IMPROVEMENT OF EPSILON COMPLEXITY ESTIMATION AND AN APPLICATION TO SEIZURE PREDICTION

A thesis presented to the faculty of
San Francisco State University
In partial fulfilment of
The Requirements for
The Degree

Master of Arts
In
Mathematics

by

Nathanael Aff

San Francisco, California

August 2017

CERTIFICATION OF APPROVAL

I certify that I have read *IMPROVEMENT OF EPSILON COMPLEXITY ESTIMATION AND AN APPLICATION TO SEIZURE PREDICTION* by Nathanael Aff and that in my opinion this work meets the criteria for approving a thesis submitted in partial fulfillment of the requirements for the degree: Master of Arts in Mathematics at San Francisco State University.

Dr. Alexandra Piryatinska
Associate Professor of Mathematics

Dr. Tao He
Assistant Professor of Mathematics

Dr. Chun Kit Lai
Assistant Professor of Mathematics

IMPROVEMENT OF EPSILON COMPLEXITY ESTIMATION AND AN APPLICATION TO SEIZURE PREDICTION

Nathanael Aff
San Francisco State University
2017

The ε-complexity of a continuous function is a measure of the amount of information needed to reconstruct a function with an absolute error not larger than ε. For Hölder class functions, ε-complexity is characterized by a pair of real numbers, the complexity coefficients. The complexity coefficients have been shown to be useful features for the segmentation and classification of time series. In this work, we extend the set of approximation methods used in estimating the complexity coefficients. The performance of these approximation methods is tested on a number of simulated time series. In addition, we test the conjecture that, for a given generating mechanism, the mean of the complexity coefficients is constant. For our set of simulations, we find that the mean of the estimated complexity coefficients is constant on a constant Hölder class of functions. Finally, we apply the ε-complexity coefficients to the prediction of seizures in epileptic mice. We use this technique to identify which EEG signal preceded a seizure with over 80% accuracy.

I certify that the Abstract is a correct representation of the content of this thesis.

Chair, Thesis Committee Date

ACKNOWLEDGMENTS

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

Many natural phenomena generate time series that exhibit complex functional and statistical characteristics. For example, an electroencephalogram(EEG) records the electrical potential generated by the synchronized firing of neurons. EEG signals are marked by both transient variations in waveforms and regime changes, distinct breaks in the character and statistical distribution of the EEG. While the exact generating mechanisms of variations in EEG may not be known, characteristics of the observed signal can be used to identify changes in the underlying dynamics.

The theory of $\varepsilon-$complexity as developed by Darkhovsky and Piryatinska provides a method of identifying these regime changes in complex signals such as EEG. The definition of $\varepsilon-$complexity is related to Kolmogorov's definition of the complexity of a sequence. Roughly, Kolmogorov identified the complexity of a discrete sequence as the size of the program, or the amount of information, needed to produce that sequence. The $\varepsilon-$complexity of a continuous function, on the other hand, is the amount of information — in this case, the proportion of the function, needed to reconstruct that function with an absolute error not greater than ε. Darkhovsky and Piryatinska have shown that the $\varepsilon-$complexity of a continuous function can

be identified by two real numbers[5]. These are the $\varepsilon-$complexity coefficients or simply the complexity coefficients of the function. In practical applications, a signal like EEG is given by a some finite set of samples. The theory of $\varepsilon-$complexity is adapted to a discrete setting by considering discrete sequences, such as time series, as restrictions of continuous functions to a uniform grid.

The procedure for estimating the $\varepsilon-$complexity of a function entails the iterative approximation of that function on a smaller set of samples at each iteration. The initial implementation of the procedure for estimating $\varepsilon-$complexity coefficients used piecewise polynomials to reconstruct to approximate the original function. We implement the estimation procedure with an enlarged set of approximation methods — basis splines, cubic splines, and an interpolating subdivision method termed the lifting scheme. We test the estimation of $\varepsilon-$complexity on simulated data using each approximation method. First, we compare the reconstruction accuracy of of each method. Then we test the performance of the complexity coefficients estimated with each approximation method when used to classify groups of related time series. We also compare the computational efficiency of each approximation method. Although the basis spline method has the lowest approximation error across all simulations, we find that low approximation error does not correspond to improvements in classification accuracy. Each approximation method performs similarly in the classification task. However, the cubic spline method is the most computationally efficient, and is used to estimate the complexity coefficients in the applications discussed in Chapters 4 and 5.

