Improvement of Epsilon-Complexity Estimation and an Application to Seizure Prediction

Nathanael Aff

Version of July 2, 2017

Contents

Chapter 1

Introduction

Many natural phenomena produce time series with complex statistical properties and with underlying dynamics that are not well understood. For example, electroencephalogram(EEG) record the electrical potential generated by the synchronized firing of neurons. EEG signals are non-stationary and exhibit varying functional characteristics over short time frames. The $\varepsilon-$complexity coefficients as defined by Darkhovsky and Piryatinska provide a model free way of characterizing a function or time series [4]. In particular, the complexity of a continuous function is quantified by estimating the amount of information needed to approximate the function within some error.

The $\varepsilon-$complexity coefficients are computed by iteratively approximating a function using successively fewer samples of the function. The first implementation of the algorithm for estimating the $\varepsilon-$complexity coefficients used piecewise polynomials. We implement the algorithm with three related approximation methods – basis splines, cubic splines, and an interpolating subdivision method termed the lifting scheme. For a suite of stochastic processes, we compare the accuracy of the approximation methods and the performance of each method in a classification task. An initial hypothesis was that an enlarged set of approximation methods would improve the behavior of the $\varepsilon-$complexity coefficients. We find that both the lifting scheme and cubic splines produced slightly higher average approximation errors than basis splines but the two methods performed better than basis splines in the classification task.

Darkhovsky and Piryatinska have proved that for a Hölder class of functions the theoretical ε-complexity coefficients capture a linear relationship between the log of the approximation error and the log of the fraction of points used in the approximation. They have conjectured that a generating mechanism leads to constant mean complexity coefficients. We use the several simulations with known Hölder exponents to first check the performance of our implementation and then to test if the conjecture holds. The Hölder exponent of a function is closely related to the fractal dimension of the graph of a function. For the same set of simulations we compare the complexity coefficients to an estimator of fractal dimension and find the slope coefficient B closely related to the fractal dimension estimator

In the final chapter, we use the complexity coefficients along with a set of spectral features to predict seizures in epileptic mice. The characteristics of the EEG signal can change rapidly over a short time period. Common classification methods use features averaged over arbitrary windows of time. We test whether segmenting time series based on changes

in the epsilon complexity coefficients improves the prediction of seizures. This dynamic segmentation method is compared to uniform segmentation of trials. The best classification is achieved on a regular partition of the time series, although segmentation on change points of the two complexity coefficients performed about as well. The data contained only a small number of trials and a comparison of segmentation methods on a broader range of data would be needed to reach any conclusions about whether these results generalize to other contexts.

Finally, we have created an `R` language package, `ecomplex`, that implements the ε-complexity algorithm. Several default settings are based on the simulation experiments described in Chapter 3. Methods used for generating simulations, computing EEG features, and the classification algorithm used in Chapter 5 have been also been included in `R` packages to encourage additional work on related topics or replication of the results found here.

Chapter 2

Background

2.1 Epsilon-complexity

The word complexity has been used to describe a diverse set of both mathematical and natural phenomena. The use of the term to describe the complexity of a continuous function agrees with Kolmogorov's definition of the complexity of a discrete sequence. Kolmogorov complexity characterizes the regularity of a sequence or string by the size of the program needed to output that sequence. For example, we can take as our sequence a natural number and let our 'program' or representation of that number in scientific notation. Then we can express 1,000,000 in 2 digits as 1E6 while the prime 7919 could not be similarly compressed [19]. In this case, 1,000,000 is less complex 7919. The approach taken by Darkhovsky and Piryatinska in defining the ε-complexity of a continuous function takes inspiration from Kolmogorov's definition of complexity[4]. The information, roughly speaking, contained in a continuous function is measured by the number of points sampled from the function that is needed to reconstruct that function within some error ε. The algorithm used to compute $\varepsilon-$complexity coefficients makes successive approximations of a function with fewer and fewer sampled points. The two ε-complexity coefficients parametrize the linear relationship between the log of the approximation error and the log of the proportion of points for each successive approximation.

We begin with the formal definition of the $\varepsilon-$complexity of a single function followed by the primary theorem of [4] which shows that $\varepsilon-$complexity characterizes the Hölder condition of a class of functions. The practical computation of $\varepsilon-$complexity takes a slightly different form than that used in the proof of this theorem and we describe this difference and present the algorithm used to compute the $\varepsilon-$complexity coefficients in practice.

Let $x(t)$ to be a continuous function defined on the unit cube $\mathbb{I} \in \mathbb{R}^k$. Let $\|\cdot\|$ be a norm on the function and we can assume $\|x(t)\| = 1$, that is the function has been normalized by taking $x(t)/\|x(t)\|$. Let $\mathbb{Z}_h$ be a k dimensional grid with spacing h. Then the set of samples at spacing h is $\mathbb{I} \cap Z_h^k$. Given some family of approximation methods $\mathcal{F}$ we define the minimal approximation error over the family $\mathcal{F}$ for grid spacing h

$$\delta(h) = \|x(t) - \hat{x}(t)\|,$$

where $\hat{x}(t) \in \mathcal{F}$. Now we fix the minimum *error* ε and find the spacing h such that some approximation method $\hat{x}(t)$ approximates $x(t)$ within ε. Here h is a function of the fixed

error ε and we define

$$h^*(\varepsilon) = \begin{cases} \inf\{h \leq 1 : \delta(h) > \varepsilon\} & \text{if}\{h : \delta(h) > \varepsilon\} > 0 \\ 1, & \text{if no such } \hat{x}(t) \in \mathcal{F}\text{exists.} \end{cases}$$

The function $h^*(\varepsilon)$ is the minimum grid spacing h, or sample density, needed to approximate $x(t)$ within ε. The quantity $\frac{1}{h^*(\varepsilon)}$ counts the number of samples needed to reconstruct $x(t)$ within a given error ε.

Definition 2.1.1. Epsilon-complexity. Let $\mathcal{F}$ be a family of approximation methods and $\|\cdot\|$ some norm. Then

$$S(\varepsilon, \mathcal{F}, \|\cdot\|) \equiv S(\varepsilon) \stackrel{\text{def}}{=} \log \frac{1}{h^*(\varepsilon)} \tag{2.1.1}$$

is the $(\varepsilon, \mathcal{F}, \|\cdot\|)-$complexity, or simply, $\varepsilon-$complexity of a continuous function $x(t)$.

The previous exposition refers to an individual function $x(t)$. However, the main primary theorem of [4] is proved for a given *class* of functions, in particular for any subset of a Hölder class of functions.

Definition 2.1.2. Hölder Continuity. A function $x(t)$ is said to be Hölder continuous if there exists non-negative constants, C, α such that

$$|x(t) - x(s)| \leq C \|x - s\|^{\alpha}$$

for all t and s. We refer to this as the Hölder condition with Hölder exponent α or say a function is Hölder α.

We now state as a theorem the corollary of the primary theorem of [4].

Theorem 2.1.3. The $\varepsilon-$complexity of any bounded subset U of the Hölder class of functions, denoted $S_{cl}^U(\varepsilon, H)$, where H indexes a particular Hölder class is given by

$$S_{cl}^U(\varepsilon, H) = \mathcal{A} + \mathcal{B} \log \varepsilon. \tag{2.1.2}$$

The subscript cl in the preceding definition denotes the class $\varepsilon-$complexity as opposed to that of an individual function. Given this statement about a Hölder class of functions, for some individual function in Hölder class X_H it should hold that the ε-complexity of the individual function is less than or equal to that of the class, that is,

$$S_x(\varepsilon, H) \leq S_{cl}^H(\varepsilon, H).$$

Later in this chapter we use some functions with known Hölder exponents and in the following chapter we use a number of simulations to test the relationship between the complexity coefficients and the Hölder exponent of these functions.

In the definition 2.1.1 the $\varepsilon-$complexity of a class of functions is expressed as a function of the error ε. In practice, we fix a finite number of spacings h and find the minimum error over all approximation methods in some family $\mathcal{F}$. As defined above in 2.1.1 $S(\varepsilon)$ is a log

of the fraction of points used in approximating the function. Here, we express the error as a function of h. Letting $\mathbb{S}_h = \frac{1}{h}$ and substituting $\mathbb{S}_h$ for $\frac{1}{h^*(\varepsilon)}$ gives us

$$\log \varepsilon = A + B\mathbb{S}.$$

This is the relation we use for the estimating the complexity coefficients A, B.

In practical applications, a given function $x(t)$ is of some finite length N and we assume the function has been sampled on some regular grid $\mathbb{Z}_h$. The given initial sampling grid determined by h_1 is the densest resolution of the function and other grid spacings $h_2, h_3, ..., h_d$ are also taken at regular intervals. For each h_k the function $x(t)$ is approximated with some family of methods $\mathcal{F}$. For a given h the minimum mean-squared error is then added to the set of errors, $\varepsilon_{\mathcal{F},h}$. Finally, setting $S_h = \frac{1}{h}$, a least squares linear model is fit to the set of errors $\{\varepsilon_{\mathcal{F},h}\}$ regressed on the proportion of function values $\{S_h\}$:

$$\log \varepsilon_{\mathcal{F},h} \sim A + B \log S_h.$$

The parameters of the linear model A, B are the $\varepsilon-$complexity coefficients or simply the complexity coefficients.

Algorithm 1: $\varepsilon-$complexity

Input : X a regularly sampled time series of length N.
Input : $\mathcal{F}$ a set of approximation methods f.
Input : $\mathcal{H}$ the set of spacings h.
Output: The complexity coefficients A, B.

foreach h *in* H **do**
  foreach f *in* $\mathcal{F}$ **do**
    Compute the approximation error
    $\varepsilon_{h,f} \leftarrow \frac{1}{N} \|f_h - X_h\|^2$.
  end
  Find minimium error over all f
  epsilons$_h \leftarrow \min \varepsilon_{h,f}$.
end
Fit a least squares linear model
$A, B \leftarrow \text{lm}(\{\text{ epsilons}_h\ \} \sim \{\mathbb{S}_h\}.)$

2.2 Time series

The theory of $\epsilon-$complexity as developed by in [4] for continuous and possibly vector-valued functions denoted $x(t)$. For the remainder of this thesis of our discussion will be restricted primarily to functions on $\mathbb{R}^1$, or univariate stochastic processes which we also refer to as time series.

A time series is a stochastic process, or set of random variables, $X(t)$ or X_t, parametrized by time, $\{X_t : t \in T = \{1, 2, 3, ...\}\}$. We will denote realizations of a time series with the

lower case x_t or $x(t)$. We also use $x(t)$ to refer to a continuous function or samples from the continuous function and we will not always distinguish between samples from a continuous function and observations of a random process.

We will begin by defining several properties of time series We also introduce a number of time series model that are used in the following chapters to generate time series on which we test the behavior of our implementation of the ε-complexity algorithm. Several of models we introduce have known properties such as Hölder class and fractal dimension and we test if and how the complexity coefficients measure these properties in Chapter 4.

The ARMA model is a simple and widely used parametric time series model. It is a linear model that combines autoregressive (AR) and moving average (MA) models. The AR component expresses an observation at time t as a linear combination of preceding observations plus a random error component:

$$X_t = b_1 X_{t-1} + \cdots + b_p X_{t-p} + \varepsilon_t \tag{2.2.1}$$

The number of previous observations p is the order of the model, denoted $AR(p)$. The random error, ε_t, is assumed to be independent, identically distributed (IID) observations from white noise a standard normal distribution with mean 0, variance σ^2 and covariance $Cov(X_i, X_j) = 0, i \neq j$. The latter is called white noise and the errors are said to be drawn from a white noise process $\{\varepsilon_{t_i}, \varepsilon_{t_j}\} \sim WN(0, \sigma^2)$.

The moving average(MA) process of order q models the observation x_t as a linear combination of q samples $\{\varepsilon_t\} \sim WN(0, \sigma^2)$

$$X_t = \varepsilon_t + a_1 \varepsilon_{t-1} + \cdots + a_q \varepsilon_{t-q}. \tag{2.2.2}$$

Depending on the parameters p, q of an ARMA(p, q) process, the resulting time series may or may not be *stationary*. A strictly stationary time series is one whose distribution independent of time t. We will be using the term to indicate a *weakly stationary* time series, or a time series whose first two moments do not depend on time.

Definition 2.2.1. Weak stationarity A *time series* with finite variance and a mean μ_t that does not depend on time is called *weakly stationary*:

$$\begin{aligned} &E(X_t^2) < \infty \\ &E(X_t) = \mu \quad \text{is a constant independent of t} \\ &\text{Cov}(X_t, X_{t+h}) \quad \text{is independenta of } t \text{ for each } h \quad . \end{aligned}$$

Where it exists, let $E(X_t) = \mu_t$ be the mean function of a time series.

Definition 2.2.2. Covariance function. The *autocovariance or covariance* of a time series with mean μ is then defined

$$\sigma(h) = \text{Cov}(X_t, X_{t+h}) = E[(X_t - \mu_t)(X_{t+h} - \mu_{t+h})]$$

In the case of a zero mean function this simplifies

$$\sigma(h) = E[(X_t)(X_{t+h}]).$$

The *autocorrelation* function of X_t is

$$\rho(h) \equiv \sigma(h)/\sigma(0) \equiv \text{Corr}(X_{t+h}, X_t), \qquad h \in \{1, 2, 3, ...\}.$$

For a stationary, mean zero stochastic process with $\sigma^2 = 1$ the autocorrelation and autocovariance function are equivalent. Some stochastic processes, like the Cauchy process we introduce later, can be specified by entirely by their autocorrelation function.

