.[3] For example, the set $\mathcal{G}$ for two-dimensional spatially related curves can be seen as the set of disks of a certain radius that covers all of the curves in the region. Figure 6.1 shows an example of some of the balls that could be in $\mathcal{G}$.

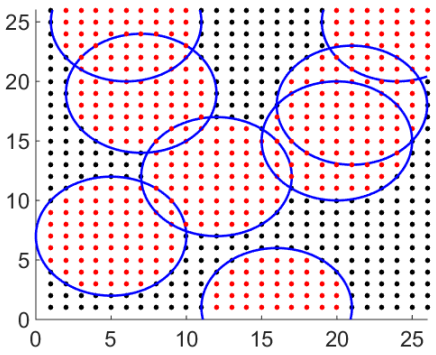


Figure 6.1: Part of the set $\mathcal{G}$ for a rectangular region $\mathcal{I}_{(26,26)}$. The points represent curves. The red points represent curves in the open balls, outlined in blue, of radii 5. Note that this is only part of the set $\mathcal{G}$ since it does not cover all of the curves.

6.3 Kernel Density Estimator

Similar to the finite dimensional case, the kernel density estimator for infinite dimensional space often relies on a kernel function and a smoothing parameter. Specifically, we will work in the case where the probability distribution of X satisfies some concentration condition as in the *i.i.d.* case.[3] With this in mind, Eq. (6.2) can be rewritten as

$$\tilde{f}_{\mathbf{n}}(x) = \frac{1}{\hat{\mathbf{n}} a_{\mathbf{n}}^{x}} \sum_{\mathbf{i} \in \mathcal{I}_{\mathbf{n}}} K\left(\|X_{\mathbf{i}} - x\| / h_{\mathbf{n}}\right), \quad \forall x \in \mathcal{E}, \tag{6.3}$$

where $(h_{\mathbf{n}})$ is a sequence of positive reals (the bandwidths) and K is an integrable kernel.

The normalizing sequence, $(a_{\mathbf{n}}^{x})$, can vary depending on the stationary spatial process and kernel. For example, $a_{\mathbf{n}}^{x} = \mu\left(B\left(x, h_{\mathbf{n}}\right)\right)$ works well in the case of the diffusion process and for some specific kernel. Dabo-Niang and Yao[3] give several examples on finding $(a_{\mathbf{n}}^{x})$ along with the asymptotic results that follow from the kernel density estimator.

6.4 Application

We can now put the KDE into practice. Here, we will be working with $X_{\mathbf{i}}$'s with values on a normed space, $\mathcal{E} = C(0,1)$, with the norm defined by $\|x\| = \int_0^1 |x(t)| dt$ for $x \in \mathcal{E}$. For the measure, μ, we will use a new distribution of $Y_{\mathbf{i}}$'s. Then, it can be shown that the coefficients, $\alpha_{\mathbf{n}}^{x}$, can be calculated using

$$a_{\mathbf{n}}^{x} = \mathbf{E}\left[K\left(\frac{\|Y - x\|}{h_{\mathbf{n}}}\right)\right] \tag{6.4}$$

and the previous theoretical results still hold with the same assumptions.[3]

Recall that the kernel density estimator will be defined over some rectangular region $\mathcal{I}_{\mathbf{n}}$. If the process $(X_{\mathbf{i}})$ is observed over the set $\mathcal{O}_{\mathbf{n}} \supseteq \mathcal{I}_{\mathbf{n}}$, then $(x_{\mathbf{j}}, \mathbf{j} \in \mathcal{O}_{\mathbf{n}})$ is the set of observations. In order to use the spatial dependency at any set $\mathbf{j} \in \mathcal{O}_{\mathbf{n}}$, we will compute $\tilde{f}_{\mathbf{n}}(x_{\mathbf{j}})$ from the observations on $V_{\mathbf{n},\rho_{\mathbf{n}}}^{\mathbf{j}} = \{\mathbf{i} \in \mathcal{O}, \|\mathbf{i} - \mathbf{j}\| < \rho_{\mathbf{n}}\}$, called a vicinity of $\mathbf{j}$, where $(\rho_{\mathbf{n}})$ is an increasing sequence of positive reals called the radii of the vicinity.[3] Instead of using $\mathcal{I}_{\mathbf{n}}$ we will now be using $V_{\mathbf{n},\rho_{\mathbf{n}}}^{\mathbf{j}}$, and Eq. (6.3), can be rewritten as

$$\tilde{f}_{\mathbf{n}}(x_{\mathbf{j}}) = \frac{1}{\hat{\mathbf{n}} a_{\mathbf{n}}^{x}} \sum_{\mathbf{i} \in V_{\mathbf{n},\rho_{\mathbf{n}}}^{\mathbf{j}}} K\left(\frac{\|x_{\mathbf{i}} - x_{\mathbf{j}}\|}{h_{\mathbf{n}}}\right). \tag{6.5}$$

Then, by specifying a set of radii, $S(\rho)$, and bandwidths, $S(h)$, and $\mathbf{j} \in \mathcal{O}_{\mathbf{n}}$, Eq. (6.5) can be evaluated using the normalizing sequence discussed below. This allows for an optimal radius and bandwidth to be calculated by

minimizing the entropy over the sets $S(\rho)$ and $S(h)$.[3] Following this, we will assume that we have the optimal bandwidth and radius.

Next, we must determine the coefficient sequence $(a_{\mathbf{n}}^{x})$, which is dependent on μ. As mentioned, we will use a new distribution as our measure. Essentially, this will allow us to determine if two spatial distributions, $F^{(1)}$ and $F^{(2)}$, are equal or not. Let $X_{\mathbf{i}}^{(1)}$ and $X_{\mathbf{i}}^{(2)}$ represent the processes for $F^{(1)}$ and $F^{(2)}$, respectively. Now, we will use $F^{(2)}$ as the measure, $\mu = F^{(2)}$, which leads to the coefficients being estimated by

$$a_{\mathbf{n}}^{x_{\mathbf{j}}} = \frac{1}{\hat{\mathbf{n}}_{\mathbf{j}}} \sum_{\mathbf{i} \in V_{\mathbf{n}}^{\mathbf{j}}} K\left(\frac{\left\| X_{\mathbf{i}}^{(2)} - x_{\mathbf{j}} \right\|}{h_{\mathbf{n}}} \right), \tag{6.6}$$

for each $x_{\mathbf{j}}$, $\mathbf{j} \in \mathcal{O}_{\mathbf{n}}$, observation from $X_{\mathbf{i}}^{(1)}$. Taking into consideration that we are using the vicinity of $\mathbf{j}$ now, the kernel density estimator, Eq. (6.3), can be rewritten as a ratio comparing the two distributions,

$$\tilde{f}_{\mathbf{n}}(x_{\mathbf{j}}) = \frac{\sum_{\mathbf{i} \in V_{\mathbf{n}}^{\mathbf{j}}} K\left(\frac{\left\| X_{\mathbf{i}}^{(1)} - x_{\mathbf{j}} \right\|}{h_{\mathbf{n}}} \right)}{\sum_{\mathbf{i} \in V_{\mathbf{n}}^{\mathbf{j}}} K\left(\frac{\left\| X_{\mathbf{i}}^{(2)} - x_{\mathbf{j}} \right\|}{h_{\mathbf{n}}} \right)}. \tag{6.7}$$

6.5 Simulation

We simulated two different sets of curves and evaluated the kernel for each observations. We will use $N = 2$, with the simulated curves being located in the region $\mathcal{I}_{(26,26)} = \{(i,j), 1 \leq i,j \leq 26\}$. The stationary Gaussian random field with mean m and covariance function defined by $C(h) = \sigma^2 \exp(-(\frac{\|h\|}{s})^2)$, $h \in \mathbb{R}^2$ and $s > 0$ is denoted $GRF(m, \sigma^2, s)$.[3] The Gaussian random field uses the mean, variance, and spread to create a field over a probability space, which represents a multivariate Gaussian distribution. This, along with a two-dimensional CDF, denoted $F_{(i,j)}$ and $H_{(i,j)}$, will be used to generate the datasets.

The first dataset, $X_{(ij)}^{(1)}$, consists of two different groups of curves on $C[0,1]$, created from

$$X_{(i,j)}^{(1)}(t) = \begin{cases} F_{(i,j)}(t-0.5)^3 + \varepsilon_{(i,j)} & \text{for } (i,j) \in R_1 \text{ (Group 1)} \\ F_{(i,j)} \cos(2\pi t)^5 + \varepsilon_{(i,j)} & \text{for } (i,j) \in R_2 \text{ (Group 2)} \end{cases}, \tag{6.8}$$

where $\varepsilon = GRF(0, 20, 5)$, and R_1, R_2 are two disjoint sets of locations in the region $\mathcal{I}_{(26,26)}$, which can be seen in Figure 6.2.

The second dataset, $X_{(ij)}^{(2)}$, can be simulated in two different ways. Case 1 is where $X_{(i,j)}^{(2)}(t) = F_{(i,j)}(t-0.5)^3 + \varepsilon_{(i,j)} + \epsilon_{(i,j)}$ with $\epsilon_{(i,j)} \sim N(0, 0.01)$. This results in the second dataset to be calculated from

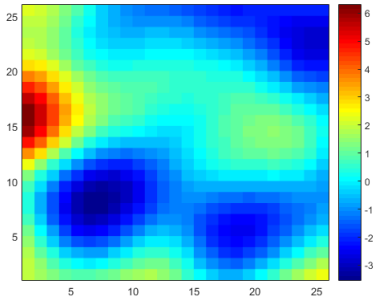
$$\begin{array}{cc} X_{(i,j)}^{(2)}(t) = X_{(i,j)}^{(1)}(t) + \epsilon_{(i,j)} & \text{if } (i,j) \in R_1 \\ X_{(i,j)}^{(2)}(t) \neq X_{(i,j)}^{(1)}(t) & \text{otherwise.} \end{array} \tag{6.9}$$

Case 2 is where $X_{(i,j)}^{(2)}(t) = H_{(i,j)} \cos(2\pi t)^5 + \varepsilon_{(i,j)} + \epsilon_{(i,j)}$, where $H_{(i,j)}$ is the field that coincides with $F_{(i,j)}$ on $\mathcal{I}_{(26,26)} \backslash \{[16,23] \times [4,11]\}$. This results in the second dataset to be calculated from

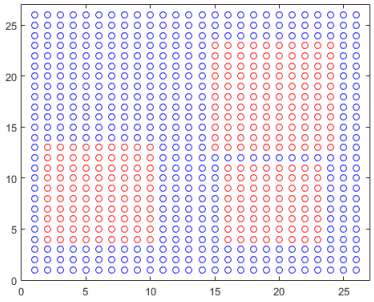
$$\begin{array}{cc} X_{(i,j)}^{(2)}(t) = X_{(i,j)}^{(1)}(t) + \epsilon_{(i,j)} & \text{if } (i,j) \in R_2 \backslash \{[16,23] \times [4,11]\}, \\ X_{(i,j)}^{(2)}(t) \neq X_{(i,j)}^{(1)}(t) & \text{otherwise.} \end{array} \tag{6.10}$$

The first dataset and the dataset for cases 1 and 2 can be seen in Figure 6.3.