Darkhovsky and Piryatinska have shown that for a Hölder class of functions the ε-complexity coefficients capture a linear relationship between the log of the approxi-

mation error and the log of the proportion of the function used in the approximation. They have conjectured that a constant generating mechanism corresponds to constant mean complexity coefficients. In Chapter 4 we use a number of simulations of deterministic and stochastic processes to test whether, on average, the complexity coefficients are constant for a constant Hölder class of functions. For these processes, a single parameter determines both the Hölder class and the fractal dimension of the time series generated by the processes. For these simulations, the slope complexity coefficient B and an estimator of fractal dimension behave simlarly as the parameters of the processes are varied. For three of the four processes tested, we find that the complexity coefficient B is on average, constant for a constant Hölder class of functions.

In the final chapter, the complexity coefficients are applied to the prediction of seizures in epileptic mice. Features from a 4 minute window of EEG are used to predict whether a stimulus applied to epilepsy-prone mice will result in a seizure. A time series of length n can be considered as a vector in an n-dimensional space. For example, 4 minutes of EEG from a single channel is a represented by 2 million observations. A common method of handling such high dimensional data is to map the data to a lower dimensional set of features. For time series like EEG, features are often calculated on a uniform partition or on a sliding window taken over the length of the signal. For an inhomogeneous time series like EEG, this may lead to features being calculated windows with widely varying characteristics. The $\varepsilon-$complexity was designed to identify such variations in a signal. In order to calculate our features on more homogeneous time periods, we use change points in the complexity coefficients to segment the EEG. The average value of features on

these segments are then used as predictors of a seizure response.

Models using this procedure to dynamically segment the EEG signal are compared to several models which use features calculated on uniform partitions of the signal. For the best performing models, we are able to accurately classify seizure outcomes with over 80% accuracy. A model with features calculated on uniform partitions performs as well or better than the models using changes in the complexity coefficients to segment the signal. Due to the small number of trials and some inherent limitations of the data, additional tests would be needed to know if how well the model performance reported here generalizes to other contexts.

Finally, we have created an `R` language package, `ecomplex`, that implements the ε-complexity algorithm. The default settings of this package 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 enable further study or replication of the experiments described in this work.

Chapter 2

Background

2.1 Epsilon-complexity

The term '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 agrees with Kolmogorov's definition of complexity[5]. The complexity, roughly speaking, contained in a continuous function is measured by the proportion of information 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 [5] which shows that for the Hölder class of functions $\varepsilon-$complexity can be characterized by a pair of real numbers. To compute the $\varepsilon-$complexity of in discrete cases a function is given by its restriction to a discrete grid and the theorem is adjusted to the discrete case. We also present the algorithm used to compute the $\varepsilon-$complexity coefficients in practice.

Let $x(t)$ be a continuous function defined on the unit inverval $\mathbb{I} = [0, 1]$. Let $\|\cdot\|$ be a norm on the function and we can assume $\|x(t)\| = 1$, that is, the function has been normalized by taking $x(t)/\|x(t)\|$. Given some family of approximation methods $\mathcal{F}$, let $\hat{x}(\cdot)$ be an approximation of $x(t)$ reconstructed from regularly samples spaced at intervals h. Then the absolute recover error function is defined

$$\delta^{\mathcal{F}}(h) = \inf_{\hat{x}\in\mathcal{F}} \sup |x(t) - \hat{x}(t)| \tag{2.1.1}$$

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

In words, the function $h^*(\varepsilon, \mathcal{F})$ is the minimum grid spacing h needed to approximate $x(t)$ within an absolute error not larger than ε.