If a stationary process has an absolutely summable autocovariance function

$$\sigma(h), \sum_{-\infty}^{\infty} |\sigma(h)| < \infty$$

then the *spectral density function* of the process is the Fourier transform of the $\sigma(h)$

$$f(\omega) = \sum_{-\infty}^{\infty} e^{2\pi i \omega h} f(\omega) d\omega \quad -1/2 \leq \omega \leq 1/2. \tag{2.2.3}$$

Conversely, $\sigma(h)$ can be represented as the inverse transform of the spectral density function. In either case, the two representations carry the same information about the process.

A stationary ARMA process has an autocorrelation function $\rho(h)$ that decays at an exponential rate as $|h| \to \infty$ [13]. A slowly decaying autocorrelation function is characteristic of a non-stationary process with long-range dependence. We include three functions in our set of simulations that can have long-range dependence – fractional Brownian motion, the Cauchy process and the fractionally integrated ARMA or FARIMA process. The first two of these we discuss in the next section. The ARIMA process is model whose integral differences d are a ARMA(p,q) processes, denoted ARIMA(p, d, q). The FARIMA model is a modifies ARIMA process allowing for fractional values d[13].

2.3 Fractal dimension

In the following chapter, we test the relationship between the Hölder continuity of a function, estimators of its fractional or fractal dimension and the $\varepsilon-$complexity coefficients. Here we define fractal dimension and introduce several functions or stochastic processes with known fractal dimension.

Although fractal dimension is a general measure on sets we will be considering only those sets formed by the graph of a function f or a sample function drawn from a stochastic process. Fractal dimension coincides with the usual sense of dimension for smooth manifolds such as the interval $[0, 1]$. Since we consider only graphs $\Gamma : \{(x, f(x))\}$ in $\mathbb{R}^2$ the fractal dimension of these graphs should take on values between $[1, 2)$.

The Hausdorff dimension of a set E, roughly speaking, measures the size of covers of that set as the diameter of covering elements $\{U_i\}$ goes to zero. We define the diameter of a set U denoted $|U|$ as $\sup\{|x - y| : x, y \in U\}$, the greatest distance between any two points in the set. A set collection $\{U_i\}$ is a cover of E if $E \subset \bigcup_i^{\infty} U_i$ where we assume $\{U_i\}$ is countable.

Definition 2.3.1. Hausdorff Dimension Let all elements in $\{U_i\}_\delta$ be a countable subset of $\{U_i\}$ with diameter greater than δ such that $\{U_i\}_\delta$ cover some set $E \subset \mathbb{R}^n$. The the Hausdorff dimension $\mathcal{H}_\delta^\alpha$ for any $\delta > 0$ is defined

$$\mathcal{H}_\delta^\alpha = \inf \left[\sum_i |U_i|^\alpha : \{U_i\} \quad \text{is a cover of} \quad E \right].$$

Then

$$\mathcal{H}_{\alpha(E)} = \lim_{\delta \to 0} H_{\delta}^{\alpha(E)}$$

For $0 < \alpha < 1$, $\mathcal{H}^{\alpha}$ is either 0 or ∞ and there exists a critical point α at which $\mathcal{H}^{\alpha}$ switches from 0 to ∞. This point is the Hausdorff dimension of the set which we denote $\dim_H E$ [5].

The Hausdorff dimension of many sets is difficult to calculate and there are a number of related measures. The box-counting dimension is one these and, for sets in $\mathbb{R}^2$, it counts the number of squares of length δ required to cover a set. It therefore provides an upper bound for the Hausdorff dimension of a graph. The box-counting dimension is defined as

$$\dim_B E = \lim_{\delta \to 0} \frac{\log N_{\delta}(F)}{-\log \delta}$$

where N_{δ} is the smallest number of sets that cover E of side length δ. In what follows we may refer to the fractal dimension of a graph of a function or of the function itself. In cases where we refer to estimators of fractal dimension we generally will not specify a specific measure.

A number of practical estimators of fractal dimension D exist. In Chapters 3 and 5 we use a fractal dimension estimator based on the variogram of a stochastic process which we define here.

Definition 2.3.2. Variogram The increments of a stochastic process are defined as $\{X_t - X_{t+h}\}$. Let X_t is a zero mean stochastic process with stationary increments. The variogram of X_t is defined

$$\gamma^2(h) = \frac{1}{2}\mathbb{E}\left(X_t - X_{t+h}\right)^2.$$

The variogram relates to the the covariance function $\sigma(h)$ via

$$\gamma^2(h) = \sigma(0) - \sigma(h)$$

[7]. Let X_t be a Gaussian process with variogram satisfying

$$\gamma(h) \sim |ch|^{\alpha}, \tag{2.3.1}$$

then $dim_H \Gamma(X_t)$ is almost surely $2 - \alpha$ [7] [11]. The estimator of fractal dimension in the next chapter uses is based on the relation 2.3.1 and estimates fractal dimension by taking log-log regression of the the empirical variogram $\hat{V}(h)$ against increments h. This method is recommended by Gneiting in [7].

Brownian motion and fractional Brownian motion(FBM) are both Gaussian processes satisfying these conditions. In particular, both processes can be defined in terms of their increment process. Brownian motion has normally, independently distributed increments and fractal dimension 1 1/2. FBM has dependent increments with variogram

$$E[(X_t - X_{t+h})^2] = h^{2\alpha}.$$

and the case when $\alpha(1/2)$ corresponds to Brownian motion. The parameter α corresponds in this case to the fractal dimension of FBM [5]. When $\alpha > 1/2$ FBM is a stochastic process with long-range dependence and α corresponds to the Hurst coefficient of the function.

Orey remarks that for a Gaussian process X_t the parameter α also corresponds to the Hölder exponent of $\Gamma(X_t)$ [11]. This means we can vary the parameter α to determine both the fractal dimension and Hölder exponent of FBM and we use this property in the following chapter to test the behavior of the complexity coefficients as the parameter α is varied.

2.4 Hölder Continuous Functions

In the following section we present results from a number of simulations that explore the relationship between the fractal dimension, Hölder class of functions and the estimated $\varepsilon-$complexity coefficients of those functions. In addition to FBM, we use two other stochastic processes and one deterministic function for which parameter of the models control the Hölder continuity or fractional dimension of the models or both.

The first of these is the *Weierstrass* function, a well-known example of a continuous but nowhere differentiable function. The Weierstrass can be written in several forms. Here we use the form that isolates the parameter α corresponding to the Hölder exponent of the function.

Definition 2.4.1. Weierstrass function. The *Weierstrass* function defined

$$W_\alpha(x) \quad = \sum_{n=0}^{\infty} b^{-n\alpha} \cos(b^n \pi x) \tag{2.4.1}$$

is a continuous periodic nowhere differentiable the function with that is Hölder α:

$$|W_\alpha(x) - W_\alpha(y)| \leq C\,|x-y|^\alpha\,.$$

The parameter α determines both the fractal and Hausdorff dimension of the Weierstrass function. The upper bound for the Hausdorff dimension of W_α is $2-\alpha$ but it has not been proven to equal $2-\alpha$ [8]. On the other hand, the Weierstrass function has the box-counting dimension equal to $2-\alpha$, providing the upper bound for the Hausdorff dimension [5]. In fact, a more general relation between the Hölder condition and fractal dimension holds for a continuous function on $\mathbb{R}$. The Hölder condition provides bounds the box-counting dimension:

Proposition 2.4.2. Let $f : [0,1] \to \mathbb{R}$ be a continuous function and

$$|f(t) - f(t+h)| \leq ch^\alpha \quad (c > 0, 0 \leq \alpha \leq 1)$$

then $dim_B f \leq 2-\alpha$[5].

A variant of the Weierstrass function is the random-phase Weierstrass function. It has the same property of being Hölder continuous with Hölder exponent α but has more complex spectral characteristics[8]. The random phase function has the same equation as the Weierstrass function with an added phase, ϕ drawn from a uniform distribution:

$$W_\alpha(x) = b^{-n\alpha}\cos(b^n\pi x + \phi_n), \quad \phi_n \sim \quad \text{unif}(0,1).$$

The Hausdorff dimension of the random phase variant Weierstrass function has been shown to equal $2-\alpha$ [8]. See Figure 4.3 in the following chapter for illustrations of both Weierstrass functions.

In the case of fractional Brownian motion the parameter α controls both the fractal dimension and the long range dependence of the function. The Cauchy process, on the other hand, has two parameters which separately control the long-range dependence of the function and its local behavior or fractal dimension [6]. The Cauchy is a stationary Gaussian process determined by its autocorrelation function:

$$\rho(t) = (1 - |ct|^{\alpha})^{-\beta/\alpha}, \quad \alpha \in (0, 2]; \beta > 0.$$

The Cauchy process parameters determine the fractal dimension D of the process

$$D = n + 1 - \frac{\alpha}{2},$$

and if $\beta \in (0, 1)$ the process has long-range dependence with Hurst coefficient

$$H = 1 - \frac{\beta}{2}.$$

2.5 Approximation methods

In calculating the complexity coefficients some set number of approximation methods $\mathcal{F}$ are used. The initial implementation used piecewise polynomials of up to degree 5. In order to test the effects of other and possibly more accurate approximation methods, we added several additional methods to our implementation of the $\varepsilon-$complexity algorithm. In chapter 4 we test our implementation and compare their performance. Here we briefly describe the implemented methods.

Splines are piecewise polynomials joined on some set of knots, or intersection points, where the additional constraint that the spline function of order n has continuous derivatives of order $n-1$ at each knot. Three spline methods were used in our implementation, cubic splines, basis splines of order up to 5 and the lifting scheme, a spline-based iterative interpolation.

Basis or $B-$splines are piecewise polynomials of order n that form a basis for a linear space which we denote $S_{\triangle,n}$, where n is the order of the polynomials and $\triangle$ defines the knot sequence

$$\triangle = t_0 < t_1 < \cdots < t_m.$$

We denote the basis spline $B_{i,n}$ of order n on segment i and $\text{span}\{B_{i,n}\} = S_{\triangle,n}$. The space $S_{\triangle,n}$ contains C^{n-1} functions that are polynomials on the segments $[\triangle_i, \triangle_{i+1}]$ [18].

The basis splines $B_{i,n}$ can be defined recursively, with $B_{i,0}$ a step function on the segment i.

$$B_{i,0}(x) = \begin{cases} 1 & \text{if } t_i \leq x \leq t_{i+1} \\ 0 & \text{otherwise.} \end{cases}$$

$$B_{i,n+1}(x) = \alpha_{i,n+1}(x) B_{i,n}(x) + [1 - \alpha_{i+1,n+1}(x)] B_{i+1,n}(x),$$

where

$$\alpha_{i,j}(x) = \begin{cases} \frac{x-t_i}{t_{i+n}-t_i} & t_{i+n} \neq t_i \\ 0 & \text{otherwise.} \end{cases}$$

A spline $s(x)$ function of order n is then a linear combination of the basis functions

$$s(x) = \sum_{i=-n+1}^{m} \beta_{i,n} B_{i,n}.$$

There are a number of numerical methods for implementing spline approximation. Each value of the spline function $s(x_j) = f_j$ can be determined by

$$s(x_j) = \sum_{i=-n+1}^{m-1} \beta_{i,n} B_{i,n}, j = 1 : m + n - 1.$$

This forms a set of linear equation which is over determined when the set of knots is smaller than the number of points to interpolate. The problem can be solved using a least squares approximation

$$\min_{\beta} \|A\beta - f\|$$

where the entries in the matrix are the spline basis functions evaluated at each interpolation point x_j [3]. This matrix is banded with an n entries in each row for splines of order n. We used the B-spline approximation method as implemented in `R` package `spline` [15][14].

An additional spline method using only cubic splines for which the number of knots was limited to 200. The banded matrix A is tridiagonal, having entries only on its diagonal and two off-diagonals. An LU type decomposition for tridiagonal matrices allows for fast cubic spline interpolation.

For a third method we implemented an interpolating subdivision technique which we will refer to as the lifting scheme after Sweldens, who gives a hands-on description in [17]. To convey the basic approach we consider a very simple example of repeated interpolation. Given a set of regularly sampled points $\{s_{0,k}\}$ with $k \in \mathbb{Z}$ points a linear interpolation of the point $x_{j+1,2k+1}$ is the average of the adjacent points $s_{j,k}$ and $s_{j,k+1}$. A recursive formula for averaging and interpolating mid-points at stage $j + 1$ is given by

$$\begin{aligned} s_{j+1,2k} &= s_{j,k} \\ s_{j+1,2k+1} &= 1/2(s_{j,k} + s_{j,k+1}). \end{aligned}$$

For regularly sampled points, the same procedure repeated with an odd $n - order$ polynomial takes the n adjacent points centered the point $x_{j,k}$ to be interpolated. If a single position $s_{j,k}$ is set to 1 and the interpolating subdivision the algorithm is run ad infinitum, the limiting function as $j \to \infty$ is a scaling function. In the case of linear interpolation, for example, the scaling the function is a tent function. For a given order polynomial the scaling function generated by interpolating subdivision on regular intervals, dilates and translates of the function form an orthonormal basis for functions in $L^2(\mathbb{R})$ [17]. While this connects interpolating subdivision to the wide-ranging subject of wavelets, we only mention the connection in passing as it is not needed for later results.

Unlike wavelet based interpolation methods, the lifting scheme can be extended to interpolating irregularly spaced points since regular spacing is not a condition for polynomial interpolation. The same recursive subdivision scheme can be used but in this case, the resulting limiting function does not share the analytic properties of scaling functions. For example, the limiting function does not necessarily form a basis of some function space.