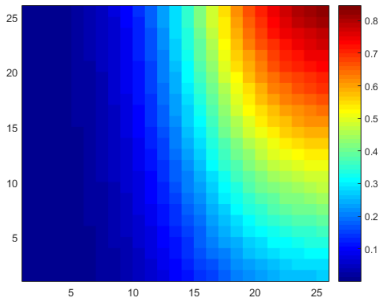
For this specific simulation, Dabo-Niang and Yao[3] have calculated the optimal radius, 5, and the optimal bandwidth, 0.05 for Case 1 and 0.15 for Case 2. Now, using the optimal parameters and the kernel densiry estimator, Eq. (6.7), a representation of the kernel density can be graphed. Since we are in the infinite dimensional



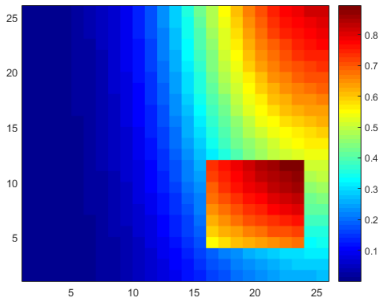
(a) $\varepsilon = GRF\,(0, 20, 5)$



(b) Spatial locations of R_1 (blue) and R_2 (red)

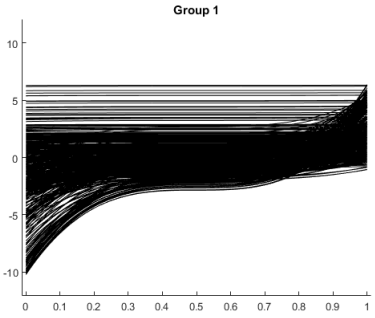


(c) The field $F_{(i,j)}$

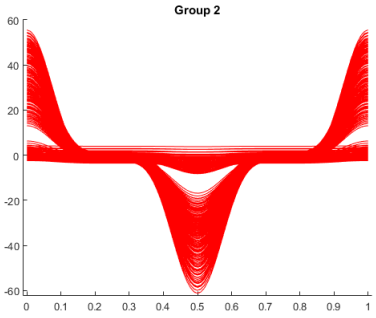


(d) The field $H_{(i,j)}$

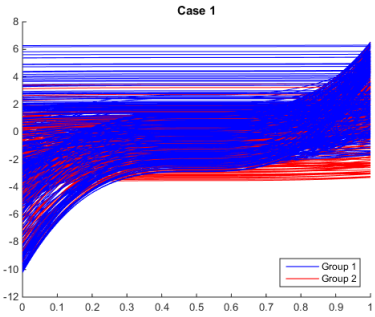
Figure 6.2: The fields and region for simulating dataset.



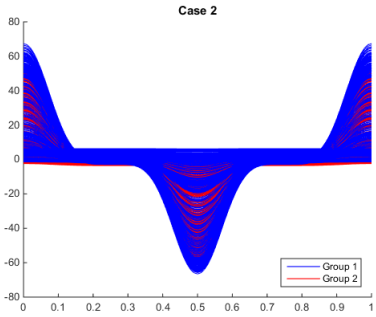
(a) Group 1 of the curves from $X^{(1)}_{(i,j)}\,(t)$



(b) Group 2 of the curves from $X^{(1)}_{(i,j)}\,(t)$

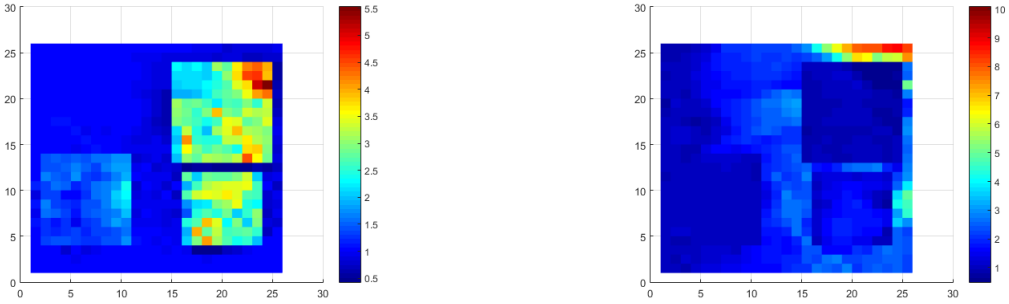


(c) The curves of $X^{(2)}_{(i,j)}\,(t)$ for Case 1



(d) The curves of $X^{(2)}_{(i,j)}\,(t)$ for Case 2

Figure 6.3: The simulated datasets.



(a) Representation using Case 1 as the measure. **(b)** Representation using Case 2 as the measure.

Figure 6.4: Representations of the kernel density estimation.

setting, the actual density functional cannot be visualized, but instead this visualization will depict which curves of the first dataset, $X^{(1)}_{(i,j)}(t)$, are close to the curves of the second dataset, $X^{(2)}_{(i,j)}(t)$. Therefore, the representation has values close to 1 if the datasets are similar, and values close to 0 if they are not. Figure 6.4 shows the results of this simulation. In Case 1, $X^{(2)}_{(i,j)}(t)$ was very similar to Group 1 of $X^{(1)}_{(i,j)}(t)$. Hence, R_1 in the representation has values close to 1, indicating that the curves in this region of $X^{(1)}_{(i,j)}(t)$ are probably equal to those of $X^{(2)}_{(i,j)}(t)$. In Case 2, $X^{(2)}_{(i,j)}(t)$ was very similar to Group 2 of $X^{(1)}_{(i,j)}(t)$, with the exclusion of $[16,23] \times [4,11]$. Hence, $R_2 \setminus \{[16,23] \times [4,11]\}$ in the representation has values close to 1, indicating that the curves in this region of $X^{(1)}_{(i,j)}(t)$ are probably equal to those of $X^{(2)}_{(i,j)}(t)$. However, the region $[16,23] \times [4,11]$ has values very close to 1, along with some other areas along the left-hand side of the representation. Finding a new optimal bandwidth and radius would be ways to improve the results for this case.

7. DISTRIBUTION FIELDS FOR CLASSIFYING FUNCTIONAL DATA

7.1 Distribution Fields

A Distribution Field (DF) is an array of probability distributions, one at each point of the "field". The probability at each point is defined by a feature value at that point.[23] The DFs used in this paper are represented by matrices with two dimensions: the first dimension, the rows, being the time indices of the functional data and the second dimension, the columns, being bins based on a chosen feature. The bins are created by selecting a set number of bins and partitioning the feature values based on a chosen maximum and minimum so the bins are of equal size. The feature used to generate a DF from functional data, which are curves in our examples, can be the amplitude of the curve itself, or of any of its derivatives. Each point in the DF is represented by a Kronecker delta function, so every entry is either a 0 or 1.[23] A DF for a curve, f, can be generated by

$$\mathrm{DF}(k,t) = \begin{cases} 1 & \text{if } f(t) \text{ is in the } k\text{th bin} \\ 0 & \text{otherwise} \end{cases}, \tag{7.1}$$

where k is the bin number and t is the time index.

One characteristic of a DF are that the sum in the desired dimension of the DF is one. Since the chosen feature only has one value for each time index, the columns of the DFs in this paper will sum to one. At this point, the DFs contain all of the same information that the original functional data contained. An example of a 5x4 DF is shown in Figure 7.1.

7.2 Convolution

Once the DFs are formed, they can then be convolved to achieve a representation that takes into account the feature values of the surrounding points.[23] In the most basic sense, convolution is a way to measure how much

Time Index

Bins				
	0	1	0	0
	1	0	0	0
	0	0	1	0
	0	0	0	0
	0	0	0	1

Figure 7.1: Example of a 5x4 DF

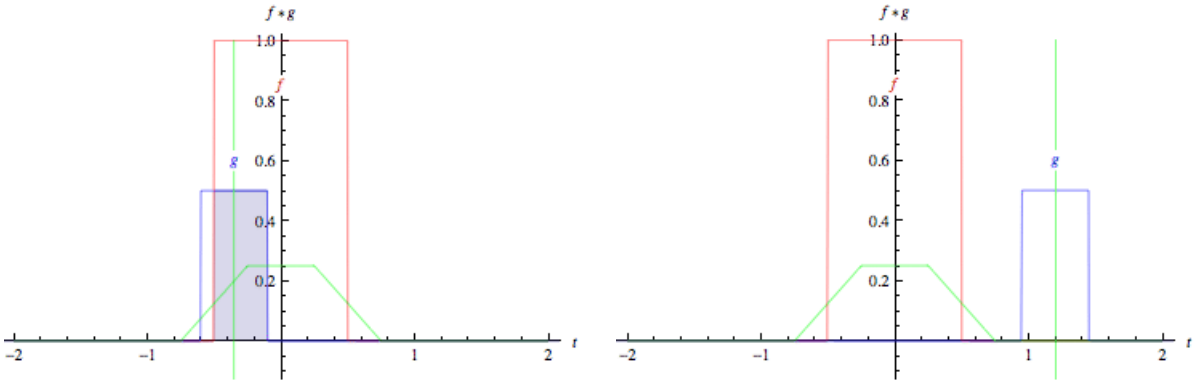


Figure 7.2: As g varies with time and passes over f, there is overlap between the two functions, seen as the dark gray on the left. The convolution of f and g can be thought of as the function that represents this overlap, seen as the green line.[30]

overlap there is between two functions as one function is moved across the other function. Convolution can be represented as an integral that expresses the overlap of function g as it is shifted over function f:

$$f * g = \int f(\tau)\, g(t - \tau)\, d\tau, \tag{7.2}$$

where $*$ is the convolution operator. This can be thought of as "blending" the two functions together.[30] For example, if f and g are functions, then $f * g$ is essentially the function that represents the overlapped area of f and g as g varies over time. An example of this can be seen in Figure 7.2.

In order to convolve distribution fields, we must use some kernel. If the kernel used is one dimensional, the convolution can be performed across the rows or down the columns. If the convolution is performed across the rows, the DF must be padded on the sides with columns. The number of columns depends on the size of the kernel used for convolution and each component in the columns depends on the number of rows in the DF:

$$\text{column padding} \coloneqq \begin{cases} \text{number of columns} & = (\text{size of kernel} - 1)/2 \\ \text{entry value} & = \frac{1}{\text{number of rows in DF}} \end{cases}. \tag{7.3}$$

Where the number of columns is the number of columns that need to be added to each side of the DF and the entry value is the value that needs to be in every cell of the padding to ensure the columns of the DF sum to 1. Adding the padding in this manner preserves the fact that we are convolving a DF and ensures that the result of the convolution will be a DF. After the convolution, the padding can be removed.