Definition 2.1.1. Epsilon-complexity. The number

$$\mathbb{S}_x(\varepsilon, \mathcal{F}) = -\log h_x^*(\varepsilon)\mathcal{F}) \tag{2.1.2}$$

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

We will be referring variously to the Hölder condition or Hölder continuity of a function which we now define:

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

$$|x(t+h) - x(t)| \leq C|h|^\alpha$$

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

Darkhovsky and Pirytinska have show that for almost any individual Hölder class function and given a rich enough family of approximation methods then the epsilon complexity has an approximately linear relationship to $\log \varepsilon$[5]

$$-\log h_x^*(\varepsilon, \mathcal{F}) \approx \mathbb{A} + \mathbb{B} \log \varepsilon. \tag{2.1.3}$$

The relationship characterizes the $\varepsilon-$complexity of a function. An analgous relationship is used for the estimation of ε-complexity for discrete time series. In practial applications any function or time series is acquired as some discrete set of samples, which we will assume to be taken at regular intervals. We consider this

discrete set to be a continuous function restricted to a uniform grid. In this case, $h*(\epsilon, \mathcal{F})$, which we simplify here to $h^*(\epsilon)$, is the proportion of sampled points and $\frac{1}{h^*(\varepsilon)}$ is the number of points needed to approximate some function $x(t)$ within ε. Let n be the total number of points in the initial sampling, then we denote the proportion of points needed for reconstruction with ε

$$S(\cdot) \stackrel{\text{def}}{=} \frac{\lfloor h^*(\varepsilon)n \rfloor}{n}. \tag{2.1.4}$$

Definition 2.1.3. The $\varepsilon-$complexity of a function given represented by a set of values sampled on a uniform grid is $-\log S(\cdot)$.

It follows from defintion 2.1.1 that

Proposition 2.1.4.

$$-\log S(\varepsilon, \mathcal{F}) \to \mathbb{S}_x(\varepsilon, \mathcal{F})$$

as $n \to \infty$.

For the discrete case, the relation analogous to 2.1.3, after substitution of $\log S$ for $\mathbb{S}_x(\varepsilon, \mathcal{F})$ is given by

$$\log \varepsilon \approx A + B \log S. \tag{2.1.5}$$

For practical computations, some set number of grid spacings h corresponding to the proportions S_h are used to estimate the linear relation in 2.1.5. For each spacing h we find the minimum error approximation at that spacing. The parameters of this linear relationship, A, B, are the $\varepsilon-$complexity coefficients or simply the complexity coefficients.

In practical applications, a function $x(t)$ is represented as a discrete set of observations. We take $x(t)$ to be the restriction of a continuous function to a grid with initial spacing 1. The given sampling density is the highest resolution of the function. For some set of integers $\{h\}, h > 1$, the function $x(t)$ is approximated using only observations at every hth sample. The approximations are taken from 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
$A, B \leftarrow \text{lm}(\{\text{ epsilons}_h \} \sim \{S_h\}.)$

2.2 Time series

In practice we observe time series as a set of discrete observations index by time. These time series are considered observations of a stochastic process. As is the case with discrete samples from a deterministic function, we take an observed time series to be a samples of a continuous function taken on a uniform grid.

A time series is a stochastic process, or set of random variables, $X(t)$ or X_t, parametrized by time, $\{X_t : t \in \{1, 2, 3, ..., N\}\}$. 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 uniform samples of a continuous function and we will not always distinguish between these samples and observations of a random process.

A *strictly stationary* stochastic process is one whose distribution independent of time t. We will be using the term to indicate a *weakly stationary* stochastic process, one 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*:

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

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

stochastic process with mean μ is then defined

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

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

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

For a stationary, mean zero stochastic process with $\sigma^2 = 1$ the autocorrelation and autocovariance function are equivalent.

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 $\sigma(h)$

$$\hat{f}(x) = \int_{-\infty}^{\infty} e^{2\pi i \omega x} f(\omega) d\omega \quad . \tag{2.2.1}$$

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 stochastic process with a slowly decaying autocorrelation function is said to exhibit *long-range-dependence*.

Some stochastic processes can be defined by the distribution of their increments. The increments of a stochastic process are defined as $\{X_t - X_{t+h}\}$. The variance of the increments of a stochastic process is called the *variogram*, although the term is more commonly used in reference to random fields – stochastic processes in dimension greater than 1.