For the $\varepsilon-$complexity algorithm the pattern of interpolation changes based on the grid spacing h at which samples are taken. However, for a set interpolation pattern the polynomial interpolant needs to be computed only once. Neville's algorithm is used to compute a set of n weights for a centered $n-$order polynomial interpolation. To use the simple example of linear interpolation above, the weights to compute $x_{j+1,2k}$ at the mid-point of $s_{j,k}, s_{j,k+1}$ are $(1/2, 1/2)$. That is

$$\langle 1/2, 1/2 \rangle \cdot \langle s_{j,k}, s_{j,k+1} \rangle = x_{j+1,2k}.$$

These weights $(1/2, 1/2)$ can be thought of as a time-domain filter. For our implementation polynomials of orders $1, 3, 5$ were used for interpolation and for each order polynomial three filters were computed for each distinct interpolation pattern. Since these filters needed only to be generated once, the interpolation method is $O(n)$ or linear in the number of inputs.

Chapter 3

Approximation Method Performance

In theory, the $\varepsilon-$complexity of a continuous function is found using the minimum error over a possibly infinite set of approximation methods. To be computationally efficient, this set of approximation methods needs to be finite and in practice, is restricted to some relatively small set of approximation methods. The initial implementation of the $\varepsilon-$complexity algorithm used piece-wise polynomial interpolation. Our implementation extends the number of methods used to three: basis splines(B-splines) of up to order 4, cubic splines and the interpolating subdivision we refer to as the lifting scheme after Sweldens [16].

The goal in including more approximation methods was to test whether drawing on a richer set of functions would improve the performance of the $\varepsilon-$complexity coefficients. In this chapter, we gauge the approximation methods in two ways. First, we measure the average approximation error of each method on a range of functions. Secondly, we test the ability of the complexity coefficients to discriminate between two sets of closely related time series. Based on these experiments, increasing the accuracy of the approximation did not lead to better performance in the discrimination task. In fact, the cubic spline method which was less accurate in terms of approximation errors performed as well or better than the more accurate B-spline method.

A separate concern was the computational efficiency of the approximation methods. Computational efficiency becomes a pressing issue as the size of a data set increases. The EEG time series studied in the next Chapter 5 were moderately sized by modern standards, consisting of several million points per time series. But even at this size computational efficiency became an issue when using the less efficient B-spline algorithm. The B-spline method, although the most accurate approximation method of the three tested, uses matrix operations which are at least quadratic both in space and time efficiency. Based on both computational efficiency and classification performance we have used the cubic spline method as our default for computing the complexity coefficients in our `R` implementation of the algorithm and in our applied work in the following two chapters.

3.1 Simulations Tests

For these experiments, we used six simulations methods comprised of both deterministic and random processes. We have presented all but one of these functions in the background

section. The *logisitc map* is a single parameter nonlinear deterministic discrete difference equation defined

$$x_t = r(x_{t-1})(1 - x_{t-1}).$$

For values of r close to but less than 4 the logistic map exhibits chaotic behavior and the graph of the function changes significantly for small perturbations of the r parameter. The other simulations used were the ARMA and FARIMA processes, the random-phase Weierstrass function, fractional Brownian motion (FBM) and the Cauchy process.

Two sets of models were created with each set having slightly varying initial parameters. For each simulation, the initial parameters perturbed by selecting a parameter uniformly in a window about the initial parameter value. A sample from each simulation group is shown in 3.1.

In addition to the three individual approximation methods – cubic splines, B-splines, and the lifting method – for a fourth method the approximation error was calculated as the minimum error over all methods. Shown in 3.2 is an example of log-log linear fit of the approximation errors against the sample fraction for a single function from each simulation type. For this figure, the lifting method was used. The figure is a simple diagnostic check that the log-log fit was roughly linear and the results of the test were similar to that shown in 3.2 for each approximation method.

Figure 3.3 is a scatterplot of the complexity coefficients generated by each approximation method for the two groups of simulations. Each point represents where an individual function lies in the feature space of the two complexity coefficients A and B. The coefficient values have been normalized to the $[0, 1]$ interval show the plots only show the relative distribution of the functions. Although it is hard to distinguish the denser regions, the plot shows that the complexity coefficients are distributed similarly for each method – the complexity coefficients for a given simulation lie in the same region of the function space for each method. Several functions are also linearly separable in coefficient space – a line could easily be drawn that divides the two groups. For example, the samples of the logistic function are tightly grouped in the upper right and corner and fractional Brownian motion and the ARMA process are also well separated in the lower left-hand corner. The separability of the two groups is quantified in by the classification error in Table 3.2 which we discuss in more detail below.

We finally observe in Figure 3.3 the direction in which, when possible, the functions can be linearly separated. In the following chapter, we compare how the complexity coefficients and fractal dimension vary with the parameters several functions. Based on those tests it is unclear whether the intercept coefficient A adds contributes information that can be used in discriminating two functions that are not found in slope coefficient B. However, in the figure 3.3 for the function types that are readily separable – FBM and the ARMA and logistic simulations – both the slope and intercept coefficient appear to be useful in separating the two sets of simulations. For example, both the FBM and ARMA simulations are well separated along the A-axis. Although how best to interpret each of the coefficients is not clear, both appear useful for separating related functions.

Using the two sets of simulations described above, with 30 samples taken from each parameter group, the approximation error and classification accuracy of each approximation method was tested. Tables 3.1 and 3.2 show the mean approximation error for each method and the overall classification accuracy when using only the two complexity coefficients as

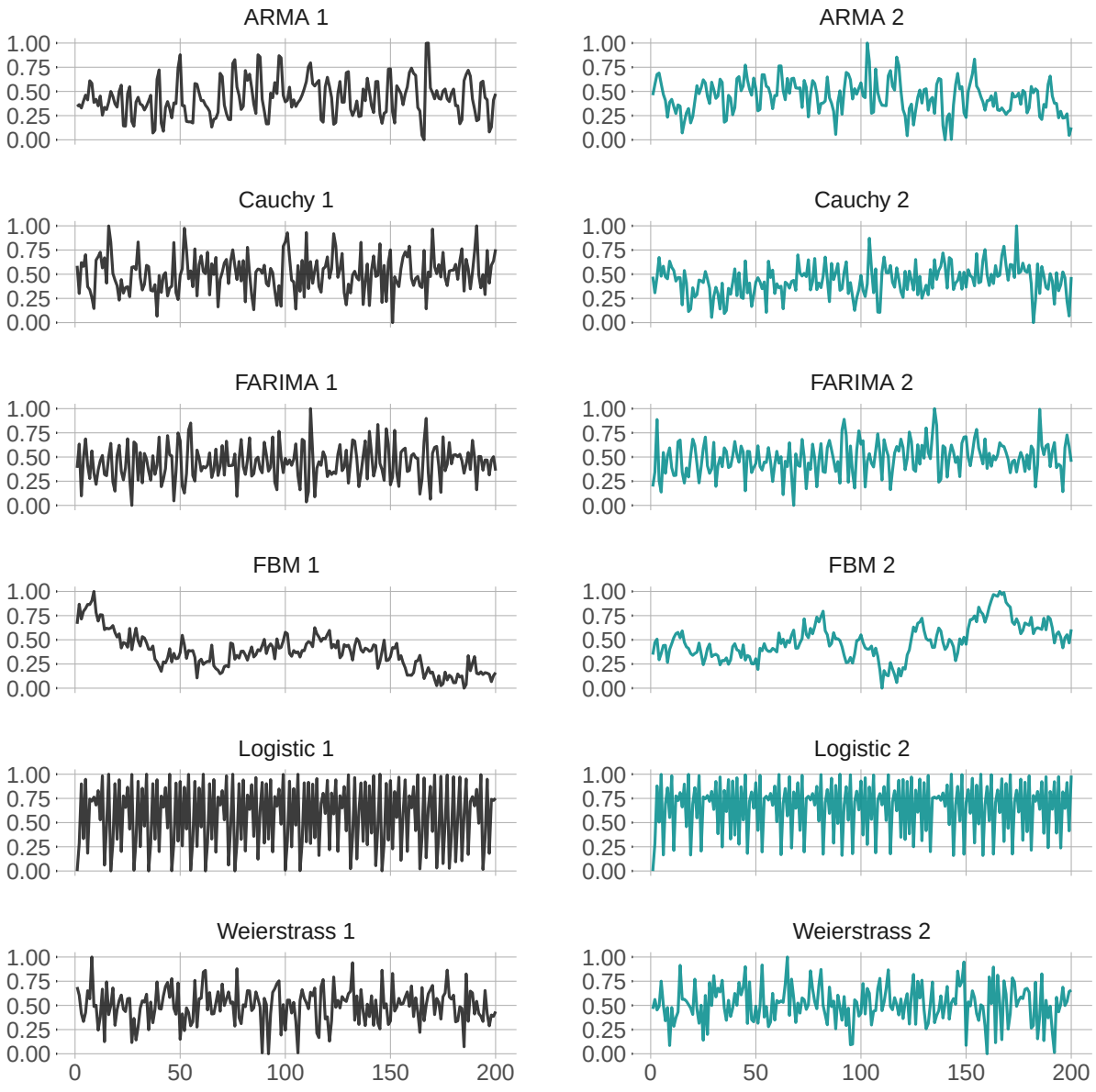

Figure 3.1: Sample functions from each simulation group.

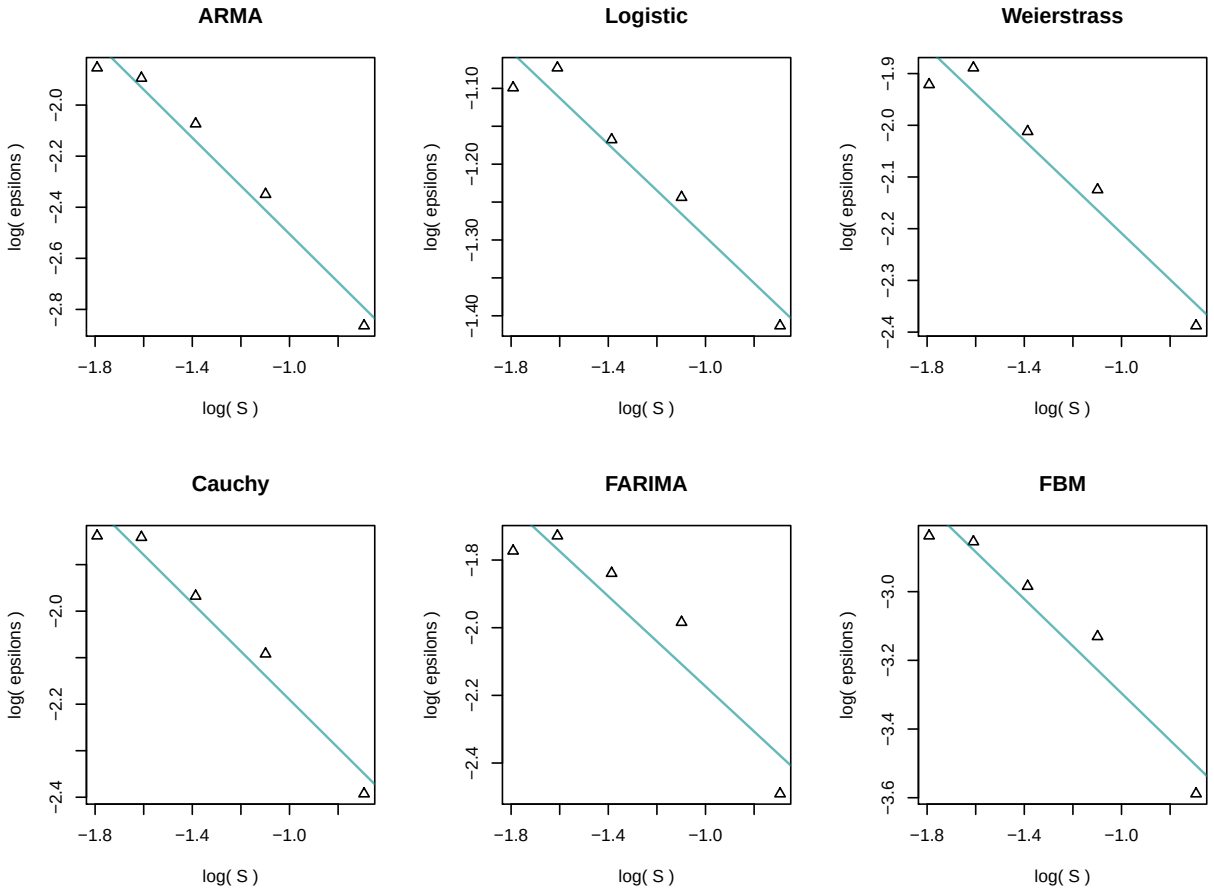

Figure 3.2: Least squares fits used to compute the complexity coefficients using the lifting scheme.