If the convolution is performed across the columns, the DF must be padded on the top and bottom with

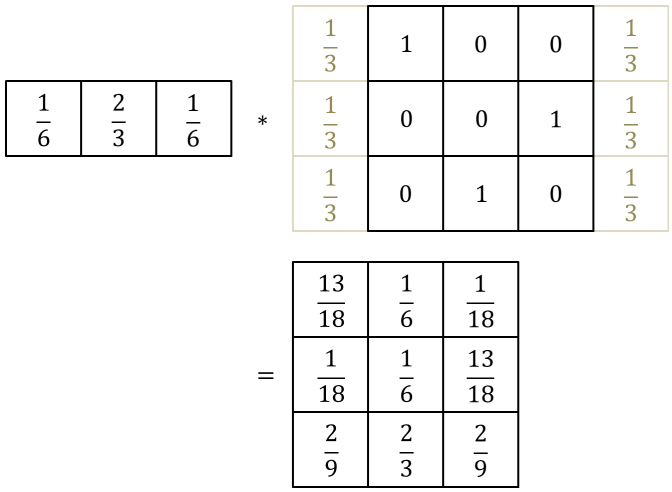


Figure 7.3: Convolving across the rows of a DF with a 1D kernel. The lighter columns of the distribution field represent the padding.

rows. The number of rows depends on the size of the kernel and each component in the row is zero:

$$\text{row padding} := \begin{cases} \text{number of of rows} & = (\text{size of kernel} - 1)/2 \\ \text{entry value} & = 0 \end{cases}. \tag{7.4}$$

Again, the purpose of this padding is to preserve the fact that we are convolving a DF and to ensure that the result is a DF. However, when convolving down the columns, we cannot simply remove the padding when we are done. Instead, we take the sum of the columns from the row padding and weight the sum based on the number of rows in the DF. This results in a weighted row, which must then be added to each row of the convolution result in order for the columns to sum to 1 again. See Figure 7.3 for a depiction of convolving a DF with a one dimensional kernel. We began exploring using DFs for classification by using the k-Nearest Neighbors (kNN) algorithm to classify a new, unlabeled DF based on the k closest training DFs.

7.3 k-Nearest Neighbors

The kNN algorithm is a type of classification algorithm used to classify new unlabeled data based on a similarity measure to a training dataset with known labels. The similarity measure is usually a distance measure such as Euclidean distance or the L1 norm, but there are other distance measures that can be used. The new case is classified based on the majority class of its kNN, determined by the distance measure. For example if $k = 1$, then the new case would be assigned a class solely on the class its one closest neighbor. The optimal value of k is chosen by either inspecting the data or by retrospectively using cross-validation. Using a larger k value, or optimally a k between three and ten, creates a more precise prediction because it will reduce noise in the dataset.[22] An example of kNN can be seen in Figure 7.4 .

7.4 Application

Using DFs and a classification algorithm, such as kNN, we can classify unlabeled curves. The advantage of using DFs is that we are able to get a better representation of the features of curves in different classes. Then by using a classifier the DFs gives a more accurate classification of curves than just using the classification algorithm on the curves themselves. As mentioned earlier, the features of a curve that is used can be the amplitude of the curve itself, or of any of its derivatives.

To begin the classification, we begin with a set of labeled training curves and unlabeled testing curves. We calculate the DFs of the training curves, based on whatever features are chosen. The DFs for the testing curves are generated in a similar manner. Using a kernel, such as a Gaussian kernel, the DFs are then convolved to smooth out the probabilities. Doing so minimizes any in-class data variability, which will minimize classification

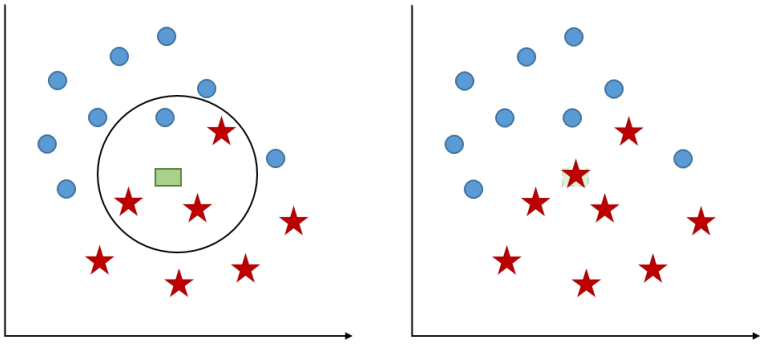


Figure 7.4: Example of the kNN algorithm with $k = 4$. In the image of the left, we apply the algorithm to the green rectangle. Since 3 of the 4 neighbors are red stars, we classify the green rectangle as a red star, seen in the image on the right.

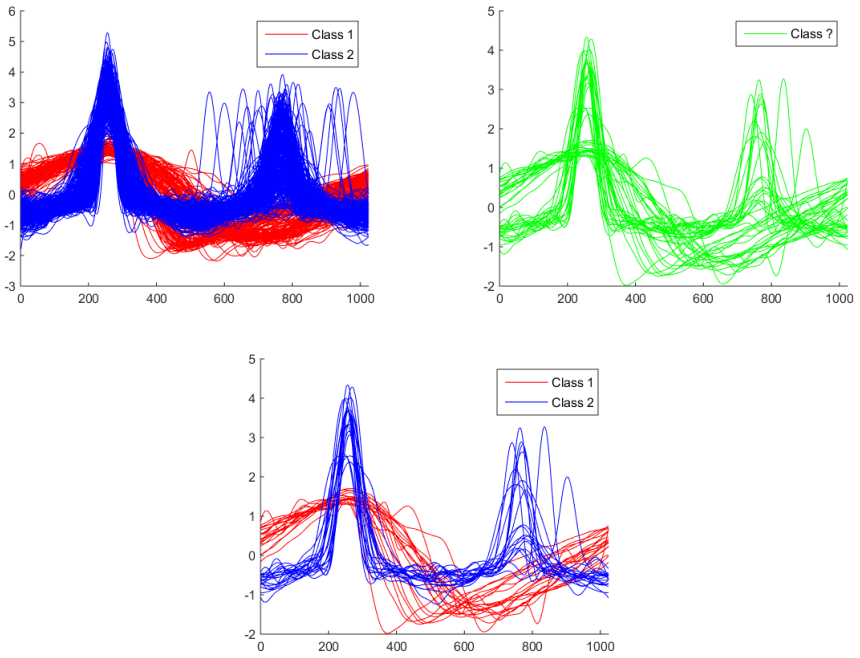


Figure 7.5: The top left shows training curves for two different classes. The top right are some testing curves whose classes are not yet known. Using DFs and kNN, the testing curves were classified correctly, which can be seen in the bottom graph.

error. Using a classification algorithm, we can then classify the testing curves based on their DFs and the training curves DFs.

A direct application of DF classification can be seen in Figure 7.5. Here, we are taking two different classes of curves from a starlight dataset. Using the training curves, we created bins based on the amplitude of the curves. With these bins, we created a DF for each of the training curves.

Once these DFs were generated, they were convolved across the rows and then down the columns. The testing curves underwent the same process using the same bins used for the training curves. Next, the testing DFs were classified based on kNN with the training DFs. Finally, the testing curves received the same classes as their corresponding DFs. Section 8 will go more in-depth about using DFs for classification and creating DFs for the curves based on an optimal combination of its features.

8. OPTIMIZATION OF CLASSIFYING WITH DISTRIBUTION FIELDS

8.1 Metrics

As mentioned in Section 7, we can use DFs based on features of curves for classification. However, just using a single feature is limiting the capabilities of the classification abilities because different classes may look similar in regards to one feature, but not another. Before getting into the actual optimization, an understanding of metrics is needed. Distance measures come from the idea of a metric, which is a mapping $D : X \times X \to \Re_0^+$ over a vector space X. For all vectors, $\forall \mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k \in X$, the metric must satisfy the following properties:

1. $D(\mathbf{x}_i, \mathbf{x}_j) + D(\mathbf{x}_j, \mathbf{x}_k) \geq D(\mathbf{x}_i, \mathbf{x}_k)$, (triangle inequality)
2. $D(\mathbf{x}_i, \mathbf{x}_j) \geq 0$, (non-negativity)
3. $D(\mathbf{x}_i, \mathbf{x}_j) = D(\mathbf{x}_j, \mathbf{x}_i)$, (symmetry)
4. $D(\mathbf{x}_i, \mathbf{x}_j) = 0 \iff \mathbf{x}_i = \mathbf{x}_j$, (distinguishability)

If the fourth property does not hold, the metric is considered a pseudometric.[29]

The most common distance measure is the Euclidean distance. Given $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{y} \in \mathbb{R}^n$, the Euclidean distance is defined as

$$\begin{aligned} d(\mathbf{x}, \mathbf{y}) &= \| \mathbf{x} - \mathbf{y} \|_2^2 \\ &= (\mathbf{x} - \mathbf{y})^T (\mathbf{x} - \mathbf{y}). \end{aligned} \tag{8.1}$$

One family of distance measures that relies on computing Euclidean distances is defined by performing linear transformations to a vector $\mathbf{x}$: $\mathbf{x}' = \mathbf{L}\mathbf{x}$. If $\mathbf{L}$ is a real-valued matrix, we are guaranteed a positive semidefinite matrix $\mathbf{M}$, such that

$$\mathbf{M} = \mathbf{L}^T \mathbf{L}. \tag{8.2}$$

This matrix $\mathbf{M}$ can be used to form a distance measure, known as the Mahalanobis metric. The Mahalanobis metric is defined to be $d_{\mathbf{M}}(\mathbf{x}, \mathbf{y}) = (\mathbf{x} - \mathbf{y})^T \mathbf{M} (\mathbf{x} - \mathbf{y})$, where $\mathbf{M} \in \mathbb{R}^{n \times n}$ such that $\mathbf{M} \succeq 0$. Since $\mathbf{M}$ has to be square, symmetric, and positive semidefinite, we can use Eq. (8.2) to find that[29]

$$\begin{aligned} d_{\mathbf{M}}(\mathbf{x}, \mathbf{y}) &= \| \mathbf{x} - \mathbf{y} \|_{\mathbf{M}}^2 \\ &= (\mathbf{x} - \mathbf{y})^T \mathbf{M} (\mathbf{x} - \mathbf{y}) \\ &= (\mathbf{x} - \mathbf{y})^T \mathbf{L}^T \mathbf{L} (\mathbf{x} - \mathbf{y}) \\ &= [\mathbf{L}(\mathbf{x} - \mathbf{y})]^T [\mathbf{L}(\mathbf{x} - \mathbf{y})] \\ &= \| \mathbf{L}\mathbf{x} - \mathbf{L}\mathbf{y} \|_2^2 . \end{aligned} \tag{8.3}$$

However, DFs are matrices, not vectors, so we need a matrix distance measure in order to compute the distances. A common matrix distance measure is the Frobenius norm, which for a matrix $\mathbf{A}$ can be calculated by

$$\| \mathbf{A} \|_F = \sqrt{tr(\mathbf{A}^T \mathbf{A})}, \tag{8.4}$$

$$\| \mathbf{A} \|_F^2 = tr(\mathbf{A}^T \mathbf{A}). \tag{8.5}$$

For our optimization, we will be learning our metric which will be parameterized in terms of $\mathbf{L}$. By finding $\mathbf{L}$ instead of $\mathbf{M}$, we avoid the complications of having to ensure that $\mathbf{M}$ is square, symmetric, and positive semidefinite. Once $\mathbf{L}$ has been learned, the distance metric becomes[29]

$$\| \mathbf{A} \|_{\mathbf{L}}^2 = \| \mathbf{L}\mathbf{A} \|_F^2 = tr\{\mathbf{A}^T \mathbf{L}^T \mathbf{L} \mathbf{A}\}. \tag{8.6}$$

8.2 Optimization Techniques

To perform this optimization, we will be using concave-convex optimization.[33] Concave-convex optimization is defined by the objective function

$$\min\ f(x) - g(x), \tag{8.7}$$

where $f(x)$ and $g(x)$ are both convex. Since $g(x)$ is convex, $-g(x)$ is concave, thus making this a concave-convex problem. To solve this problem, we need to find the convex approximation, $\hat{f}(x)$, which is simply replacing $-g(x)$ with a linearization based on the current solution, x_k:[10]

$$\hat{f}(x) = f(x) - g(x_k) - \nabla g(x_k)^T (x - x_k). \tag{8.8}$$

If $\hat{f}(x)$ does not have a closed form, then gradient descent can be performed. The gradient descent will find the updated solution, x, based on the current solution, which in turn will minimize Eq. (8.7). Once the difference between $\hat{f}(x)$ and $\hat{f}(x_k)$ is within some error threshold, the gradient descent is complete.