Definition 2.2.3. Variogram Let X_t is a zero mean stochastic process with stationary increments. The variogram of X_t is defined

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

Let $N(h)$ be the total all pairs of points in a time series at distance h and $|N(h)|$ be the size of that set. Then the *empirical variogram* is defined

$$\hat{\gamma}(h) = \frac{1}{2|N(h)|}\sum_{i,j\in N(h)} |x_i - x_j|^2.$$

2.3 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: for example, the fractal dimension of a line in $\mathbb{R}^1$ is 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 $\{U_i\}$ approaches 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 $\mathcal{H}^\alpha_\delta$ for any $\delta > 0$ is defined

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

Then

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

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 $\dim_H E$ [6].

For many sets, the Hausdorff dimension is difficult to find and there a number of related measures are used as more tractable alternatives. 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.

There is a general relation between the Hölder-α functions and the fractal dimension of a continuous function on $\mathbb{R}$. The Hölder condition is an upper bound on the box-counting dimension [6].

Proposition 2.3.2. 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 $dim_B f \leq 2 - \alpha$.

We use a variogram estimator of fractal dimension in Chapter 4. This method estimates fractal dimension by taking the log-log regression of the empirical variogram against increments h. This method is recommended by Gneiting in [9]. One justification for estimation method is the relation between the variogram of a function and its fractal dimension.

Proposition 2.3.3. 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 $2 - \alpha$

In addition for Gaussian processes satisifying 2.3.1, the Hölder exponent of its sample paths is almost surely α[9] [12].

2.4 Simulations

In the following two chapters we use a set of simulations to test our implementation of $\varepsilon-$complexity estimation and to explore properties of the ε-complexity coefficients. These simulations include both deterministic functions and stochastic processes. We introduce these processes and functions here and describe some of their properties.

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 above approximately 3.6 but less than 4, the trajectory of the logistic map exhibits chaotic behavior and the graph of the function changes significantly for small perturbations of the r parameter.

The *ARMA* model is a widely used stochastic 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:

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

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 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 or Gaussian 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 $\{\varepsilon_t\} \sim WN(0, \sigma^2)$

$$X_t = \varepsilon_t + a_1\varepsilon_{t-1} + \cdots + a_q\varepsilon_{t-q}. \tag{2.4.1}$$

For our simulations we use parameters that define stationary ARMA(p, q) processes.

A stationary ARMA process has an autocorrelation function $\rho(h)$ that decays at an exponential rate as $|h| \to \infty$ [13]. We include several functions that have a slowly decaying autocorrelation function, a characteristic of processes with long-range dependence. The first of these is the fractionally integrated ARMA process or FARIMA process. 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].

The *Weierstrass* function is 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

$$W_\alpha(x) \quad = \sum_{n=0}^{\infty} b^{-n\alpha} \cos(b^n \pi x) \tag{2.4.2}$$

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 Weierstrass function has the box-counting dimension equal to $2-\alpha$, providing the upper bound for the Hausdorff dimension [6]. That the lower bound of the Hausdorff dimension is also equal to $2-\alpha$ was recently proved by Shen [16].

A variant of the Weierstrass function is *the random-phase Weierstrass* function. The random phase function has the form 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 $2-\alpha$ [10]. See Figure 4.3 in the following chapter for illustrations of both Weierstrass functions.

Brownian motion and *fractional Brownian motion(fBm)* are both Gaussian processes that can be defined in terms of their increment process. Brownian motion has normally, independently distributed increments and fractal dimension $1\frac{1}{2}$. On the other hand, fBm has *dependent* increments with and

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

The case when $\alpha = \frac{1}{2}$ corresponds to Brownian motion. The parameter α corresponds in this case to the fractal dimension of fBm [6]. When $\alpha > \frac{1}{2}$ fBm is a stochastic process with long-range dependence and α corresponds to the Hurst coefficient of the function. When $\alpha < \frac{1}{2}$ sample paths of fBM have a higher fractal

dimension and tend to be less correlated. For fBM, the parameter α determines both the fractal dimension, Hurst parameter and the Hölder exponent of the sample paths.

In the case of fractional Brownian motion the parameter α controls both the fractal dimension and the long-range dependence or Hurst parameter of sample path. 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 [8]. The Cauchy process 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 of approximation methods $\mathcal{F}$ are used. The initial implementation used piecewise polynomials of up to degree 5. For our implementation, we added several additional approximation methods to

procedure for estimating $\varepsilon-$complexity. In Chapter 4 compare the accuracy of these approximation methods on or set of simulations. The $\varepsilon-$complexity coefficients estimated for two groups of simulations using each approximation method. We compare the classification accuracy attained using coefficients estimated with each approximation method.