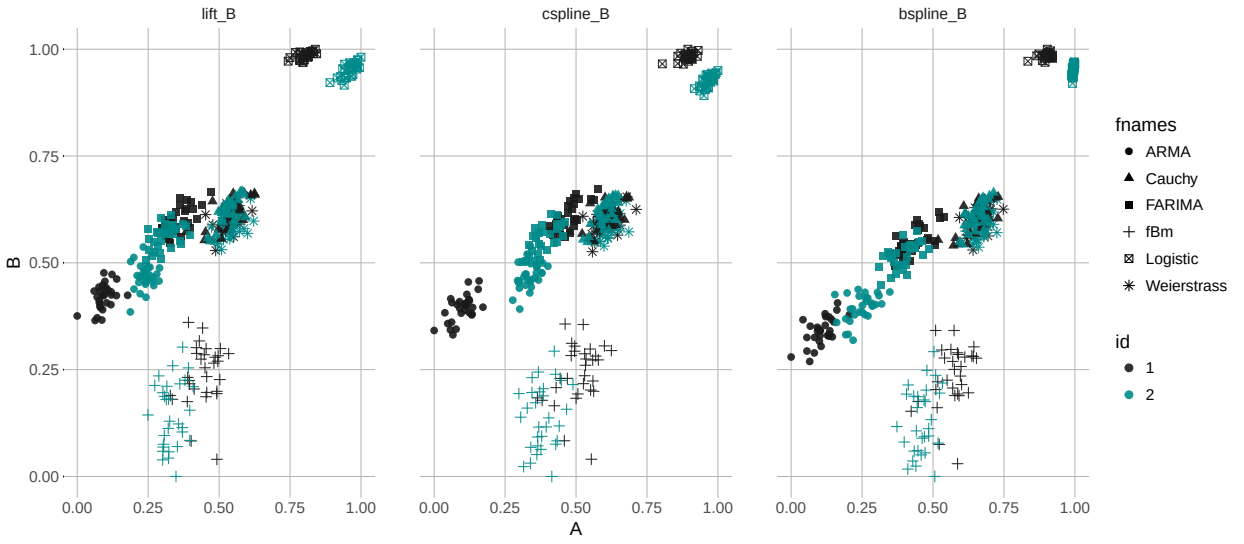

Figure 3.3: Functions from the two simulation groups plotted in the complexity coefficient space.

predictors. The B-spline approximation had the lowest MSE approximation for all functions and the combined method simply reflects the B-spline errors. The lift approximation method also has slightly lower MSE for most functions.

Function	Lift	Cspline	Bspline	Combined
ARMA	0.10	0.11	0.07	0.07
Cauchy	0.09	0.10	0.06	0.06
FARIMA	0.13	0.14	0.08	0.08
FBM	0.03	0.04	0.02	0.02
Logistic	0.30	0.32	0.20	0.20
Weierstrass	0.14	0.15	0.09	0.09

Table 3.1: Mean of the approximation errors of each method summed over each downsampling level. Errors were computed on 30 simulations of each model.

The complexity coefficients as computed by each approximation method were used to classify the two sets of simulations. A random forest classifier was used and the overall out-of-bag classification error is shown in 3.2. No method is dominant for all simulation types, and the cubic spline method performs as well or better than the two approximation methods.

Based only on the previous two tests there is not a clear reason to choose one of the approximation methods over another. Classification performance did not seem affected by the slightly lower approximation accuracy of the lifting and cubic spline methods. However, both the cubic spline and lifting approximation methods are computationally more efficient. Both methods are computationally linear in the number of inputs, or $O(n)$, and both are

Function	Lift	Cspline	Bpsline	Combined
ARMA	0.03	0.00	0.07	0.07
Cauchy	0.65	0.53	0.52	0.53
FARIMA	0.32	0.30	0.32	0.33
FBM	0.20	0.22	0.20	0.22
Logistic	0.00	0.00	0.00	0.02
Weierstrass	0.50	0.43	0.45	0.45

Table 3.2: Classification errors using complexity coefficients A and B for each approximation method using a random forest classifier.

linear or constant in terms of their space complexity – the amount of memory needed to store intermediate computations is a linear or constant function of the size of the inputs. Figure 3.4 show the computational time of each method as a function of the log of the input size.

3.2 Computational Complexity

For our implementation, we scaled the number of basis elements used in the B-spline method with the number of data points. Growing the number of basis elements with the size of the input made the B-spline method infeasible for time series over a few thousand points in length. Although both the lifting method and cubic splines are linear, our lifting implementation was written entirely in the `R` language while the most of the computations in the cubic spline method are written in the more computationally efficient `C` language.

Finally, we tested whether the classification error varied with the loss function used to compute approximation error. Again, accuracy computed as the out-of-bag classification accuracy of a random forest classifier. Both the mean squared error, mean absolute error(MAE) loss function performed similarly with MSE performing slightly better.

The purpose of the preceding test was to verify that the approximation methods used in our implementation of the $\varepsilon-$complexity algorithm behaved as expected and to compare the performance of the complexity coefficients as the algorithm parameters where altered. Based on these tests we used cubic splines as the approximation method and set MSE as our loss function for the application to EEG classification discussed in chapter 5.

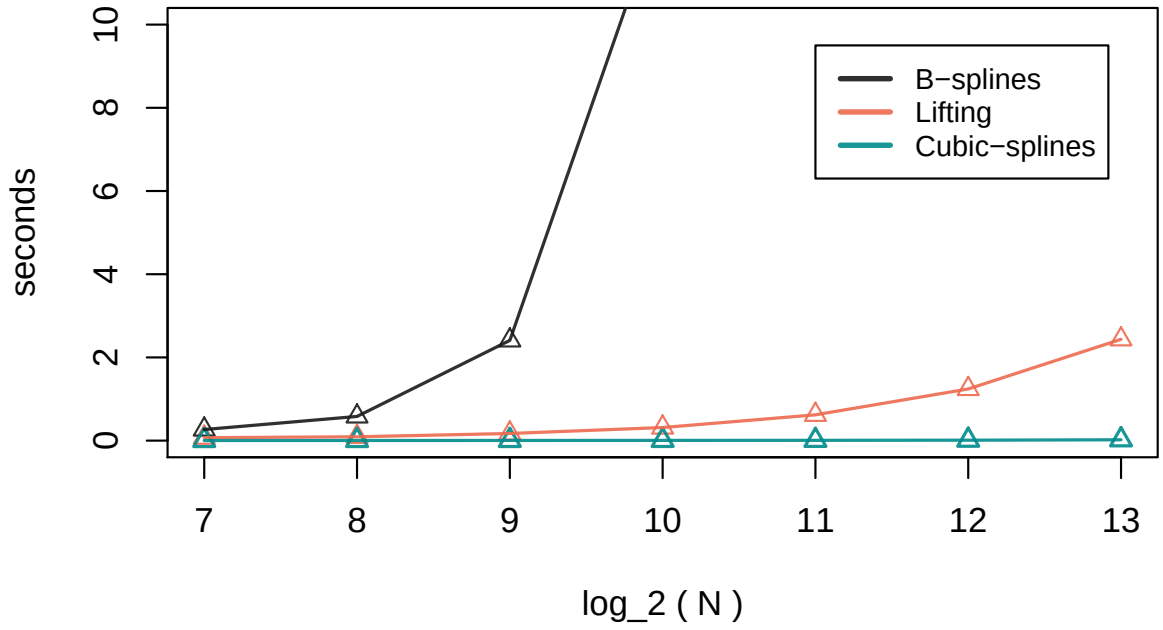

Figure 3.4: Computational time in seconds as function of the log of the inputs.

Function	MAE	Max	MSE
ARMA	0.00	0.33	0.03
Cauchy	0.52	0.47	0.66
FARIMA	0.18	0.39	0.22
FBM	0.13	0.33	0.16
Logistic	0.01	0.18	0.00
Weierstrass	0.44	0.51	0.53

Table 3.3: Classification error using a random forest classifier and three different loss functions.

Chapter 4

Hölder Conjecture

In this chapter, we explore how the complexity coefficients relate to other characteristics of time series. In particular, we use a number of simulations to examine the relationship between the complexity coefficients and both the Hölder class of a function and the fractal dimension of the function's graph. We begin by looking the behavior of the complexity coefficients for a few simple functions with added noise. The complexity coefficients are given by parameters of $\log - \log$ linear fit to points whose original values is less than 1 and the interpretation of the parameters may not be intuitive. These simple examples are meant to orient the reader in interpreting some aspects of the complexity coefficients.

4.1 The Complexity Coefficients

In the theoretical development of $\varepsilon-$complexity, the continuous function to be analyzed was assumed to on the interval $[0, 1]$. Here we refer instead to the index of the samples of the function which, like the time parameter of a time series, are in $\{1, 2, 3, ...\}$. The complexity coefficients are computed by finding the best approximation to a function as that function is downsampled by integer amounts, that is, points indexed by $h\mathbb{N}$ are used in the reconstruction. In our implementation of the algorithm, a function is downsampled at $h \in \{2, 3, 4, 5, 6\}$ giving us five downsampling levels. The minimal approximation error, ε_h is computed for each level h. We denote the proportion of the samples retained $\mathbb{S}_h = \frac{1}{h}$. The $\varepsilon-$complexity coefficients A, B are then result of an ordinary least squares fit:

$$\log(\varepsilon_h) = A + B\mathbb{S}_{\sim}.$$

Figure 4.1 shows a linear function and three sinusoidal functions. Figure 4.2 shows the $log - log$ regression of the errors ε_h on the fraction of samples kept $\mathbb{S}_h$. The complexity coefficients for these fits are given in Table 4.1. While these simple functions will not reveal much about the behavior of the complexity coefficients for more complicated time series, the results highlight some basic properties of the complexity coefficients.

The linear fits in 4.2 determine the complexity coefficients A, B. The $x-$axis is the log of the $\mathbb{S}_h$, percent of points used to approximate the original function at each step. The finest scale approximation is when $h = 2$, represented by the point with the least (log) error located furthest to the right at $log(S) \approx -0.7$. Small values for this error correspond to

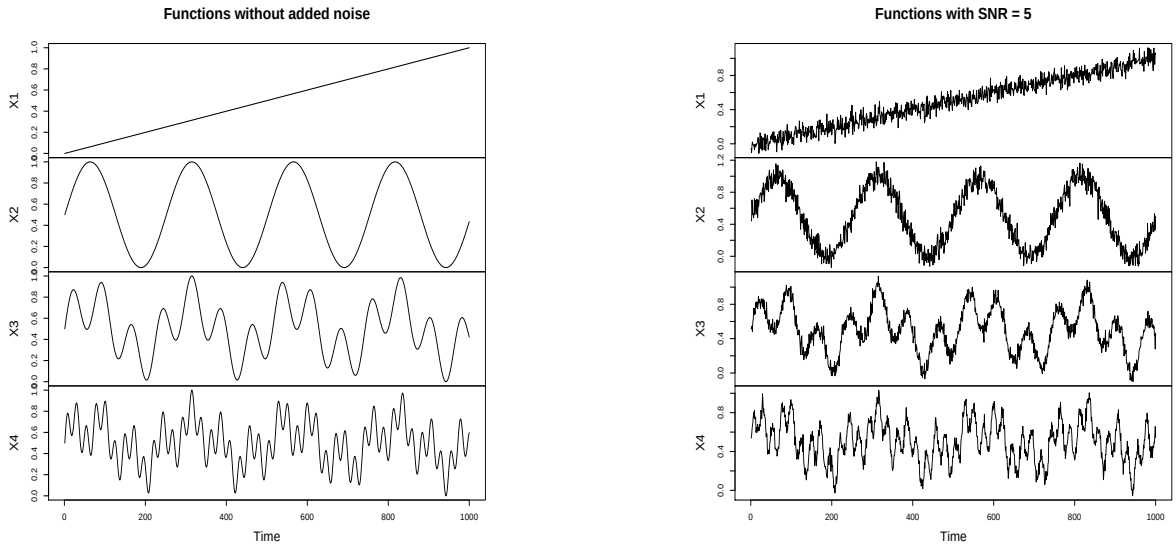

Figure 4.1: Simple linear and sinusoidal functions with and without added noise.

Without Noise	A	B	$\hat{D}$
x	-41.45	-1.92	1.000
$\sin 5x$	-22.71	-4.33	1.001
$\sin 5x + \sin 17x$	-18.71	-4.61	1.001
$\sin 5x + \sin 17x + \sin 53x$	-14.54	-4.81	1.010
SNR = 20	A	B	$\hat{D}$
x	-4.04	-0.57	1.97
$\sin 5x$	-3.9	-0.59	1.94
$\sin 5x + \sin 17x$	-4.07	-0.55	1.94
$\sin 5x + \sin 17x + \sin 53x$	-4.12	-0.52	1.67
SNR = 5	A	B	$\hat{D}$
x	-3.39	-0.54	1.99
$\sin 5x$	-3.48	-0.55	1.99
$\sin 5x + \sin 17x$	-3.58	-0.55	1.99
$\sin 5x + \sin 17x + \sin 53x$	-3.70	-0.58	1.82

Table 4.1: Complexity coefficients and fractal dimension, $\hat{D}$, estimates for a linear and several sinusoidal functions.

accurate approximations and less variability at the finest scale of the sampled function. In theory, the approximation error at zero represents the exact approximation of the function. Since we are computing the log of the approximation error ε, as $\varepsilon \to 0, \log(\varepsilon) \to -\infty$. For the three simple functions in Figure 4.1 the overall approximation error is low, and at least for these examples the intercept value A decreases with simpler functions.

The slope coefficient B should measure the rate at which the approximation error grows as

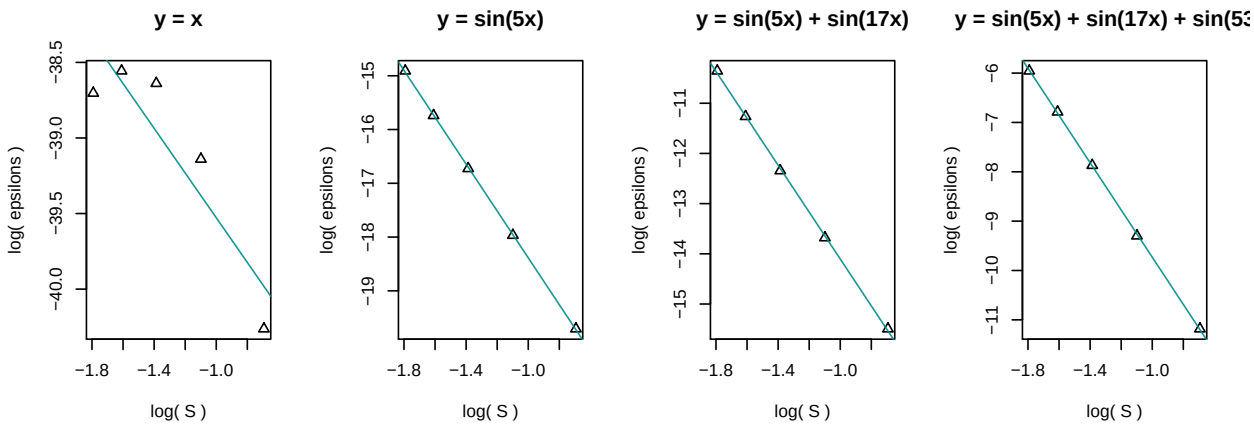

Figure 4.2: The log-log linear fit of the approximation errors against the proprotion of points $\mathbb{S}_h$ used to approximate the functions 4.1

.

fewer points are used to approximate the original function. The coefficient for the sinusoidal functions is fairly similar at each noise level. The relation between the value B and the number of frequency components in the functions is not clear as the order B for these functions changes with added noise.