8.3 Optimization Notation

Before we get into the actual optimization, let us define some of the notation that will be used.

- $\alpha = 1, \ldots, K$: total number of classes
- $i = 1, \ldots, N$: number of curves
- $j = 1, \ldots, M$: number of features
- N_α : number of curves in class α
- γ : regularizer term, where $\gamma \geq 0$
- DF^{ij} : DF for curve i, feature j
- $DF^{\alpha j}$: mean DF for class α, feature j
- $\boldsymbol{\lambda}^\alpha$: the weights for class α

Note that each feature of each class has its own weight. So, for any α, $\boldsymbol{\lambda}^\alpha = [\lambda_1^\alpha\ \lambda_2^\alpha\ \ldots\ \lambda_M^\alpha]^T$. Also, the weights for each class are subject to some constraints:

$$\begin{aligned} \sum_j \lambda_j^\alpha = 1 \quad &\Leftrightarrow \quad \mathbf{1}^T \boldsymbol{\lambda}^\alpha = 1, \text{ and} \\ \lambda_j^\alpha \geq 0,\ \forall j, \alpha \quad &\Leftrightarrow \quad \boldsymbol{\lambda}^\alpha \succeq \mathbf{0},\ \forall \alpha. \end{aligned}$$

For this optimization, we will be optimizing the weights for each class and the metric $\mathbf{M}$, which will be learned from $\mathbf{L}$ since $\mathbf{M}$ decomposes as $\mathbf{L}^T\mathbf{L}$. The objective function we use is as follows:

$$\min_{\boldsymbol{\lambda}^\alpha, \mathbf{L}} E = \min_{\boldsymbol{\lambda}^\alpha, \mathbf{L}} \frac{1}{2} \sum_{\alpha=1}^{K} \left[\frac{1}{N_\alpha} \sum_{i \in C_\alpha}^{N_\alpha} \| DF^i - \widetilde{DF}^{\alpha} \|_{\mathbf{L}}^2 - \frac{1}{N - N_\alpha} \sum_{i \notin C_\alpha}^{N - N_\alpha} \| DF^i - \widetilde{DF}^{\alpha} \|_{\mathbf{L}}^2 \right] + \frac{\gamma}{2} \| \mathbf{L} \|_F^2, \tag{8.9}$$

where

$$
\begin{aligned}
DF^i &= \sum_j^M \lambda_j^\alpha DF^{ij}, \text{ and} \\
\widetilde{DF}^{\alpha} &= \sum_j^M \lambda_j^\alpha \widetilde{DF}^{\alpha j} = \sum_j \lambda_j^\alpha \sum_{i \in C_\alpha}^{N_\alpha} \frac{1}{N_\alpha} DF^{ij}.
\end{aligned}
$$

Eq. (8.9) is minimizing the distance between the class mean DFs and the DFs within the same class, while maximizing the distance with the DFs in other classes. We perform this optimization using block descent, optimizing $\mathbf{L}$ first. We continue until the difference between E and the previous value of E is less than some error threshold: $|E - E_{old}| < \varepsilon$.

8.4 Optimizing L

We begin by optimizing $\mathbf{L}$ for a fixed class weight $\boldsymbol{\lambda}^\alpha$. The objective function, Eq. (8.9), becomes

$$
\begin{aligned}
\min_{\mathbf{L}} E_{\mathbf{L}} &= \min_{\mathbf{L}} \frac{1}{2} \sum_{\alpha=1}^{K} \Bigg[\frac{1}{N_\alpha} \sum_{i \in C_\alpha}^{N_\alpha} \| \underbrace{\sum_j^M \lambda_j^\alpha (DF^{ij} - \widetilde{DF}^{\alpha j})}_{D_{i\alpha}} \|_{\mathbf{L}}^2 \\
&\quad \ldots - \frac{1}{N - N_\alpha} \sum_{i \notin C_\alpha}^{N - N_\alpha} \| \underbrace{\sum_j^M \lambda_j^\alpha (DF^{ij} - \widetilde{DF}^{\alpha j})}_{\bar{D}_{i\alpha}} \|_{\mathbf{L}}^2 \Bigg] + \frac{\gamma}{2} \| \mathbf{L} \|_F^2 \\
&= \min_{\mathbf{L}} \underbrace{\frac{1}{2} \sum_{\alpha=1}^{K} \frac{1}{N_\alpha} \sum_{i \in C_\alpha}^{N_\alpha} tr\{D_{i\alpha}^T \mathbf{L}^T \mathbf{L} D_{i\alpha}\} + \frac{\gamma}{2} tr\{\mathbf{L}^T \mathbf{L}\}}_{f(\mathbf{L})} \\
&\quad \ldots - \underbrace{\frac{1}{2} \sum_{\alpha=1}^{K} \frac{1}{N - N_\alpha} \sum_{i \notin C_\alpha}^{N - N_\alpha} tr\{D_{i\alpha}^T \mathbf{L}^T \mathbf{L} D_{i\alpha}\}}_{g(\mathbf{L})}.
\end{aligned} \tag{8.10}
$$

Since $f(\mathbf{L})$ and $g(\mathbf{L})$ are both convex, we can use concave-convex optimization to find the optimal $\mathbf{L}$:

$$
\hat{f}(\mathbf{L}) = f(\mathbf{L}) - g(\mathbf{L}_k) - tr\{\nabla g^T(\mathbf{L}_k)(\mathbf{L} - \mathbf{L}_k)\}. \tag{8.11}
$$

By reducing $f(\mathbf{L})$ and $g(\mathbf{L})$, we get

$$
\begin{aligned}
f(\mathbf{L}) &= \frac{1}{2} tr\left\{ \sum_{\alpha=1}^{K} \frac{1}{N_\alpha} \sum_{i \in C_\alpha}^{N_\alpha} D_{i\alpha}^T \mathbf{L}^T \mathbf{L} D_{i\alpha} \right\} + \frac{\gamma}{2} tr\{\mathbf{L}^T \mathbf{L}\} \\
&= \frac{1}{2} tr\left\{ \mathbf{L}^T \mathbf{L} \Big(\underbrace{\sum_{\alpha=1}^{K} \frac{1}{N_\alpha} \sum_{i \in C_\alpha}^{N_\alpha} D_{i\alpha} D_{i\alpha}^T}_{D} + \gamma I \Big) \right\} \\
&= \frac{1}{2} tr\{\mathbf{L}^T \mathbf{L} (D + \gamma I)\} \\
&= \frac{1}{2} tr\{\mathbf{L}^T \mathbf{L} D\} + \frac{\gamma}{2} tr\{\mathbf{L}^T \mathbf{L}\}
\end{aligned} \tag{8.12}
$$

$$
\begin{aligned}
g(\mathbf{L}) &= \frac{1}{2} tr\Big\{ \sum_{\alpha=1}^{K} \frac{1}{N-N_\alpha} \sum_{i \notin C_\alpha}^{N-N_\alpha} D_{i\alpha}^T \mathbf{L}^T \mathbf{L} D_{i\alpha} \Big\} \\
&= \frac{1}{2} tr\Big\{ \mathbf{L}^T \mathbf{L} \underbrace{\Big(\sum_{\alpha=1}^{K} \frac{1}{N-N_\alpha} \sum_{i \notin C_\alpha}^{N-N_\alpha} D_{i\alpha} D_{i\alpha}^T \Big)}_{\bar{D}} \Big\} \\
&= \frac{1}{2} tr\{ \mathbf{L}^T \mathbf{L} \bar{D} \}. \qquad (8.13)
\end{aligned}
$$

It follows that

$$
\nabla g(\mathbf{L}) = \mathbf{L}\bar{D}. \qquad (8.14)
$$

Now, Eq. (8.11) can be rewritten as

$$
\hat{f}(\mathbf{L}) = \frac{1}{2} tr\{\mathbf{L}^T \mathbf{L} D\} + \frac{\gamma}{2} tr\{\mathbf{L}^T \mathbf{L}\} - \frac{1}{2} tr\{L_k^T \mathbf{L} \bar{D}\} - tr\{(\mathbf{L}_k \bar{D})^T (\mathbf{L} - \mathbf{L}_k)\}. \qquad (8.15)
$$

To find the new value of $\mathbf{L}$, we begin by taking the partial derivative of $\hat{f}(\mathbf{L})$ with respect to $\mathbf{L}$:

$$
\begin{aligned}
\frac{\partial \hat{f}(\mathbf{L})}{\partial \mathbf{L}} &= \mathbf{L} D^T - \mathbf{L}_k \bar{D} + \gamma \mathbf{L} = 0 \\
&\Rightarrow \mathbf{L}(D^T + \gamma I) = \mathbf{L}_k \bar{D} \\
&\Rightarrow \mathbf{L} = \mathbf{L}_k \bar{D} (D^T + \gamma I)^{-1}, \qquad (8.16)
\end{aligned}
$$

where $\mathbf{L}$ is the new optimal value based on the current class weights. Since Eq. (8.16) has a closed form, gradient descent does not need to be performed, and the optimal $\mathbf{L}$ can be calculated directly.