Splines are piecewise polynomials joined on some set of knots, or intersection points, with 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.

Basis or B-splines are piecewise polynomials. A function is approximated by a linear combination of the basis elements. For general regression splines, an intermediate design matrix is used in the calculation of the final approximation. This means the computational complexity and memory requirements of the method can grow at a greater than the quadratic rate with the number of basis elements used. We used the basis spline regression method as implemented in `R` package `spline` [15][14].

We used an additional spline method limited to only cubic splines. For cubic splines the design matrix is tridiagonal, having entries only on its diagonal and two off-diagonals. An LU type decomposition for tridiagonal matrices allows for linear time cubic spline interpolation [4]. In practice, this means cubic spline interpolation with a fixed number of knots is much less costly in terms of both memory and computational efficiency compared to a B-spline regression that uses intermediate matrices and matrix arithmetic.

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 [18]. To convey the basic approach we consider a 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

$$s_{j+1,2k} = s_{j,k}$$
$$s_{j+1,2k+1} = 1/2(s_{j,k} + s_{j,k+1}).$$

The lifting scheme is similar to wavelet interpolation methods. Unlike wavelet based interpolation, though, the lifting scheme can be extended to interpolating irregularly spaced points since uniform spacing is not a condition for polynomial interpolation.

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 an $n-$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 Methods

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 5, cubic splines and an interpolating subdivision method we refer to as the lifting scheme following Sweldens [17]. A fourth method took the minimum error of these three methods at each stage of the $\varepsilon-$complexity algorithm.

The goal of including additional approximation methods was to find whether drawing on a richer set of reconstruction procedures 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 and stochastic processes. We then test the ability of the complexity coefficients to discriminate between two sets of simulations with similar parameters. Based on these experiments, increasing the accuracy of the approxima-

tion 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. Although the lifting scheme and cubic spline approximation computationally linear in their inputs, the cubic spline implementation was the fastest method by an order of magnitude. Based on both computational efficiency and classification performance we have used the cubic spline method for computing the complexity coefficients in our `R` implementation and in the applied work in the following chapters.

Process	Parameters	Group 1	Group 2
ARMA	p, q	(-0.1, 0.3), (0.2, 0.1)	(0.4, 0.3), (0.1, -0.5)
Cauchy	α, β	0.1, 0.5	0.3, 0.7
FARIMA	p, d, q	(0.1, -0.5),0.3, (0.6, 0.01)	(0.2, -0.4),0.3, (0.4, 0.02)
fBm	α	0.1	0.3
Logistic	r	3.87	3.70
Rand. Weierstrass	α	0.65	0.17

Table 3.1: Simulation parameters for each group.

3.1 Simulations Experiments

For these experiments, we used six simulations methods comprised of both deterministic and stochastic processes. Two groups of simulations were used with two sets of initial parameters for each simulation. The simulations and the parameters

of each group are listed in Table 3.1. For each simulation, 30 samples were generated by drawing a parameter uniformly from a small window centered on the initial parameter values. A single sample observation from each simulation group is shown in Figure 3.1.

For each of the approximation methods we used a simple diagnostic plot to check that the expected log-log relation of approximation errors ε_h to the proportion of points used for reconstruction was roughly linear. 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. The lifting method generated these regression plots and the B-spline and cubic spline methods resulted in similar regression plots.

Figure 3.4 is a plot 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 so the plots show the relative distribution of the functions in the feature space. Although we quantify the ability of the approximation methods to separate each group, the scatter plots show that the methods place each simulation type at similar coordinates in the complexity coefficient space. The scatter plots also show that some methods separated the particular processes better. For example, the cubic spline method separates the two groups generated by ARMA(2,2) processes.