The fractal dimension reported in Table 4.1 is estimated using a variogram method described in Chapter 2. Below we compare the variogram estimate of the fractal dimension of for a number of functions to the complexity coefficients. Here we point out that increasing noise dramatically affects the fractal dimension estimate $\hat{D}$. This fractal dimension estimator sets the maximum estimate of a function whose graph is in $\mathbb{R}^2$ to the theoretical maximum of 2. Adding a small amount of noise makes the estimator approach its maximum quickly.

4.2 Hölder Class and Fractal Dimension

Theorem 2.1.3 states a relationship between a Hölder class of functions and the complexity coefficients. The proof of the theorem differs in some details from the method of estimating the complexity coefficients. For example, the proof assumes the error is achieved by taking the infimum over a possibly infinite family of functions $\mathcal{F}$, and the max norm rather than the Euclidean norm. In addition, the statement relates the $\varepsilon-$complexity to a class Hölder continuous functions, although the same relation should hold for individual members of that Hölder class. In this section, we use several functions that have a known Hölder exponent and that have parameters that can modulate the Hölder exponent of the function. We use these functions to test the conjecture that the Hölder class of a function can be characterized by its the complexity coefficients.

In Chapter 2 we reviewed some relationships between the Hölder exponent and to the fractal dimension of a function. For example, for a bounded continuous function $f : \mathbb{R} \to \mathbb{R}$,

satisfying the Hölder condition

$$|f(t) - f(t+h)| \leq c|h|^{\alpha}$$

the box counting dimension $\dim_B(f)$ is bounded above by $2 - \alpha$[5]. Our experiments show a close relationship between the variogram estimator of fractal dimension $\hat{D}$ and the slope parameter of the complexity coefficients B. For the functions or sample paths of stochastic processes used in our tests there is a linear relationship between the Hölder class of the function and the functions fractal dimension. If the complexity coefficients and the slope parameter B identify the Hölder class then for these functions, they should identify the fractal dimension of the graphs as well. There is also an intuitive relationship between the variogram estimator of fractal dimension and the method of computing $\varepsilon-$complexity coefficients; for a given increment h, the variogram estimator uses the expected value of the square the increments, $E[(X_t - X_{t+h})^2]$, while the complexity cofficients finds the approximation error over all intervals h. A larger approximation error on increments h likely corresponds with greater expected difference $X_t - X_{t+h}$ and vice-versa.

For the following tests, we generated samples of between 500 and 2000 for each of four functions, the Weierstrass and random phase Weierstrass functions, fractional Brownian motion(FBM) and the Cauchy process. The first three have a single parameter α which controls both the Hölder exponent and fractal the dimension of each while the Cauchy process has two parameters which separately control the fractal dimension and long-range dependence of the function.

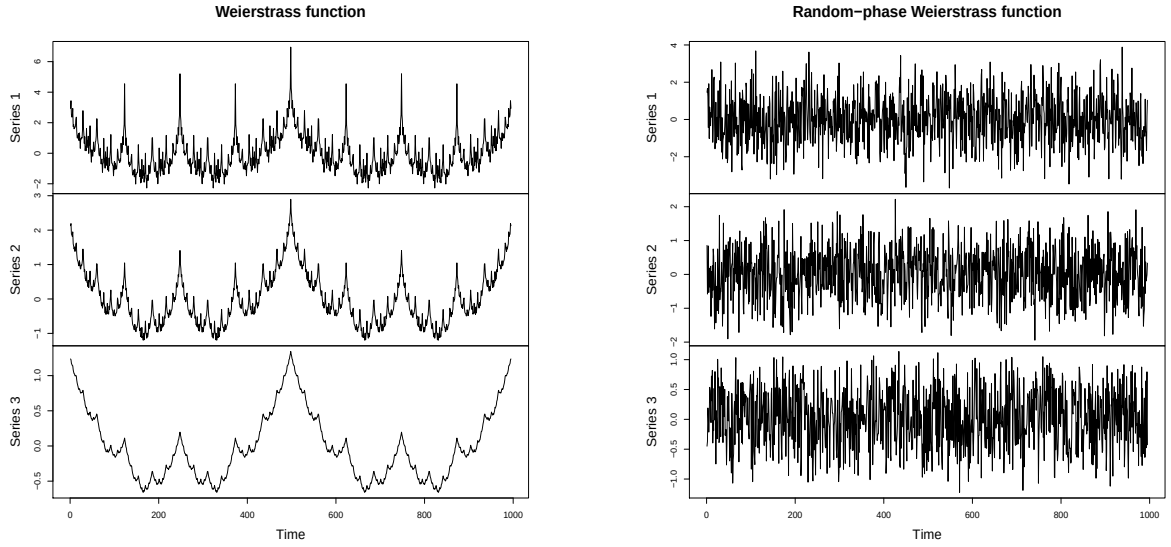

Figure 4.3: The Weierstrass and random-phase Weierstrass function for $\alpha = \{0.20, 0.42, 0.80\}$

The Weierstrass function written

$$W_{\alpha}(x) \quad = \sum_{n=0}^{\infty} b^{-n\alpha} \cos(b^n \pi x).$$

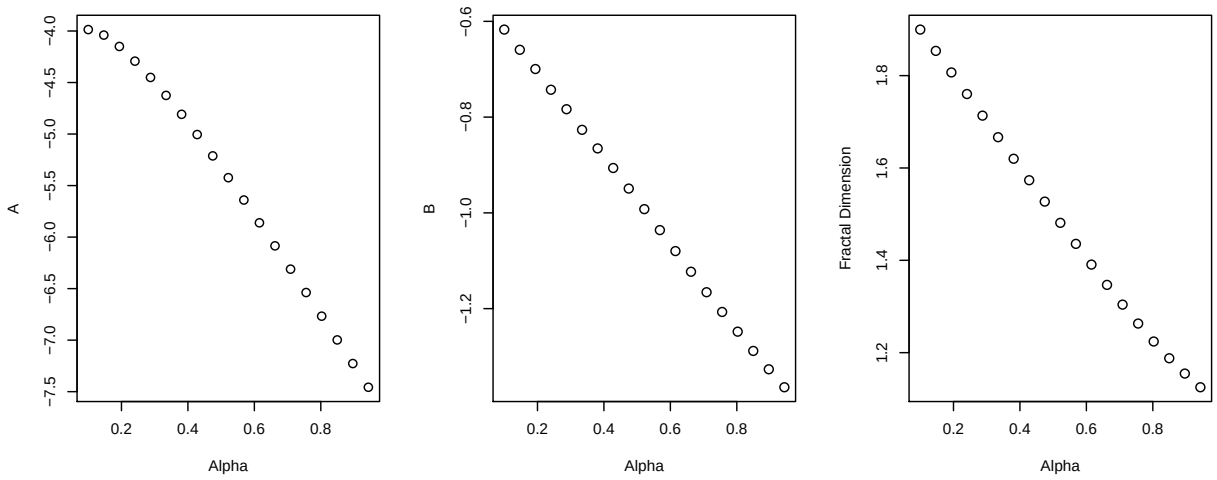

Figure 4.4: Complexity coefficients and fractal dimension plotted against the Weierstrass functions α parameter.

is Hölder α for $0 < \alpha < 1$ and has box-counting dimension $D - \alpha$. For the random-phase Weierstrass function, as proved by Hunt[8], the parameter α determines the Hausdorff dimension of its graph. The two functions are illustrated in Figure 4.3 for three different paramaeter values $\alpha = \{0.20, 0.42, 0.80\}$. For smaller α the fractal dimension D of the graphs increase.

Figure 4.4 shows the change in the complexity coefficients A and B and the variogram fractal estimator $\hat{D}$ as the parameter α is varied. The relation of both the complexity coefficient B and fractal estimator $\hat{D}$ to α is linear. For the complexity intercept A there is some non-linear change for low values of α. The fractal estimator $\hat{D}$ closely tracks the theoretical fractal.

The results for the Weierstrass function are deterministic, so the results shown are for a single estimate for each parameter. The remaining functions are stochastic so the same the test was run on fifty samples functions or simulations the estimated complexity coefficients and fractal dimension estimate are displayed as box plots to shown the distribution of the estimates.

For the random-phase Weierstrass function Figure 4.5 shows there was no relationship between the change in the α parameter and the complexity coefficient B or fractal estimator $\hat{D}$. The intercept coefficient A increases linearly but the total magnitude of the change is relatively small– the median of the estimates has a range of less than 0.5. The reported results were for simulations of 2000 points. The simulation was repeated at for lengths between 500 and 2000 but the returned the same results. We do not have an explanation for the discrepancy between the theoretical Hölder class and fractal dimension of the function and the results of the simulation. Unlike the determinstic Weierstrass function and the Cauchy and Fractional Brownian motion processes, the graph of the random-phase Weierstrass did not become locally smoother as α was increased and the coefficient B and estimator $\hat{D}$ are

numerically similar to the results given when white noise was added to the simple functions as reported in Table 4.1.

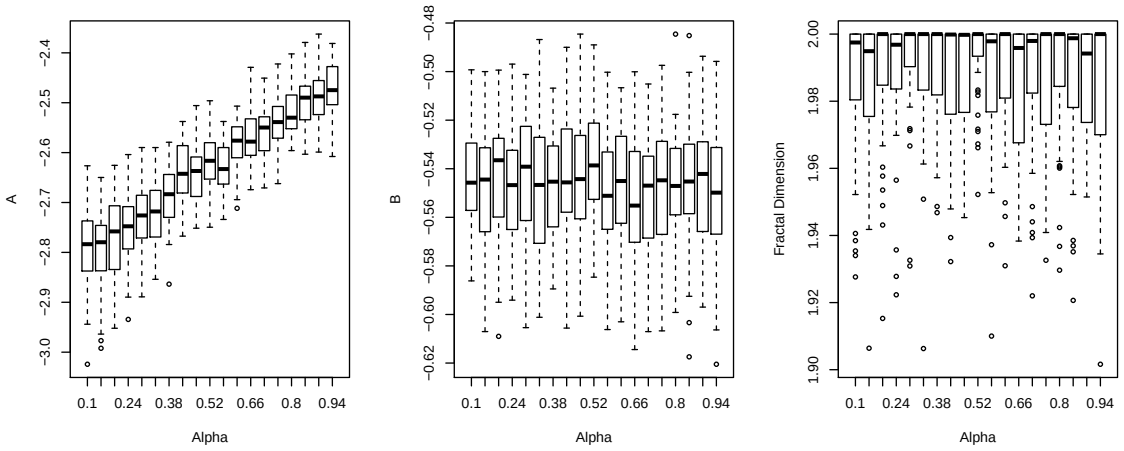

Figure 4.5: Complexity coefficients and fractal dimension plotted against the α parameters of the random-phase Weierstrass function.

Examples of the FBM and Cauchy process time series used in our simulation are shown in figure 4.6. The results are similar to those of the deterministic Weierstrass function and both the complexity coefficient B and $\hat{D}$ change linearly with the Hölder class parameter. Also similar to the results for the Weierstrass function, the coefficient A has a non-linear relation to α.

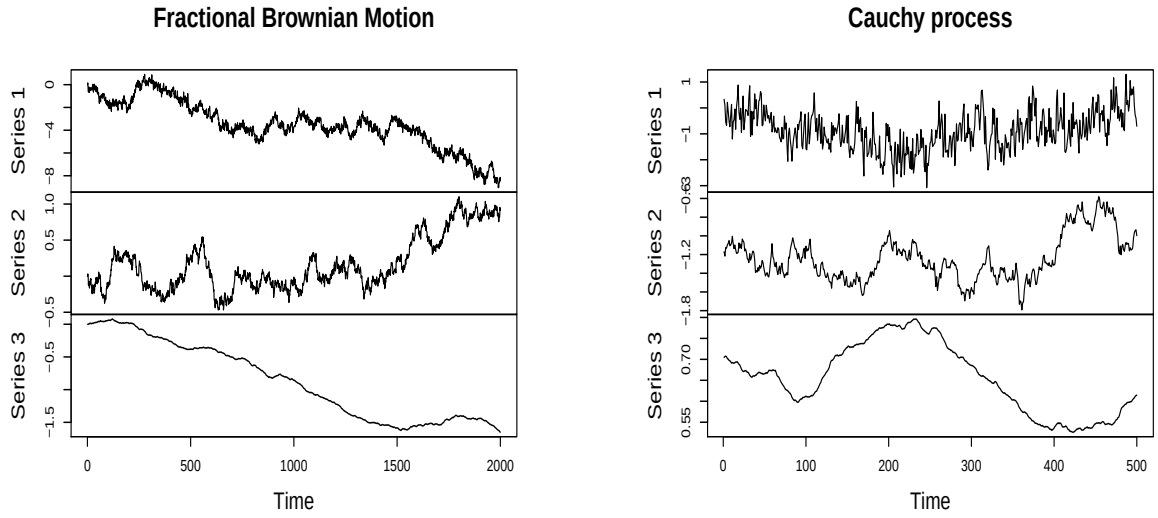

Figure 4.6: Fractional Brownian motion and the Cauchy process with $\alpha = \{0.20, 0.42, 0.80\}$. The Cauchcy β parameter was held constant.