8.5 Optimizing the Feature Weights

Once $\mathbf{L}$ is learned, we optimize Eq. (8.9) for $\boldsymbol{\lambda}^\alpha$, which becomes

$$
\begin{aligned}
\min_{\boldsymbol{\lambda}^\alpha} E_{\boldsymbol{\lambda}^\alpha} &= \min_{\boldsymbol{\lambda}^\alpha} \frac{1}{2} \sum_{\alpha=1}^{K} \Bigg[\frac{1}{N_\alpha} \sum_{i \in C_\alpha}^{N_\alpha} tr\Big\{ \sum_{j}^{M} \lambda_j^\alpha \Big(DF^{ij} - \widetilde{DF}^{\alpha j} \Big)^T \mathbf{L}^T \mathbf{L} \sum_{k}^{M} \lambda_k^\alpha \Big(DF^{ik} - \widetilde{DF}^{\alpha k} \Big) \Big\} \\
&\qquad \ldots - \frac{1}{N-N_\alpha} \sum_{i \notin C_\alpha}^{N-N_\alpha} tr\Big\{ \sum_{j}^{M} \lambda_j^\alpha \Big(DF^{ij} - \widetilde{DF}^{\alpha j} \Big)^T \mathbf{L}^T \mathbf{L} \sum_{k}^{M} \lambda_k^\alpha \Big(DF^{ik} - \widetilde{DF}^{\alpha k} \Big) \Big\} \Bigg] \\
&= \min_{\boldsymbol{\lambda}^\alpha} \frac{1}{2} \sum_{\alpha=1}^{K} \Bigg[\sum_j \sum_k \lambda_j^\alpha \lambda_k^\alpha tr\underbrace{\Big\{ \frac{1}{N_\alpha} \sum_{i \in C_\alpha}^{N_\alpha} \Big(DF^{ij} - \widetilde{DF}^{\alpha j} \Big)^T \mathbf{L}^T \mathbf{L} \Big(DF^{ik} - \widetilde{DF}^{\alpha k} \Big) \Big\}}_{q_{jk}^\alpha} \\
&\qquad \ldots - \sum_j \sum_k \lambda_j^\alpha \lambda_k^\alpha tr\underbrace{\Big\{ \frac{1}{N-N_\alpha} \sum_{i \notin C_\alpha}^{N-N_\alpha} \Big(DF^{ij} - \widetilde{DF}^{\alpha j} \Big)^T \mathbf{L}^T \mathbf{L} \Big(DF^{ik} - \widetilde{DF}^{\alpha k} \Big) \Big\}}_{\bar{q}_{jk}^\alpha} \Bigg] \\
&= \min_{\boldsymbol{\lambda}^\alpha} \frac{1}{2} \sum_{\alpha=1}^{K} \Big[\sum_j \sum_k \lambda_j^\alpha \lambda_k^\alpha \Big(q_{jk}^\alpha - \bar{q}_{jk}^\alpha \Big) \Big] \\
&= \min_{\boldsymbol{\lambda}^\alpha} \frac{1}{2} \sum_{\alpha=1}^{K} \boldsymbol{\lambda}^{\alpha T} (Q_\alpha - \bar{Q}_\alpha) \boldsymbol{\lambda}^\alpha, \qquad (8.17)
\end{aligned}
$$

where $Q_\alpha = \{q_{jk}^\alpha\}$ and $\bar{Q}_\alpha = \{\bar{q}_{jk}^\alpha\}$. This is still subject to the original constraints for the class weights mentioned earlier:

$$\begin{aligned} \mathbf{1}^T \boldsymbol{\lambda}^\alpha &= 1, \forall \alpha, \text{ and} \\ \boldsymbol{\lambda}^\alpha &\succeq \mathbf{0}, \forall \alpha. \end{aligned}$$

Now, for a specific class α we can write

$$\min_{\boldsymbol{\lambda}^\alpha} E_{\boldsymbol{\lambda}^\alpha} = \min_{\boldsymbol{\lambda}^\alpha} \underbrace{\frac{1}{2}\boldsymbol{\lambda}^{\alpha T} Q_\alpha \boldsymbol{\lambda}^\alpha}_{f(\boldsymbol{\lambda}^\alpha)} - \underbrace{\frac{1}{2}\boldsymbol{\lambda}^{\alpha T} \bar{Q}_\alpha \boldsymbol{\lambda}^\alpha}_{g(\boldsymbol{\lambda}^\alpha)} \tag{8.18}$$

It follows that

$$\nabla g(\boldsymbol{\lambda}^\alpha) = \bar{Q}_\alpha \boldsymbol{\lambda}^\alpha.$$

We can then write the convex approximation for the optimization as

$$\begin{aligned} \hat{f}(\boldsymbol{\lambda}^\alpha) &= \frac{1}{2}\boldsymbol{\lambda}^{\alpha T} Q_\alpha \boldsymbol{\lambda}^\alpha - \frac{1}{2}(\boldsymbol{\lambda}_k^\alpha)^T \bar{Q}_\alpha \boldsymbol{\lambda}_k^\alpha - (\boldsymbol{\lambda}_k^\alpha)^T \bar{Q}_\alpha^T (\boldsymbol{\lambda}^\alpha - \boldsymbol{\lambda}_k^\alpha) \\ &= \frac{1}{2}\boldsymbol{\lambda}^{\alpha T} Q_\alpha \boldsymbol{\lambda}^\alpha + \frac{1}{2}(\boldsymbol{\lambda}_k^\alpha)^T \bar{Q}_\alpha \boldsymbol{\lambda}_k^\alpha - (\boldsymbol{\lambda}_k^\alpha)^T \bar{Q}_\alpha^T \boldsymbol{\lambda}^\alpha, \end{aligned} \tag{8.19}$$

where $\boldsymbol{\lambda}_k^\alpha$ is the previous weights for a given class. Since $\hat{f}$ is convex, it can be minimized over $\boldsymbol{\lambda}^\alpha$ using any optimization technique. To do so, we utilize the CVX optimization package for MATLAB with the same constraints on $\boldsymbol{\lambda}^\alpha$.

8.6 Classification

Once the optimal $\mathbf{L}$ and $\boldsymbol{\lambda}^\alpha$'s are calculated, we can begin to classify test curves using a metric based on the optimal $\mathbf{L}$ and optimal $\boldsymbol{\lambda}^\alpha$'s, denoted $\hat{\mathbf{L}}$ and $\hat{\boldsymbol{\lambda}}^\alpha$, respectively. To classify a given test curve, we first create the DFs for the chosen features. For each feature, we take the difference between the test curve's DF and the class mean DFs, which are based on the training curves. Then, we create a weighted sum with the optimal class weights and compute a distance using our metric. We can then assign the class label of whichever class mean DF resulted in the minimum distance:

$$\underset{\alpha}{\operatorname{argmin}} \parallel \sum_j \hat{\lambda}_j^\alpha (DF_{test}^j - \widetilde{DF}^{\alpha j}) \parallel_{\mathbf{L}}^2 . \tag{8.20}$$

This can be rewritten using Eq. (8.6) to become

$$\underset{\alpha}{\operatorname{argmin}} \parallel \hat{\mathbf{L}} \sum_j \hat{\lambda}_j^\alpha (DF_{test}^j - \widetilde{DF}^{\alpha j}) \parallel_F^2 . \tag{8.21}$$

9. EXPERIMENTS

In this section, we discuss the performance of classifying with an optimized combination of DFs, which we will refer to as Distribution Fields for Classifying Functional Data (DFCFD), using functional datasets from the "UCR Time Series Data Mining Archive".[14] The multiclass datasets that we are using are divided into training and testing sets, as shown in Table 9.1. The datasets we use range from 2 classes – the GunPoint and Coffee datasets, up to 37 classes – the ADIAC dataset. Figure 9.1 shows the curves from the Coffee dataset and Figure 9.2 shows the curves and their derivatives from the GunPoint dataset, which would be used to create the DFs.

We compare our results from DFCFD against two other classification methods: kNN of the curves and kNN of the DFs based on only the amplitude of the curves. For both of the methods using kNN, we use the L1

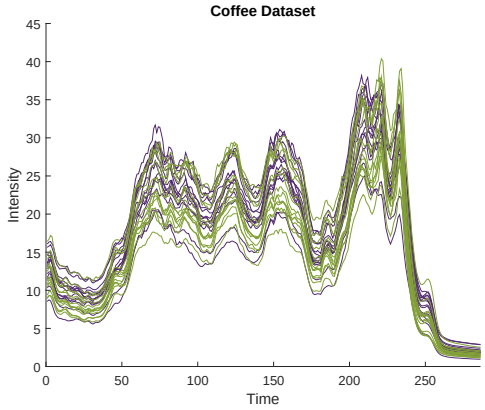


Figure 9.1: Curves from the training set of the Coffee dataset, where purple is class one and green is class two.

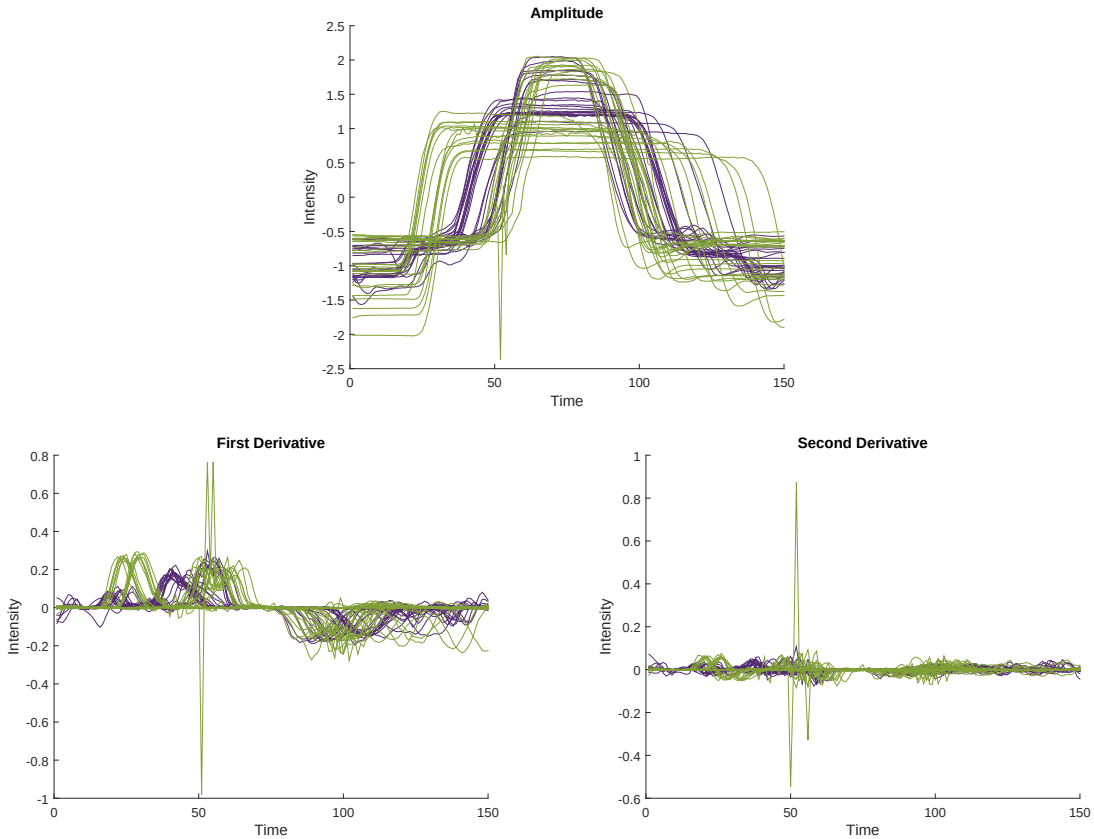


Figure 9.2: Features of the curves from the GunPoint training set, where purple is class one and green is class two.

Table 9.1: Functional datasets with good variation between number of classes, training and testing sizes, and class sizes.

Dataset	Number of Classes	Size of Training Set	Size of Testing Set	Length
Synthetic Control	6	300	300	60
GunPoint	2	50	150	150
CBF	3	30	900	128
Face(four)	4	24	88	350
ADIAC	37	390	391	176
Coffee	2	28	28	286

Table 9.2: Optimal parameters for kNN of DFs based on the amplitude and DFCFD.