The scatter plot of the simulations also provides some information about the angle at which the simulations are separable. Figure 3.4 shows that 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

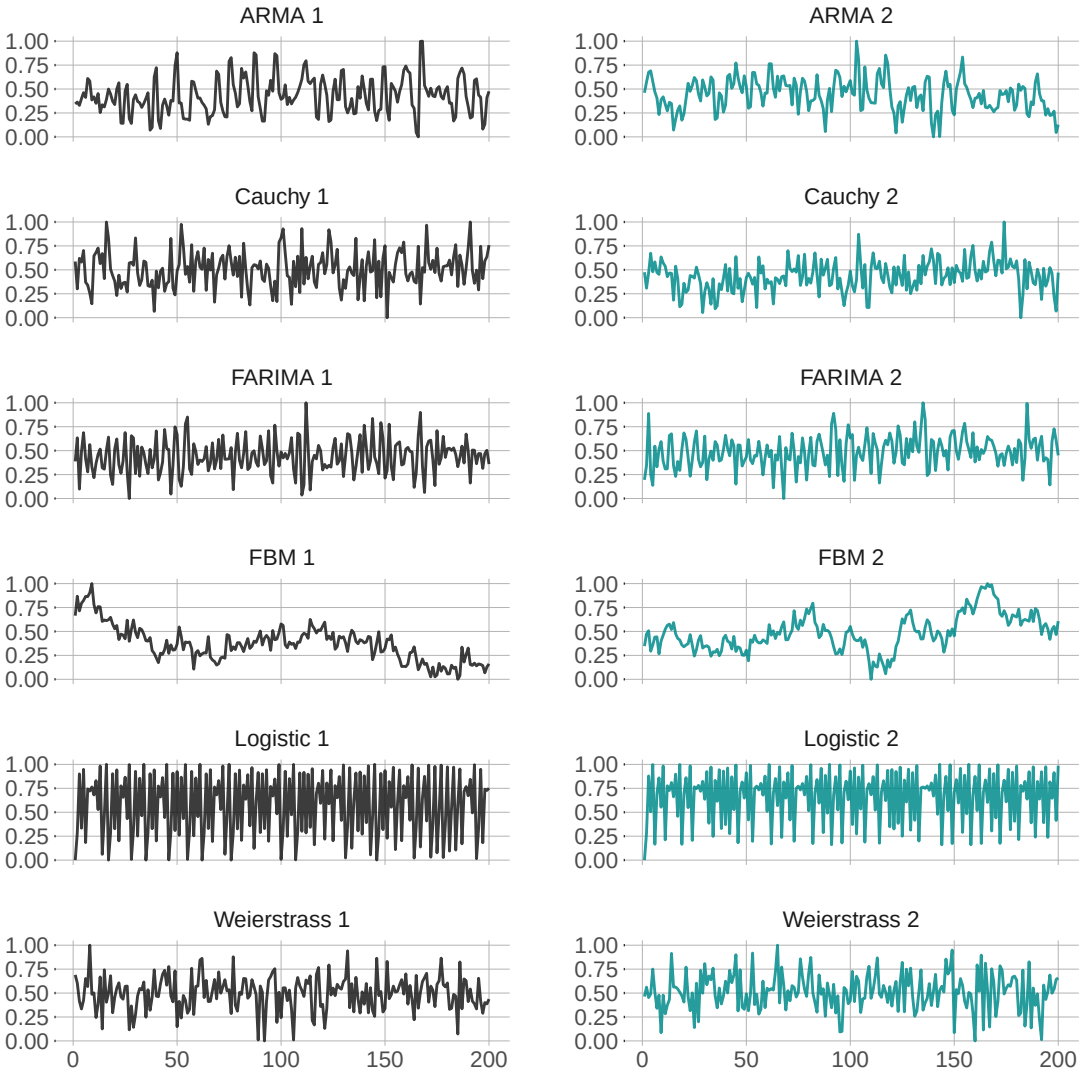

Figure 3.1: Sample functions from each simulation group.