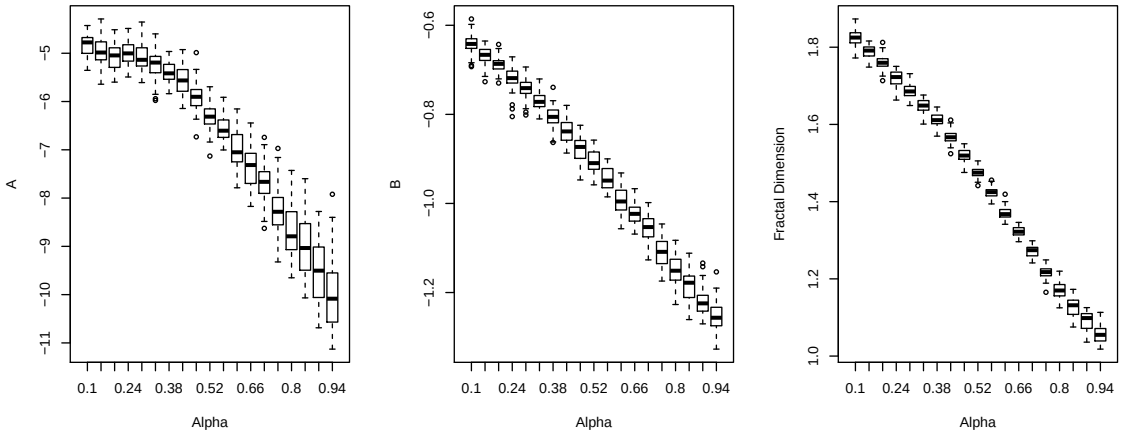

Figure 4.7: Complexity coefficients and fractal dimension plotted against the α coefficients of fractional Brownian motion.

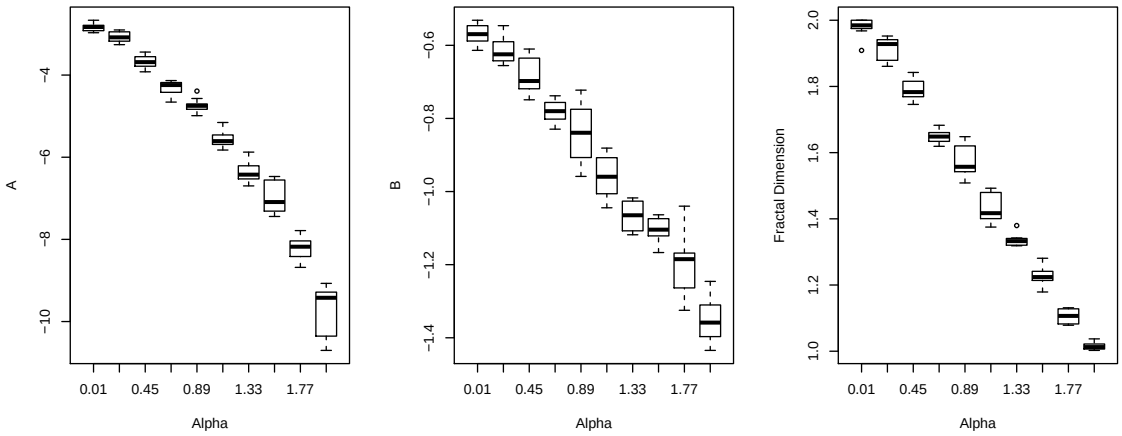

Figure 4.8: Complexity coefficients and fractal dimension plotted against the Cauchy process α coefficient.

The Cauchy process has two parameters, α determines fractal dimension and the parameter β determines the Hurst coefficient, or the long range correlation of the process. 20 simulations of the Cauchy process were generated on a 10×10 grid of parameters α, β for $0 < \alpha < 2$ and $0 < \beta < 1.5$. In figure 4.9 plots the mean value of the complexity coefficients as the long-range parameter α changes. Differences in the β parameter show no effect the estimate of the complexity coefficients or the fractal estimator.

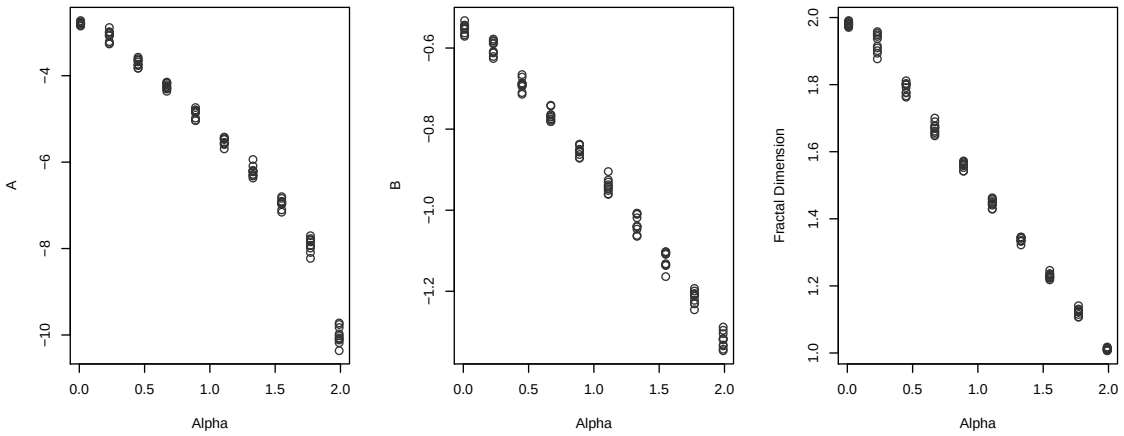

Figure 4.9: The mean value of the complexity coefficients and fractal dimension plotted against the Cauchy process α coefficient.

Figure 4.10 shows another view of the complexity coefficients plotted against changes the Cauchy process parameter *beta*. The complexity coefficient B and fractal dimension clearly preserve the grid from the parameter space – for a fixed β, the parameter varies linearly with α. The β parameter controls the long-range dependence of the Cauchy process and the plots indicate that the complexity coefficients, like fractal dimension, measure a local, not global, property of a function.

Although these results are intended to be exploratory, for three of the four functions there is strong evidence that the complexity coefficient B behaves similarly to the variogram estimator of fractal dimension. The complexity coefficient A does not seem to add much information for the three cases in which there was a linear relation between B, $\hat{D}$, and the parameter determining the Hölder exponent of the function, α. On the other hand, our initial example of simple functions and the random-phase Weierstrass functions appears shows a relationship between the intercept coefficient A and fine-scale noise. These results are for a small set of processes with well-known behavior. A comparison of the complexity coefficient B and fractal estimators on a wider variety of time series might show whether this relation holds under a wider range of conditions.

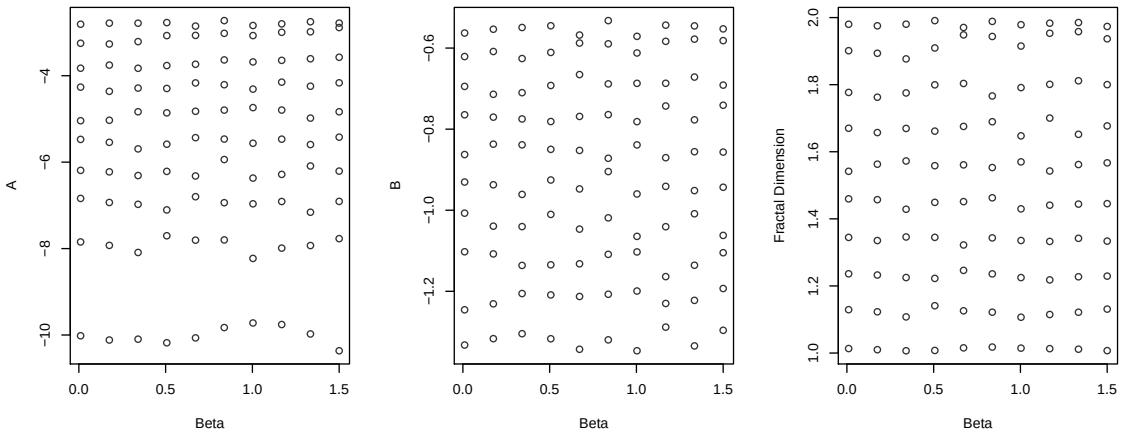

Figure 4.10: The mean value of the omplexity coefficients and fractal dimension plotted against the Cauchy process β coefficient.

Chapter 5

Seizure Prediction

Electroencephalograms(EEG) capture the electromagnetic potential of similarly aligned neurons firing in unison. Although a number of correlations between EEG and physiological phenomena have been found the relation between EEG and neural circuitry dynamics are still not well-understood [2]. one difficulty is relating in vitro behavior of individual neural cells with in vivo behavior of both local and global neural dynamics. For a neuropathological condition like epilepsy, the challenge is to relate global spread of synchronized firing to local cellular mechanisms.

In this chapter, we analyze EEG data gathered from a group of mice afflicted with a genetic mutation that causes early onset epilepsy similar to Dravet syndrome. This was an exploratory study to assess how well stimulus-induced seizures could be predicted and to find those features correlated with the likelihood of a seizure.

The study of complex biological signals like EEG was one of the motivations for developing the $\varepsilon-$complexity feature. EEG are non-stationary signals that exhibit transient waveforms and apparent regime changes over relatively short periods of times. In the context of dynamical systems, a regime change describes an alternation between semi-stable states [10]. While we do not assume the brain dynamics can be described as a non-linear dynamical system, the term is useful for characterizing the observed behavior of EEG signals.

A common method of analyzing EEG is to compute some set of features on the raw signal and use averages of the features over fixed windows. The procedure serves as a way of reducing the high-dimensional EEG signal and extracting a more interpretable feature set. One drawback of this method is that the process of averaging over arbitrary windows may obscure important features associated with the varying dynamics of the brain.

Previous studies have used a change-point detection algorithm applied to the raw EEG signal to segment and cluster these varying brain states, for example, the sleep states of neo-nates[12]. The $\varepsilon-$complexity feature is meant to capture an intrinsic feature of the signal. As shown in Chapter 4, the feature may also be related to the fractal dimension of a signal which Gneiting describes as a second order statistical property of a signal through its relation to the variogram or correlation function[7].

In this chapter, we design a classifier that modifies the typical feature extraction method segmenting the signal based on the $\varepsilon-$complexity coefficients. We compare classification accuracy on a dynamically segmented set of features to several models that uniformly segment the feature set. For each of these models, we attempt to predict seizures based on features

extracted from individual EEG channels rather than from the full set of six. The set of EEG channels correspond to specific functional regions of the brain. Feature extraction and classification based on individual channels allows us to make more fine-grained inferences about the combination of feature and region that are associated with a seizure response.

5.1 EEG Data

The EEG data gathered from 4 mice with a genetic mutation in the voltage-gated sodium channel gene Scn1a a similar mutation to that causing Dravet syndrome The mutation results in early onset epileptic seizures [9]. The mice were equipped with 4 electrocortical(ECoG) sensors –intracranial sensors placed directly on the brain cortex – along with 2 local field potential (LFP) sensors located in the thalamus. The mice were genetically modified to produce a light sensitive protein, an opsin, allowing for direct stimulation of neurons with a laser pulse train.

Data was gathered from 13 distinct time periods during which the was applied one to four times. 4-minute segments preceding the stimulus were used as the final set of 26 trials. The results of each trial was were coded as either a seizure, which we refer to as a positive response, or no seizure. Coding was based on the visible response of the mouse. The EEG signals around the stimulus period were also visually examined to check the accuracy of the coding.

The EEG was sampled at 1220.7 Hz and a bandpass filter removed frequencies below 0.5 Hz and greater than 300 Hz. A notch filter was applied at 60 Hz and its harmonic frequencies.

5.2 EEG Features

We selected a feature set consisting of standard spectral characteristics along with a few additional features. The power in frequency bands delta, theta, alpha, beta, and gamma was calculated as the integral of the spectral density $f(\lambda)$ computed as a smoothed periodogram. For example, delta band power, δB, corresponds to the frequency band $0.5 - 4Hz$ and integrates $f(\lambda)$ over over this interval

$$\mathbf{B} = \int_{0.5}^{4} f(\lambda) d\lambda.$$

The frequency bins corresponding to the remaining bands are, in order,$4-8Hz, 8-12Hz, 12-30Hz, 30 - 100Hz$.

Additional features were selected based on two tests. A set of non-linear features including sample entropy, and permutation entropy, spectral entropy, a Hurst parameter estimator and fractal dimension along with wavelet coefficients and variance were tested for their ability to discriminate between the same two groups of simulations used in Chapter 3 to test the approximation methods used in estimating complexity coefficients. Based on these results, spectral entropy, fractal dimension, the Hurst parameter and the wavelet coefficients, along with the spectral band power features and the complexity coefficients were computed on the

EEG. In informal testing with a baseline classification model, the wavelet coefficients did not improve classification results and were removed from the final feature set. Given more data, feature selection might have been better performed through cross-validation on a training set. On the other hand, one of our goals was to work with a reduced feature set that was more readily interpretable.