Dataset	kNN of Amplitude DFs			DFCFD		
	k	Number of Bins	Size of Kernel	γ	Number of Bins	Size of Kernel
Synthetic Control	5	8	9	0.001	5	5
GunPoint	1	24	9	6	6	5
CBF	1	13	9	1000	15	7
Face(four)	1	27	9	2500	29	9
ADIAC	1	39	9	910	5	7
Coffee	10	18	9	910	15	7

norm as the distance measure. We used k-fold cross-validation across the training sets to determine the optimal parameters for classifying the testing sets. In each method, we performed 4-fold cross-validation, excluding ADIAC which we performed 3-fold cross-validation because not all of the classes had enough curves. Cross-validation for kNN of the curves only required cross-validating over one parameter: k. Cross-validation for kNN of DFs based on the amplitudes required cross-validating over both k and the number of bins, while DFCFD required cross-validating over γ and the number of bins. The parameters for classifying using DFs based on the amplitudes and DFCFD can be seen in Table 9.2. Once we found the optimal parameters, we then used them to classify the testing sets and calculated the classification accuracy using

$$\text{Accuracy} = 100 \cdot \frac{\#\text{ of correctly classified curves}}{\text{Total Number of Curves}}. \tag{9.1}$$

The results achieved from using these three classification methods can be seen in Table 9.3. For all datasets except GunPoint and CBF, DFCFD did better than using kNN. In both these cases using kNN on the amplitude DFs still was able to outperform kNN on the curves. In particular, using DFs to represent the functional data was able to achieve over 90% classification accuracy in all cases except ADIAC. Expanding from binary classification to multiclass classification significantly increases the difficulty of the classification process; yet it did not affect the accuracy of DFCFD much (again except for ADIAC). This is extremely evident with ADIAC, which has 37 classes with small class sizes ranging from 4 to 13 curves in each. Nevertheless, we can see that by representing the functional data using DFs and then performing the classification manages to achieve higher classification accuracies.

10. CONCLUSION

Being able to analyze functional data is an important task since many real-world sensor measurements can be represented as functional data. One such form of analysis is density estimation. While density estimation for finite dimensional data is a well researched topic, density estimation for functional data is a fairly new concept. The implementation of Functional Density Estimation (FDE) would be very useful, but implementing using the current theory would be computationally exhaustive. However, we were able to implement kernel density estimation for spatially dependent functional data, but this severly limits the usefuleness of the density estimator.

For this reason, we focused on a different area of analysis: classification. While there has been a large amount of research done on classifying functional data, we developed a novel approach using Distribution Field (DF)

Table 9.3: Classification accuracy using the three different classifying methods. Using DFs results in a higher accuracy than using just the curves for all datasets.

Dataset	kNN of Curves	kNN of Amplitude DFs	DFCFD
Synthetic Control	88.00	98.67	91.67
GunPoint	95.33	98.00	76.00
CBF	88.89	99.33	85.00
Face(four)	84.10	92.05	97.73
ADIAC	59.85	56.52	68.03
Coffee	96.43	100	100

representation of functions. Our Distribution Fields for Classifying Functional Data (DFCFD) classification method has proven useful and outperformed k-Nearest Neighbors (kNN) on the curves themselves. In the future, we plan to look at other features of the functional data to represent, such as the phase, and cross-validating over additional parameters that may be affecting the accuracy.

ACKNOWLEDGMENTS

The authors acknowledge support from National Science Foundation (NSF) grant No. 1263011. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

REFERENCES

1. Juan Antonio Cuesta-Albertos and Ricardo Fraiman. Impartial trimmed k-means for functional data. *Computational Statistics & Data Analysis*, 51(10):4864–4877, 2007.
2. Antonio Cuevas. A partial overview of the theory of statistics with functional data. *Journal of Statistical Planning and Inference*, 147(0):1 – 23, 2014.
3. Sophie Dabo-Niang and Anne-Françoise Yao. Kernel spatial density estimation in infinite dimension space. *Metrika*, 76(1):19–52, 2013.
4. Nate Eldredge. Wiener measure, May 2006.
5. Frédéric Ferraty. *Recent advances in functional data analysis and related topics*. Springer, Berlin New York, 2011.
6. Frédéric Ferraty and Philippe Vieu. *Nonparametric functional data analysis: Theory and practice*. Springer, New York, 2006.
7. Karen Fuchs, Jan Gertheiss, and Gerhard Tutz. Nearest neighbor ensembles for functional data with interpretable feature selection. *Chemometrics and Intelligent Laboratory Systems*, 2015.
8. A Goia and EG Bongiorno. Clustering for functional data. In *GRASPA WORKING PAPERS*. IT, 2014.
9. Tomasz Górecki, Mirosław Krzyśko, and Waldemar Wołyński. Classification problems based on regression models for multi-dimensional functional data. *STATISTICS*, 1, 2015.
10. Martin Hast, KJ Aström, Bo Bernhardsson, and Stephen Boyd. Pid design by convex-concave optimization. In *Proceedings European Control Conference*, pages 4460–4465. Citeseer, 2013.
11. Lajos Horváth and Piotr Kokoszka. *Inference for functional data with applications*. Springer, New York, NY, 2012.
12. Mia Hubert, Peter J Rousseeuw, and Pieter Segaert. Multivariate and functional classification using depth and distance, 2015.
13. Viji Diane Kannan. Measure theory for applied research, 2014.
14. E. Keogh, X. Xi, L. Wei, and C. Ratanamahatana. The ucr time series classification/clustering homepage, 2011.
15. Steven Krantz. *Real analysis and foundations*. Chapman & Hall/CRC, Boca Raton, Fla, 2005.
16. Nicole Lazar. The big picture: Functional data analysis. *CHANCE*, 27(1):38–40, 2014.

17. Wuan Luo. Wiener chaos expansion and numerical solutions of stochastic partial differential equations. Master's thesis, California Institute of Technology, 2006.
18. G. H. Meisters. Lebesgue measure on the real line. 1997.
19. Jifeng Ning, Wuzhen Shi, Shuqin Yang, and Paul Yanne. Visual tracking based on distribution fields and online weighted multiple instance learning. *Image and Vision Computing*, 31(11):853–863, 2013.
20. J. O. Ramsay and B. W. Silverman. *Functional Data Analysis*. Springer, New York, 2005.
21. David K. Ruch and Patrick J. Van Fleet. *Wavelet Theory: An Elementary Approach with Applications*. John Wiley & Sons, Inc., 2009.
22. Saed Sayad. K nearest neighbors - classification, December 2010.
23. L. Sevilla-Lara and E. Learned-Miller. Distribution fields for tracking. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1910–1917, June 2012.
24. A.V. Skorokhod. Wiener measure. Encyclopedia od Mathematics, 2011.
25. V.I. Sobolev. Wiener integral. Encyclopedia of Mathematics, 2011.
26. Michael E. Taylor. *Measure Theory and Integration*, volume 76. American Mathematical Society, 2006.
27. Marina Vannucci. Nonparametric density estimation using wavelets, 1995.
28. Melvin M Varughese, Rainer von Sachs, Michael Stephanou, and Bruce A Bassett. Nonparametric transient classification using adaptive wavelets, 2015.
29. Kilian Q Weinberger and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10:207–244, 2009.
30. Eric W. Weisstein. Convolution. From MathWorld – A Wolfram Web Resource.
31. Eric W. Weisstein. Lebesgue measure. From MathWorld – A Wolfram Web Resource.
32. Wikipedia. Lebesgue integration – wikipedia, the free encyclopedia, 2015.
33. Alan L Yuille and Anand Rangarajan. The concave-convex procedure (cccp). *Advances in neural information processing systems*, 2:1033–1040, 2002.
34. Zhen Zhang and Hans-Georg Müller. Functional density synchronization. *Computational Statistics & Data Analysis*, 55(7):2234–2249, 2011.

APPENDIX A. MEASURE THEORY

A.1 Functions

A function establishes a relationship between two spaces, called the domain and the range of the function. For the relationship to be considered a function, it must map each element in the domain to exactly one element in the range. While it is allowed for a function's range to have multiple elements from the domain map to it, it is not allowed for an element in the domain to map to multiple elements in the range. Furthermore, it is not necessary for every element in the range to be mapped to from the domain. An example of a function is depicted in Figure A.1.

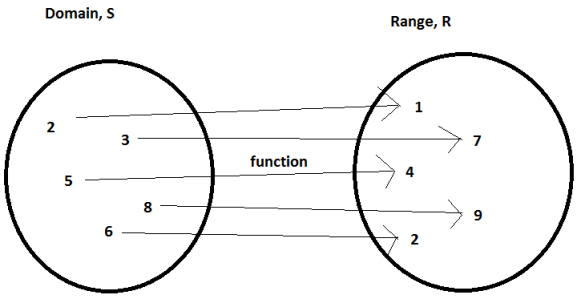


Figure A.1: The function maps every element of the domain to exactly one element of the range.

In addition to the previous rules, a function can only map from single elements. That is, a function cannot map from a subset of elements in the domain to the range. In order to map from a subset of elements in a domain, a new domain must be created where the subset is now an element of the newly formed domain. Since the subset is now a single element, the function can now map from that element, even though it is still a subset of the old domain. Figure A.2 shows how this may be done.

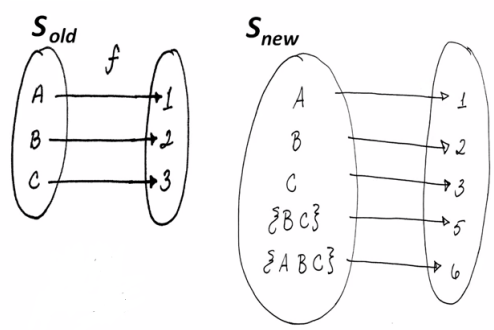


Figure A.2: Mapping a subset of elements from the domain S_{old} using elements of the domain S_{new}.[13]

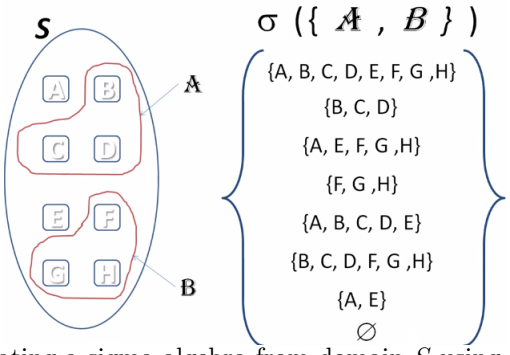


Figure A.3: Creating a sigma algebra from domain S using sets A and B.[13]

A.2 Sigma Algebras and Measurable Spaces

An algebra, denoted $\mathcal{A}$, is a new domain defined on a set S. This means that the algebra is a set of subsets from the defining set, where the defining set is also referred to as the universal set. To construct an algebra, three rules must be followed:[13]

1. $S \in \mathcal{A}$,
2. if $X \in \mathcal{A}$, then $X^c \in \mathcal{A}$,
3. if $X \in \mathcal{A}$ and $Y \in A$, then $X \cup Y \in \mathcal{A}$.

The space defined by an algebra is called a measurable space. Since the algebra itself was defined by set S, the measurable space is written $(S, \mathcal{A})$.