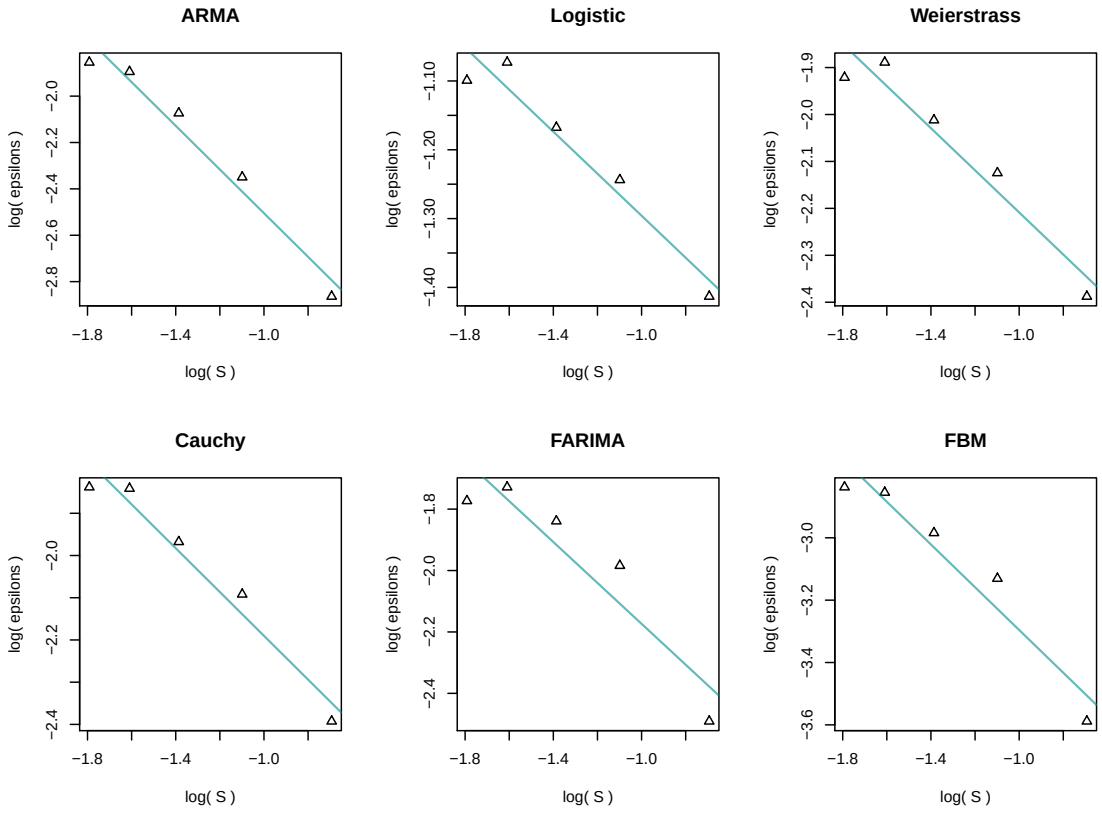

Figure 3.2: Linear regression plots for the lifting scheme.

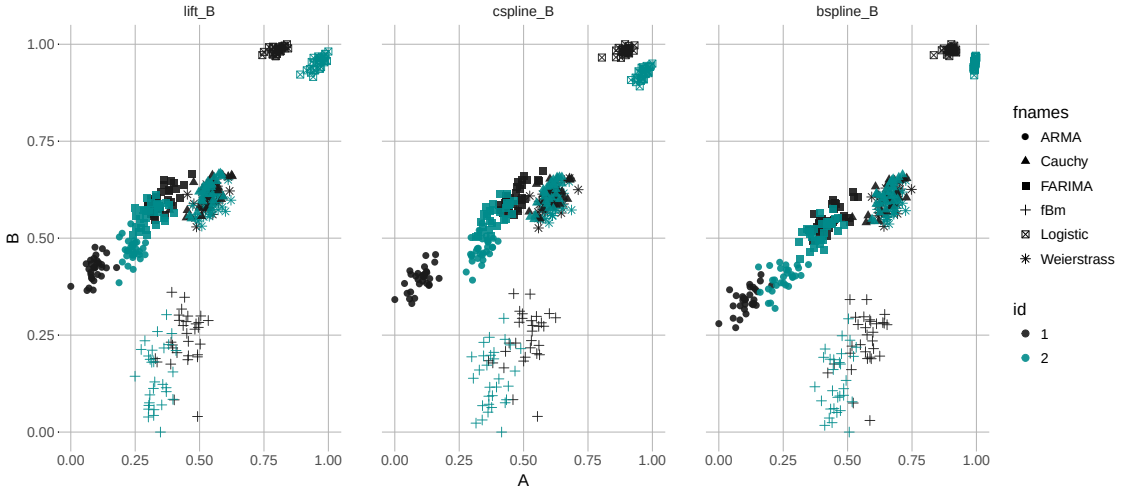

Figure 3.3: The two simulation groups plotted in the complexity coefficient space.