We have defined most of the features used in classification in Chapter 2. Spectral Entropy is a measure of the distribution in the spectrum of a signal.

$$SE = -\int_{-\pi}^{\pi} f_x(\lambda) \log f(\lambda) d\lambda.$$

The Hurst parameter was estimated using the corrected empirical Hurst exponent[20].

All features were calculated on non-overlapping 2-second intervals. The final list of feature used in classification was

1. Delta, theta, alpha, beta, gamma
2. Spectral entropy
3. Hurst parameter
4. Variance

The complexity coefficients were used in our segmentation classifier but not as a feature. The variogram estimator of fractal dimension, $\hat{D}$ was also computed but was omitted from the final models as the feature appeared closely correlated with the complexity coefficient B.

5.3 Segment Classifier

Whether it is spectral features or $\varepsilon-$complexity features, the mapping of the raw data to the features results both in a dimensionality reduction and an averaging over some variation and structure in the data. That is, there is a trade-off between losing important variation in the data and reducing the dimension of the raw input. In our approach, we compute features in three basic steps. Features are computed at a relatively fine resolution for each EEG channel. Given an individual channel, the time series formed by the features is segmented based on changes in the complexity coefficients. The weighted averages of the features set on these segments is used as a training set for the classifier.

1. For each channel compute set of features on regular intervals including $\varepsilon-$complexity.
2. For each channel, compute the change points in $\varepsilon-$complexity.
3. For each trial, segment all features using the change point set computed in the previous step.
4. Compute the mean of each feature on the segments.

5. Label the means of the segments in a training set with the label of the full time series and use this set to train a classifier.

6. Train a classifier on the means of the segments.

7. For the means of the segmented test set, compute the class probability of each segment using the trained classifier.

8. For each trial use the prediction probabilities of each segment weighted by segment length to compute the final class probability.

In the case of a uniform partition of the features, this the method is equivalent to simply using the average class probability for each segment to classify the trial.

In theory, any classifier, or several classifiers could be used in steps 5 and 6. We used a random forest classifier. The random forest is constructed from a large number of individual decision trees. For numeric data, the decision trees partition the feature space into $d-$dimensional rectangles each labeled with a class. The decision trees branches correspond to partitions of the feature space. For each branch, a subset of the features are chosen and the split is made in order to increase the class purity of each partition. The final prediction is based on the combined vote of the individual trees. In addition to random selection of features, a random subset of observations are used to build each tree. This allows for an out-of-bag(OOB) estimate of the classifier accuracy which is calculated by classifying each observations using the set of trees which were built without seeing that observation [1].

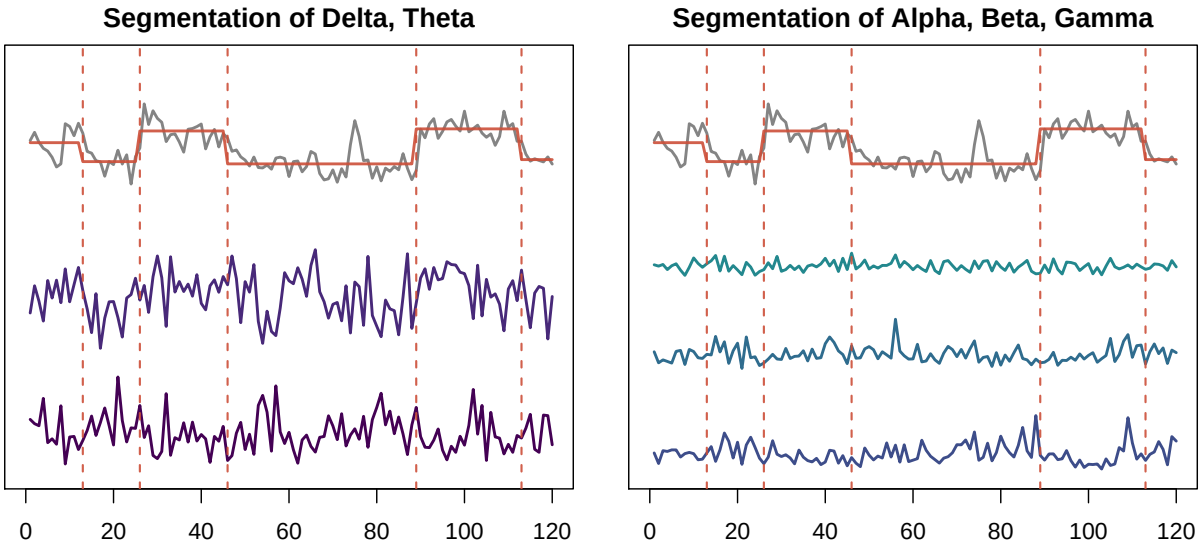

Figure 5.1: Spectral features segmented on changes in the B complexity coefficient.

5.4 Model Performance

We begin by defining several terms we will be using to describe classifier performance. We term a seizure a positive response or simply a response. A *true positive* is an accurate

prediction of a seizure while a *true negative* is an accurate prediction of a non-response. The *sensitivity* of the classifier is then defined as the proportion of true positives to total positive and *specificity* is the total of true negatives to total negatives. We use the term accuracy to refer to balanced accuracy

$$\text{Balanced Accuracy} = \frac{\text{Sensitivity} + \text{Specificity}}{2}.$$

We also report AUC, or area under the ROC curve, see Figure 5.4 for an example from two of the classifiers tested. The 'curve' is the plot of the sensitivty against the false positive rate or $1-$specificity and a higher AUC indicates a better performing classifier.

We created a simple baseline classification model to compare to the segmented classifiers. The feature set for this model was the mean of each of the 8 features on the six channel resulting in 48 features. A random forest classifier was trained on these features and the out-of-bag classification results are reported in table 5.1.

	Sensitivity	Specificity	Accuracy
1	0.544	0.935	0.740

Table 5.1: Classification performance of baseline classifier.

We compared six models to this baseline model. For each model a different partition scheme was used. For the models using complexity coefficients we denote $A, B, A+B$ where each trial was partitioned based on change points detected in the corresponding complexity coefficients and the $A+B$ model is simply the union of the change points of both complexity coefficients. Three other models used regular partitions. Each trial was uniformly segmented into $8, 15$ or 30 segments and we refer to these models by the partition number.

The performance of these models was assessed using 5-fold cross-validation where balanced sets were used for the hold-out set for each fold. The reported results are based on 10 repetitions or bootstrap samples. Results for a larger number of repetitions, 50, were similar. We report to smaller sample to emphasize the relative sizes of the confidence intervals of the reported performance measures. In general, all models performed best on the LFP channels, here labeled channel 1 and 2. These sensors are located in the thalamus and measure the electrical potential of a small set of neurons. The best performing model was the regularly partitioned model with 8 segments followed by the model $A + B$.

The performance measures for each classifier on channel 1 are shown in figure 5.2. The mean and 95% confidence interval based on 10 bootstrap classifications are reported. All methods classified non-response trials well but the two better performing models – model 8 and model $A + B$ – had relatively high accuracy in classifying seizure responses: 76% for model $A + B$ and 79% for model 8.

Classification is generally more difficult when classes are unbalanced, that is, when there are significantly fewer observations in the training set of one class compared to the other. Other than performing cross-validation on a balanced hold-out set, no other adjustments were made in the underlying random forest model or the threshold probability for selecting a class.

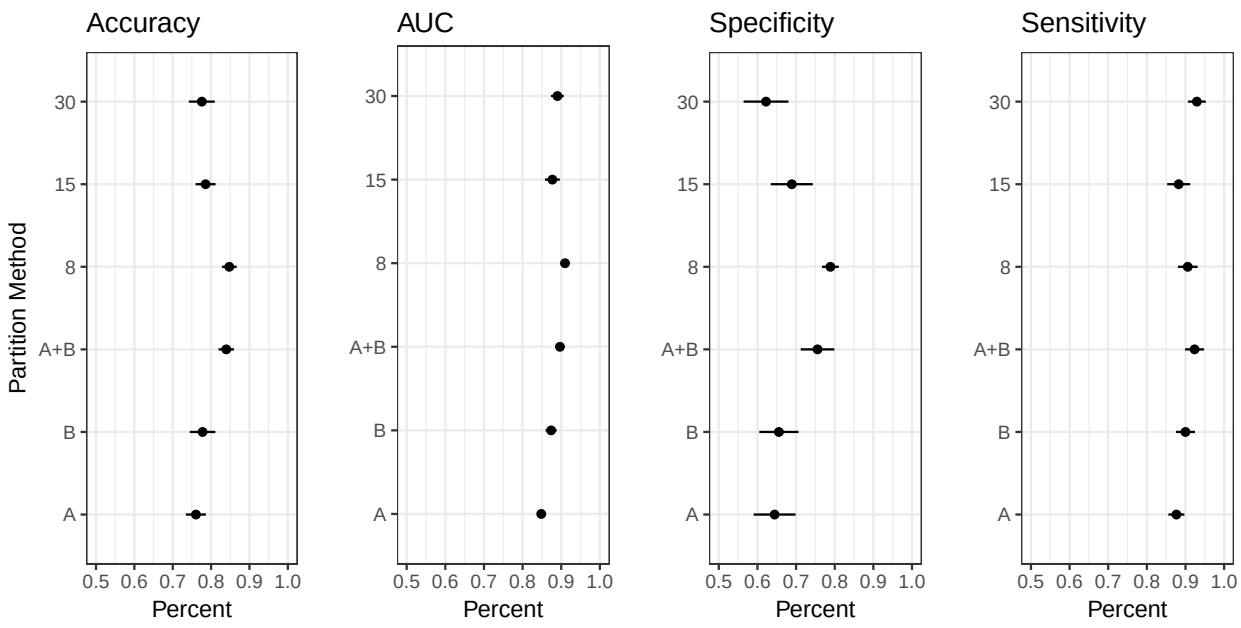

Figure 5.2: Classification diagnostic values for each partition method for LFP channel 1.

	Ch1	Ch2	Ch3	Ch4	Ch5	Ch6
A	0.76	0.76	0.70	0.71	0.57	0.60
B	0.78	0.77	0.70	0.69	0.60	0.64
A+B	0.84	0.70	0.69	0.71	0.59	0.61
8	0.85	0.81	0.69	0.71	0.68	0.82
15	0.79	0.79	0.73	0.71	0.73	0.73
30	0.78	0.80	0.72	0.71	0.72	0.84

Table 5.2: Balanced accuracy for each model and channel

	Ch1	Ch2	Ch3	Ch4	Ch5	Ch6
A	0.64	0.58	0.46	0.49	0.23	0.28
B	0.66	0.59	0.48	0.53	0.36	0.34
A+B	0.76	0.46	0.46	0.52	0.28	0.30
8	0.79	0.68	0.48	0.51	0.50	0.69
15	0.69	0.63	0.52	0.50	0.62	0.53
30	0.62	0.67	0.51	0.49	0.57	0.76

Table 5.3: Specificity for all models and channels.

The balanced accuracy all models for each channel is shown in Table 5.2. All partition schemes resulted in fairly high rates of accuracy for channels 1 and 2. The models with a higher number of partitions performed significantly better on the ECoG channels 3-6. On the other hand, accurate classification of seizures tailed off sharply for all ECoG channels as seen in Table 5.3.

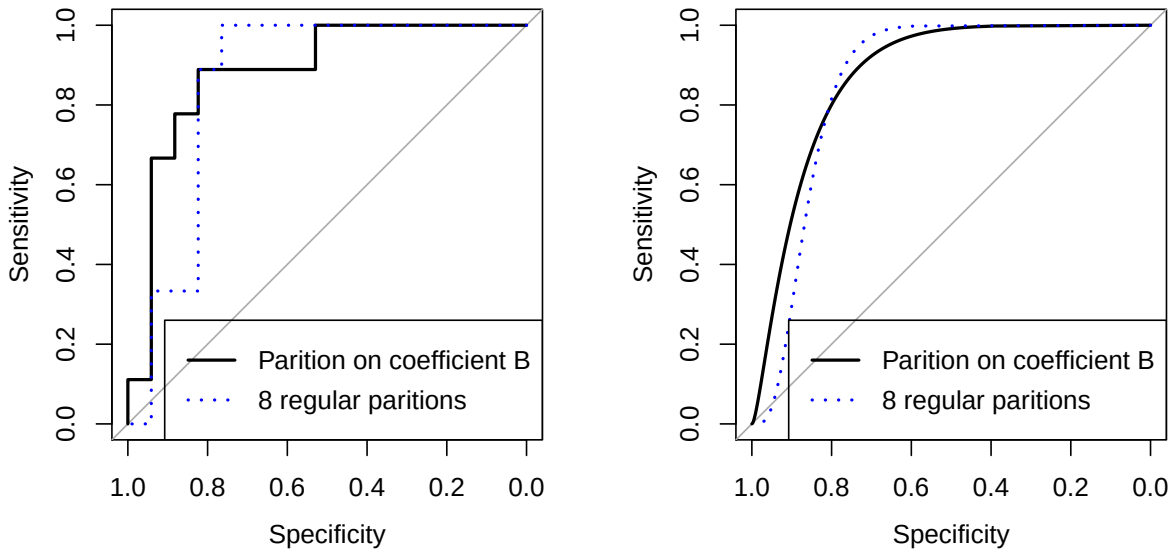

Figure 5.3: ROC curves and smoothed ROC curves for classification on channel 1 using partitions on coefficient B and a regular partition into 8 segments.

5.5 Feature Importance

As mentioned above, with only a small set of observations, improved classification on one or two instances would affect the performance of the classifier. Whether the model used here – partitions based on complexity coefficients – would be successful in other contexts remains to be seen. One of our assumptions was that arbitrary partitioning would average over changes variations in the features that corresponded to transient states. But whether the complexity coefficients capture significant changes in the underlying dynamics that are useful for classfication might depend on both feature selection and the classification task.