Sigma algebras are a specific type of algebra. Formally, a sigma algebra, $\mathcal{A}$, on a subset, C, of elements is denoted $\mathcal{A} = \sigma(C)$, where $C = \{A, B, \ldots\}$, and $A, B, \ldots \subset S$. The only difference between an algebra and a sigma algebra is in rule 3, which changes to

3. if $\{A_n : n \in N\}$ is a sequence of sets in $\mathcal{A}$, then $\bigcup_N A_n \in \mathcal{A}$.

Figure A.3 shows how to create a sigma algebra given a set S.

One special kind of sigma algebra is called the power set. The power set is the set containing all possible subsets from a set S. Figure A.4 shows how to create the power set.

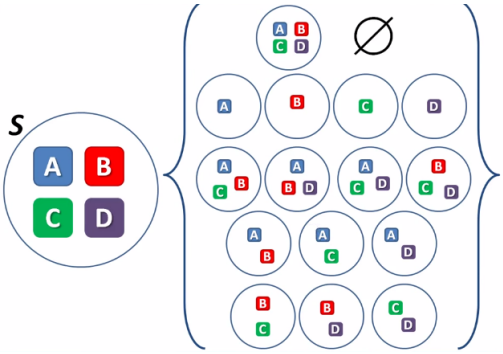


Figure A.4: The power set from the domain S, which is also a sigma algebra.[13]

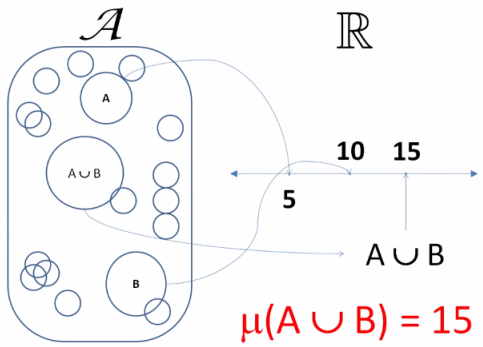


Figure A.5: For disjoint sets A and B, $\mu(A \cup B) = \mu(A) + \mu(B)$.[13]

A.3 Measures and Measure Spaces

A measure, μ, is a function which acts on a sigma algebra, which results in measure space, a measurable space equipped with a measure, denoted $(S, \mathcal{A}, \mu)$. The measure accomplishes this by mapping the sigma algebra to the real numbers. Hence, $\mu : \mathcal{A} \to \mathbb{R}$. Like algebras and sigma algebras, measures have their own three rules they must follow:[13]

1. $\mu(A) \geq 0$, for all $A \in \mathcal{A}$,
2. $\mu(\emptyset) = 0$,
3. for any disjoint collection of sets $\{A_n : n \in N\}$ in $\mathcal{A}$, $\mu\left(\bigcup_N A_n\right) = \sum_N \mu(A_n)$.

An example of rule 3 is shown in Figure A.5.

One important thing to note is that if two sets in the sigma algebra overlap such that the intersection is not the empty set, the measure of the intersection is not the sum of each individual measure. This is because anything in the intersection of the two sets would be counted for by the measure of both sets and therefore would be double counted. This leads to the constraint[13] $\mu\left(\bigcup_N A_n\right) \leq \sum_N \mu(A_n)$. Figure A.6 shows an example of this concept.

A.4 Measurable Functions

For two measurable spaces, $(S, \mathcal{A})$ and $(T, \mathcal{F})$, a function $f : S \to T$ is said to be a measurable function if there exists a function $f^{-1} : \mathcal{F} \to \mathcal{A}$. A property of a measurable function is that the assignment of values in the range of f^{-1}, $\mathcal{A}$, to the elements in the domain, $\mathcal{F}$, are based on the assignment of elements in the range of f, T, to its domain, S. In other words, f^{-1} takes subsets of the range of f and maps them to subsets of the domain of f. Figure A.7 shows an example of a measurable function.

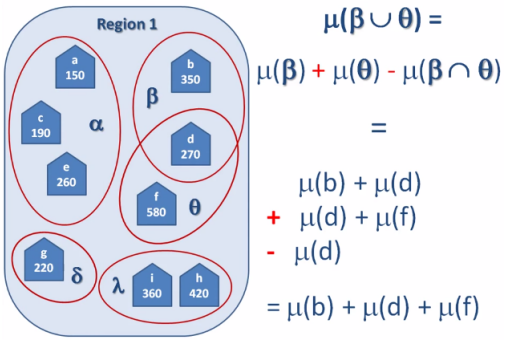


Figure A.6: Since β and θ overlap, $\mu(\beta \cup \theta) \neq \mu(\beta) + \mu(\theta)$. The problem is that $\mu(d)$ is counted twice since it is in β and θ.[13]

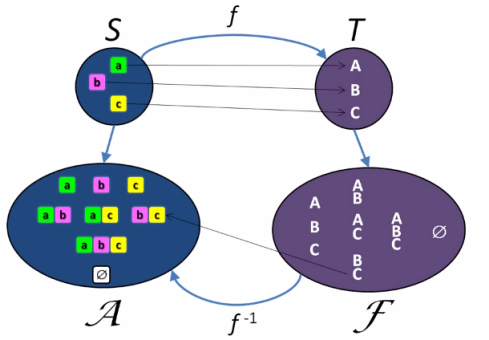


Figure A.7: Function f is a measurable function because f maps S to T and there exists f^{-1} that maps $\mathcal{F}$ to $\mathcal{A}$.[13]

When a measurable function is applied to two measure spaces, $(S, \mathcal{A}, \mu)$ and $(T, \mathcal{F}, \mu^*)$, the function μ^*will assign the same values to $\mathcal{F}$ as the function μ will assign to $\mathcal{A}$. This leads to the following three properties:[13]

1. $\mu = |A|$ for all $A \in \mathcal{A}$,
2. $\mu^* = |\mathcal{A}|$,
3. $\mu^*(F) = \mu\left(f^{-1}(F)\right) = |A|$, for all $F \in \mathcal{F}$.

Figure A.8 gives a specific example of what μ^* actually represents.

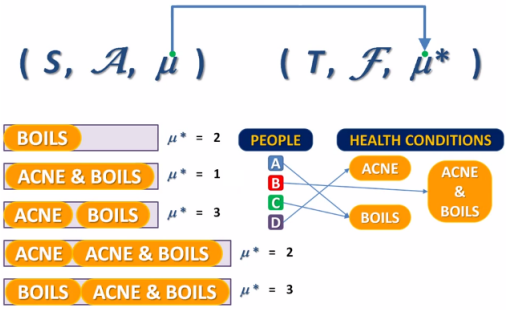


Figure A.8: In this scenario, $\mathcal{A}$ is the power set of all the people and $\mathcal{F}$ is the power set of all the health conditions. So, $\mu^*(F)$ is the size of $A \in \mathcal{A}$ that maps to the element $F \in \mathcal{F}$.[13]

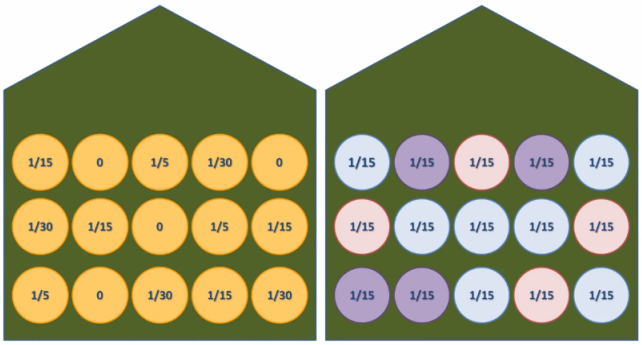


Figure A.9: On the left, there is a probability space with unequal probability sampling. On the right, there is a probability space with equal probability spacing. In both cases, the sum of all of the possible outcomes is equal to 1.[13]

A.5 Probability Spaces

A probability space is a special type of measure space, denoted $(\Omega, \mathcal{A}, P)$. The outcome set, Ω, is the set containing all possible outcomes of the event set, $\mathcal{A}$. The elements of the outcome set are called outcomes and are denoted by ω, while elements of the event set are called events. The outcome set is every possible outcome for a data generating process, which represents a population of interest. The event set is the sigma algebra on the outcome set, which represents the events to which a measure function can be assigned. and is typically the power set of the outcome set. The probability measure, P, is the measure of the probability of a certain outcome. The sum of the probability of every outcome is 1, hence, $P(\Omega) = \sum P(\omega) = 1$. This constraint is what differentiates a probability space from a measure space. Furthermore, this leads to the idea that if a probability space is partitioned, the sum of the probability measure of each partition component will be equal to 1.

There are two main data generating process models, equal probability sampling and unequal probability sampling. In an equal probability sampling, every object in the population of interest has an equal probability of being selected. That is, $P(\omega_i) = P(\omega_j)$ for all i, j. Additionally, an equal probability sampling implies that $P(\omega) = 1/|\Omega|$ for all $\omega \in \Omega$. In an unequal probability sampling, some objects have higher probabilities of being selected than others, and the probabilities vary based on the object and the properties of that object. Therefore, there is some i, j such that $P(\omega_i) \neq P(\omega_j)$. However, regardless of equal or unequal probability sampling, the sum of each individual probability measure must be 1. Figure A.9 shows an example of equal and unequal probability sampling.

APPENDIX B. THE L^2 SPACE

B.1 Basics

To carry out FDE on functions, the estimation needs to be done in a vector space where elements can be treated as functions. For example, $\mathbb{R}^N$ is not a proper vector space for FDE because all of its elements are N-tuples instead of functions. In contrast, the L^2 space is suitable for FDE because all of its elements are functions. Formally, theL^2 space is a measurable space defined in Eq. (B.1).[21]

$$L^2(\mathbb{R}) = \left\{ f : \mathbb{R} \to \mathbb{C} \mid \int_{\mathbb{R}} |f(t)|^2 dt < \infty \right\}. \tag{B.1}$$

In the L^2 space, every element has two principal features: norm and support. The norm of a function is a measure of the energy or size of the function, while the support is the interval for which the function is nonzero. For any $f(t) \in L^2(\mathbb{R})$, the norm is defined in B.2 and the support is defined in Eq. (B.3).[21]

$$\|f(t)\| = \left(\int_{\mathbb{R}} |f(t)|^2 dt \right)^{1/2}, \tag{B.2}$$

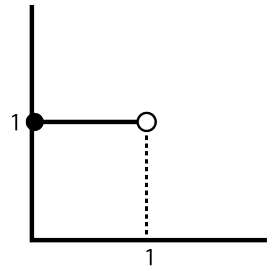


Figure B.1: The box function, $\sqcap$.

$$\operatorname{supp}\left(f\left(t\right)\right)=\left\{t\in\mathbb{R}\mid f\left(t\right)\neq 0\right\}. \tag{B.3}$$

In particular, the support of elements in L^2 is important when using wavelets for density estimation.

One of the simplest elements of L^2 vector space is the box function, denoted $\sqcap\left(t\right)$, which is defined in Eq. (B.4) and shown inFigure B.1. From its definition, we can easily determine the support: $\operatorname{supp}(\sqcap\left(t\right))=[0,1]$. Even though the box function is a simple function, it is very useful when performing density estimation with wavelets.

$$\sqcap\left(t\right)=\begin{cases}1, & 0\leq t<1\\ 0, & otherwise\end{cases}. \tag{B.4}$$

An example of a function that is not an element of L^2 is e^t; simply because $\int_{\mathbb{R}}\left|e^t\right|^2dt=\infty$.

B.2 Inner Products and Orthonormal Bases

Just like the inner product of any two elements in $\mathbb{R}^N$ can be calculated, the inner product between any two functions in L^2 can be calculated. Given $f\left(t\right),g\left(t\right)\in\mathrm{L}^2\left(\mathbb{R}\right)$, the inner product is defined by Eq. (B.5).[21]

$$\left\langle f\left(t\right),g\left(t\right)\right\rangle=\int_{\mathbb{R}}f\left(t\right)\overline{g\left(t\right)}dt. \tag{B.5}$$

The inner product is useful is because if $f\left(t\right)$ and $g\left(t\right)$ are orthogonal, $\left\langle f\left(t\right),g\left(t\right)\right\rangle=0$. Orthogonality of functions is necessary when using wavelets for orthogonal series density estimation. Additionally, the inner product allows for the relationship in Eq. (B.6) to be created from the norm of a function in L^2.

$$\left\|f\left(t\right)\right\|=\left\langle f\left(t\right),f\left(t\right)\right\rangle. \tag{B.6}$$

FDE depends heavily on finding an orthonormal bases. If W is a subset of $\mathrm{L}^2(\mathbb{R})$ and $\{e_k(t)\}_{k\in\mathbb{Z}}$ is the basis for W, then the basis is orthonormal only if each component of the basis is orthogonal to every other component, as in Eq. (B.7).[21] One there is an orthonormal basis for W, any $f\left(t\right)\in W$ can be written as a linear combination of the basis as in Eq. (B.8). In order to determine the coefficients of the linear combination, the inner product between $f(t)$ and every component of the basis, $e_j\left(t\right)\in\left\{e_k\left(t\right)\right\}_{k\in\mathbb{Z}}$, must be calculated, shown in Eq. (B.9). However, since the basis is an orthonormal basis, every inner product will be zero except for when $k=j$, therefore the coefficients can be calculated by using Eq. (B.10).

$$\left\langle e_j\left(t\right),e_k\left(t\right)\right\rangle=\delta_{j,k}=\begin{cases}1, & j=k\\ 0, & j\neq k\end{cases}\quad\text{for all } j,k. \tag{B.7}$$

$$f\left(t\right)=\sum_{k\in\mathbb{Z}}\alpha_k e_k\left(t\right). \tag{B.8}$$

$$\begin{aligned}\langle f(t), e_j(t)\rangle &= \left\langle \sum_{k\in\mathbb{Z}} \alpha_k e_k(t), e_j(t) \right\rangle \\ &= \sum_{k\in\mathbb{Z}} \alpha_k \langle e_k(t), e_j(t)\rangle \\ &= \sum_{k\in\mathbb{Z}} \alpha_k \int_{\mathbb{R}} e_k(t)\,\overline{e_j(t)}dt. \end{aligned} \tag{B.9}$$

$$\alpha_j = \langle f(t), e_j(t)\rangle = \int_{\mathbb{R}} f(t)\,\overline{e_j(t)}dt. \tag{B.10}$$

B.3 Gaussian Measure

Consider all functions of real numbers equipped with the Gaussian measure, Eq. (B.11), where the Gaussian distribution is defined as Eq. (B.12) and dx is the Lebesgue measure. Since $\mathrm{L}^2(\mathbb{R})$ is measurable, the Gaussian measure can be applied to the space, resulting in the Gaussian L^2 measure space, shown in Eq. (B.13).[17] In this measure space, the inner product is defined in Eq. (B.14). If ξ is the standard Gaussian random variable distributed $N(0,1)$, the inner product of functions f and g can be written as expected values, as in Eq. (B.15).

$$\mu(dx) = \rho(x)\,dx, \tag{B.11}$$

$$\rho(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2} \tag{B.12}$$

$$\mathrm{L}^2(\mathbb{R},\mu) = \left\{ f(x) : \int f(x)^2\,\mu(dx) < \infty \right\}. \tag{B.13}$$

$$\langle f, g\rangle = \int f(x)\,g(x)\,\mu(dx) = \int f(x)\,g(x)\,\rho(x)\,dx. \tag{B.14}$$

$$\langle f, g\rangle = E\left[f(\xi)\,g(\xi)\right]. \tag{B.15}$$

APPENDIX C. INTEGRATION MEASURES

C.1 Riemann Integration

The most basic approximation of integrals is Riemann integration, which uses simple approximations for the area under a curve If $f(x)$ is a real-valued function defined on $[a,b]$, then these approximations are the Riemann sums, denoted by Eq. (C.1), which represents the area of a rectangle with a specified width $x_{i+1} - x$ and height $f(t_i)$ when $x_0, ..., x_n$are the partitions of the domain into smaller intervals and t_i is some point in the partition at which the height of f is determined.

$$\sum_{i=0}^{n-1} f(t_i)(x_{i+1} - x_i) \tag{C.1}$$

The Riemann integration is approximated by the limit of Riemann sums as the number of partitions, n, gets larger, or the size of the partitions, $x_{i+1} - x$, becomes smaller. If this limit exists, the function is Riemann-integrable.[15] However, many functions are not Riemann-integrable since Riemann sums can only be used on functions defined on a compact interval that is bounded and continuous almost everywhere. An example of a function that is Riemann-integrable is $f : [0,1] \to \mathbb{R}$ such that $f(x) = 1$ for every point on the domain.

Therefore every Riemann sum on the domain will have a value of 1, and so the limit converges and the function is Riemann-integrable. An example of a function that is not Riemann-integrable is the indicator function of rational numbers, I_Q, on the domain $[0,1]$ such that $I_Q$is 1 if the input is a rational number and 0 if the input is an irrational number. This function's limit of Riemann sums does not converge, so it does not have a Riemann integral. Most functions that are not Riemann-integrable are Lebesgue-integrable, which extends the Riemann integral to a larger set of functions.[26]

C.2 Lebesgue Measure

The Lebesgue measure and Lebesgue integration extends the set of functions that are Riemann-integrable and extends the domain on which functions can be defined. The Lebesgue measure extends the idea of length and area to arbitrary subsets of $\mathbb{R}$ and given an open set $S \equiv \sum_k (a_k, b_k)$ containing disjoint intervals or a closed set $S' \equiv [a,b] - \sum_k (a_k, b_k)$ the Lebesgue measure is defined in Eq. (C.2).[31]

$$\begin{aligned} \mu_L(S) &\equiv \sum_k (b_k - a_k) \\ \mu_L(S') &\equiv (b-a) - \sum_k (b_k - a_k) \end{aligned} \tag{C.2}$$

The Lebesgue measure, denoted $\mu(E)$ for a set E of real numbers, has the following properties:

1. Extends length such that for every interval I, $\mu(I) = l(I)$ where l is the length of interval I
2. Monotonic such that if $A \subset B \subset \mathbb{R}$, then $0 \leq \mu(A) \leq \mu(B) \leq \infty$
3. Translation invariant such that $\forall A \subset \mathbb{R}. \forall x_0 \in \mathbb{R}. A + x_0 := \{x + x_0 : x \in A\} \Rightarrow \mu(A + x_0) = \mu(A)$
4. Countably additive such that if A and B are disjoint subsets of $\mathbb{R}$, then $\mu(A \cup B) = \mu(A) + \mu(B)$. This can be generalized to if $\{A_i\}$ is a sequence of disjoint sets, then $\mu(\bigcup_{i=1}^{\infty} A_i) = \sum_{i=1}^{\infty} \mu(A_i)$

Lebesgue measurable subsets of $\mathbb{R}$ are subsets that have properties (1)-(4) and that have $\mu : \mathcal{M} \rightarrow [0, \infty]$.[18] The Lebesgue measure is particularly used to define Lebesgue integration, which is defined by the integral in Eq. (C.3) for measurable real-valued functions f defined on E.

$$\int_E f d\mu = \int_E f(x) \mu(dx) \tag{C.3}$$

Using the Lebesgue measure, we are able to define the indicator function that was not Riemann-integrable as an integral in Eq. (C.4), since Q is a countable set. Since Q is a measurable set, we are able to define the indicator function using the Lebesgue measure, even though it was not Riemann-integrable. The integral can be seen in Eq. (C.4), and therefore I_Q is Lebesgue-integrable.

$$\int_{[0,1]} I_Q d\mu = \mu(Q \cap [0,1]) = 0 \tag{C.4}$$

Gerald Folland once described the difference between Riemann integrals and Lebesgue integrals of the function f defined on domain $[a,b]$ as "comput[ing] the Riemann integral of f, one partitions the domain $[a,b]$ into subintervals", but for the Lebesgue integral, "one is in effect partitioning the range of f". This difference can also be depicted in Figure C.1.

Density estimations in finite space, typically $\mathbb{R}^d$where d is the dimension of the space, are typically defined with respect to the Lebesgue measure.[32]

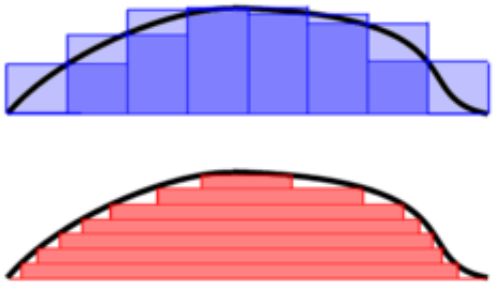


Figure C.1: Depiction of Riemann Integral (top) and Lebesgue Integral (bottom) [32]

C.3 Wiener Measure

When integration is expanded to functional space, the Lebesgue measure no longer works for integration and the Wiener measure must now be used. The Wiener measure is the corresponding measure to the Wiener space, which is the set of all continuous paths $\omega : [0\infty) \to \mathbb{R}$ satisfying $\omega(0) = 0$.[4] The Wiener measure is also the unique probability measure, μ_W, supported on the space $C[0,1]$ of continuous real-valued functions x on the interval $[0,1]$. If $0 < t_1 < ... < t_n \leq 1$ are arbitrary sample points from the domain $[0,1]$, $A_1, .., A_n$are Borel sets, the smallest σ-algebra containing all of the open sets on an interval , on $\mathbb{R}$, $C(t_1, ..., t_n; A_1, ..., A_n)$ is the set of functions $x \in C[0,1]$ for which $x(t_k) \in A_k$for $k = 1, ..., n$, then the Wiener measure is defined in Eq. (C.5) , where $p(x,t)$ is the Gaussian distribution with mean of zero and variance of t.[24]

$$\mu_W(C((t_1, \ldots, t_n;\, A_1, \ldots, A_n)) = \int_{A_1} p(t_1, x_1)dx_1 \int_{A_2} p(t_2 - t_1, x_2 - x_1)dx_2 \ldots \int_{A_n} p(t_n - t_{n-1}, x_n - x_{n-1})dx_n \quad \text{(C.5)}$$

If F is a functional defined on C that is Wiener measurable, then the Wiener integral is defined in Eq. (C.6).[25] Although this integral appears to act the same way as the Lebesgue integral, it is much more difficult to compute, even for simple functions.

$$\int_C F(x)d\mu_W(x) \quad \text{(C.6)}$$