A goal of our study was to determine features predictive of seizures. The selection of a reduced the feature set and prediction on isolated channels allow us to measure the feature importance for classification for each sensor location. The distributions of features for response and non-response trials are looked at below. Examining the one-dimensional distribution of the feature may not reveal interactions between the features in higher dimensions. On the other hand, decision trees divide up the feature space in complex ways and the final ensemble votes are what determines the classification outcome. In theory, those features found to be useful in classification will not necessarily be identifiable with basic summary

statistics such as the mean, median or variance. However, for our data, the features used by the best predictors were associated with significant distributional differences across classes.

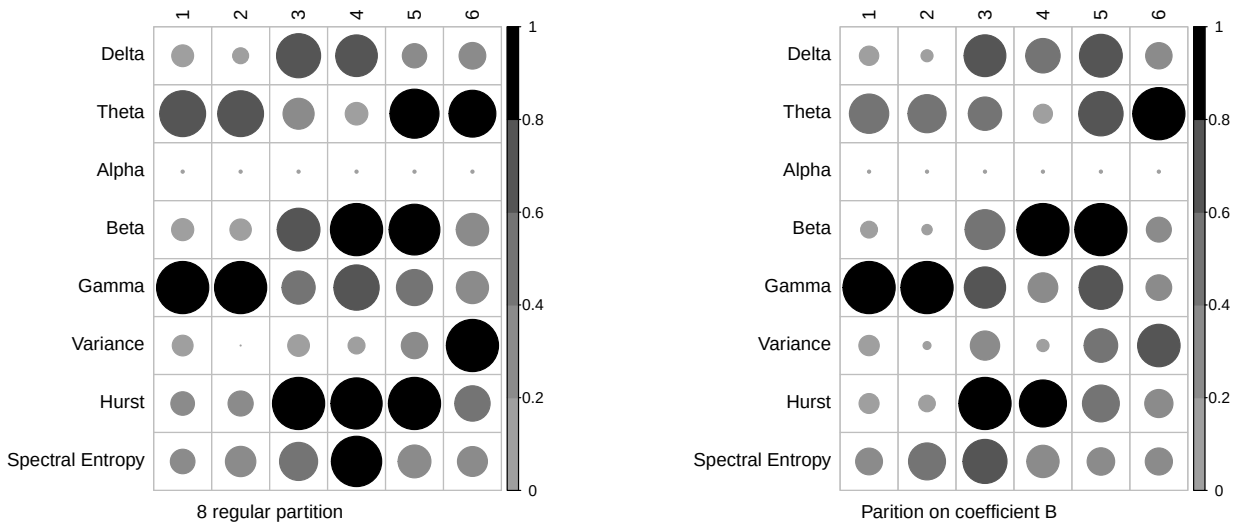

Figure 5.4: Variable importance for partition methods normalized to a $[0, 1]$ interval for each channel.

We report the mean variable importance of the random forest classifiers trained during cross-validation. Variable importance is determined by the mean decrease in the class Gini index when branches are made on a given feature [1]. Informally, variable importance measures how well a feature divides response trials from non-response trials. In Figure 5.4 we show the variable importance for the two best performing models $A + B$ and the model 8. We have normalized variable importance to a $[0, 1]$ interval for each model and channel so the figure shows the relative importance of the variable for each model. There is a common pattern in the variable importance across the two models. Gamma and theta bands have the highest importance for the best performing models, those trained on the LFP channels 1 and 2. Both partition methods also show increased importance for beta on channels 4 and 5 and increase the importance of the Hurst coefficient for channels 3 and 4.

Variable importance as measured by the random forest classifiers was also reflected in distributional differences in the features. The boxplots in figure 5.5 show the distribution and features for channels 1 and 3. For channel 1, the relative power of gamma is higher and theta lower for trials with seizure responses. The median of theta and gamma fall outside the inter-quartile range and a similar distribution was similar for channel two. For channel three, the relative power of theta and gamma are reversed with gamma lower and theta higher. On the other hand, the distribution of beta for channels 1 and 2 were similar while beta was significantly lower for channels 3 and 4. Table 5.4 shows the p-value determined by

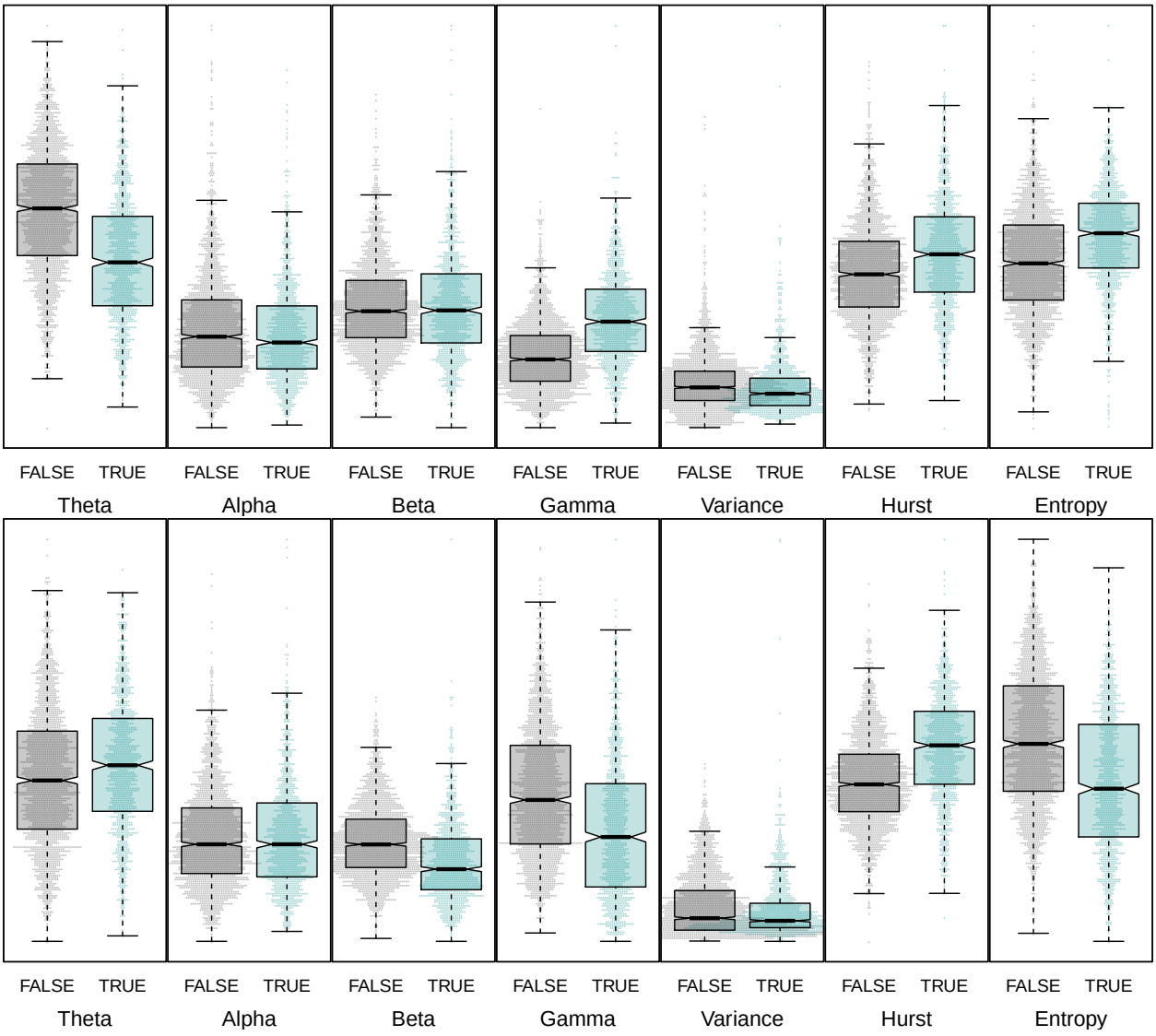

Figure 5.5: Feature distribution for channels 1 and 3.

the Wilcox rank-sum test. While a large number of data points guarantees that even small differences will be statistically significant, the table does highlight the non-significant values.

Channel	1	2	3	4	5	6
Delta	$< .0001$	$< .0001$	$< .0001$	$< .0001$	$< .0001$	$< .0001$
Theta	$< .0001$	$< .0001$	$< .0001$	$< .0001$	$< .0001$	$< .0001$
Alpha	0.022	0.699	0.897	0.891	0.008	0.302
Beta	0.826	0.226	$< .0001$	$< .0001$	$< .0001$	$< .0001$
Gamma	$< .0001$	$< .0001$	$< .0001$	$< .0001$	$< .0001$	$< .0001$
Variance	$< .0001$	$< .0001$	0.687	0.005	$< .0001$	$< .0001$
Hurst	$< .0001$	$< .0001$	$< .0001$	$< .0001$	$< .0001$	$< .0001$
Spectral Entropy	$< .0001$	$< .0001$	$< .0001$	$< .0001$	0.0003	$< .0001$

Table 5.4: Wilcox rank sum test p-values for difference of medians.

5.6 Discussion

While we were able to predict a seizure result with relatively high accuracy using several classification methods, there are some limitations to the study. Several mice were used in the experiments but the trials resulting in seizures came from a single mouse. Additional data from other mice would be needed to know whether the results reported here generalize. In addition, most of the seizures came from stimuli applied within two trials. This means several seizures occured after a previous seizure. The features found predictive of a seizure may be conflated with features that are a result of a seizure. Because of the small number of trials, we did not have a hold-out data set. The lab from which produced this set of data is currently collecting additional data which will enable us to test how well the predictive models generalize and whether the observed differences in features hold for other subjects.

The segmentation model based on the complexity coefficients performed as well or somewhat better than two of the models with regular partitions. However, the regular partition models tended to perform more consistently across channels and the partition model with 8 segments outperformed the models segmented on the complexity coefficients. All models classified negative responses well so differences in model performance were based mostly on the classification of seizure responses of which there were only 9. The partition based on complexity coefficients is based on a change point algorithm which is somewhat sensitive to the number of data points. Using an increased number of data points, for example, by taking measurements on an overlapping sliding window rather than on non-overlapping windows, may reduce the variability in how a time series is segmented based on the complexity coefficients. Tests of the model on simulated data sets or a wider range of time series would be needed to assess the performance of the model in more general contexts.

Bibliography

[1] Leo Breiman, *Random forests*, Machine learning **45** (2001), no. 1, 5–32.

[2] Michael X Cohen, *Where does eeg come from and what does it mean?*, Trends in Neurosciences (2017).

[3] Germund Dahlquist and Ake Björck, *Numerical methods in scientific computing. Vol. I*, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2008. MR 2412832

[4] Boris Darkhovsky and Alexandra Piryatinska, *Epsilon-complexity of continuous functions*, (2013).

[5] Kenneth Falconer, *Fractal geometry*, second ed., John Wiley & Sons, Inc., Hoboken, NJ, 2003, Mathematical foundations and applications.

[6] Tilmann Gneiting and Martin Schlather, *Stochastic models that separate fractal dimension and the Hurst effect*, SIAM Rev. **46** (2004), no. 2, 269–282.

[7] Tilmann et al. Gneiting, *Estimators of fractal dimension: assessing the roughness of time series and spatial data*, Statist. Sci. **27** (2012), no. 2, 247–277.

[8] Brian R. Hunt, *The Hausdorff dimension of graphs of Weierstrass functions*, Proc. Amer. Math. Soc. **126** (1998), no. 3, 791–800.

[9] Susumu Ito, Ikuo Ogiwara, Kazuyuki Yamada, Hiroyuki Miyamoto, Takao K Hensch, Makiko Osawa, and Kazuhiro Yamakawa, *Mouse with na v 1.1 haploinsufficiency, a model for dravet syndrome, exhibits lowered sociability and learning impairment*, Neurobiology of disease **49** (2013), 29–40.

[10] Edward N. Lorenz, *Regimes in simple systems*, J. Atmospheric Sci. **63** (2006), no. 8, 2056–2073.

[11] Steven Orey, *Gaussian sample functions and the Hausdorff dimension of level crossings*, Z. Wahrscheinlichkeitstheorie und Verw. Gebiete **15** (1970), 249–256. MR 0279882

[12] Alexandra Piryatinska, Gyorgy Terdik, Wojbor A Woyczynski, Kenneth A Loparo, Mark S Scher, and Anatoly Zlotnik, *Automated detection of neonate eeg sleep stages*, Computer methods and programs in biomedicine **95** (2009), no. 1, 31–46.

[13] Jianqing Fan; Yao Qiwei, *Nonlinear time series*, Springer, New York, 1993.

[14] R Core Team, *R: A language and environment for statistical computing*, R Foundation for Statistical Computing, Vienna, Austria, 2016.

[15] Jeffrey S. Racine, *A primer on regression splines*.

[16] Wim Sweldens, *The lifting scheme: a construction of second generation wavelets*, SIAM J. Math. Anal. **29** (1998), no. 2, 511–546.

[17] Wim Sweldens and Peter Schröder, *Building your own wavelets at home*, Wavelets in the Geosciences (2000), 72–107.

[18] Michael Unser, *Splines: A perfect fit for signal and image processing*, IEEE Signal processing magazine **16** (1999), no. 6, 22–38.

[19] Paul Vitanyi and Ming Li, *An introduction to kolmolgorov complexity and its applications*, Springer-Verlag, New York, 1993.

[20] RafałWeron, *Estimating long-range dependence: finite sample properties and confidence intervals*, Phys. A **312** (2002), no. 1-2, 285–299.