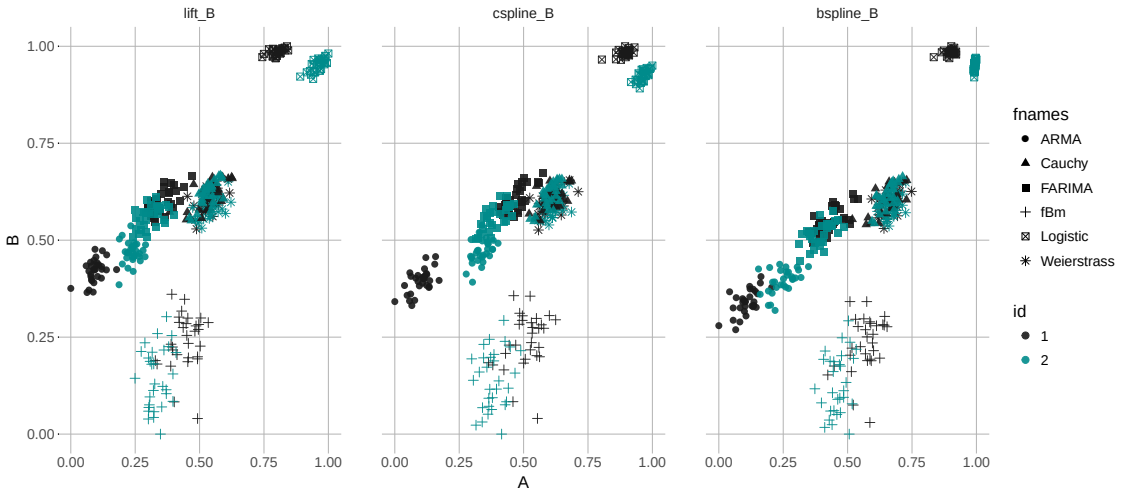

Figure 3.4: Functions from the two simulation groups plotted in the complexity coefficient space.

sets of simulations. For example, both the fBm and ARMA simulations are well separated along the A-axis. On the other hand, the realizations of the logistic map in the upper right hand corner appears separable on the diagonal meaning both complexity coefficients contributed to differentiating the two simulation groups.

Using the two sets of simulations described above, with 30 samples taken from each parameter group, the sum of the errors of each methods was calculated. Table 3.2 show the mean approximation error over the 30 samples. The approximation error was calculated by taking a simple sum of errors at each down sampling level h:

$$\varepsilon_{\mathcal{F}} = \sum_{h} \varepsilon_{h,\mathcal{F}}$$

The B-spline method had the lowest MSE for all processes. Since the B-spline method produced the minimum error for each function, the combined method simply reflects that of the B-spline approximation **??**. The lifting method produced a slightly lower mean error but the cubic and lifting errors are very similar for each simulation.

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 3.2: Mean total approximation error for 30 samples of each simulation.

The set of complexity coefficients as computed by each approximation method were then used to classify the 30 samples from each simulations. The classifier was

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

a random forest with 500 trees with 3 features used to determine branches. The overall out-of-bag classification error is reported in 3.3. The cubic spline method performed equally well or better on 4 of the 6 methods, but the results of all methods are similar.

3.2 Computational Complexity

Classification performance was not 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 than the B-spline method. Both methods are linear in the number of inputs and both are linear or constant in terms of their space complexity, that is, the amount of memory needed to store intermediate computations is a linear or constant function of the size of the inputs.

The execution time of each approximation method estimated using the system time taken to complete approximations on inputs of size 10^7 to 10^{13}. The average computational time for each input size on log-scale is shown in Figure 3.5. For our

implementation, the B-spline method expands the number of knots linearly with the inputs. This leads to the B-spline method becoming infeasible for even moderately large inputs. Although the cubic spline and lifting method are computationally linear, the lifting scheme was implemented entirely in the `R` language. The cubic spline method is called from `R` but most calculations are written in the computationally efficient `C` language. Although it is not clear in Figure 3.5, the cubic spline method was an order of magnitude faster than the lifting scheme.

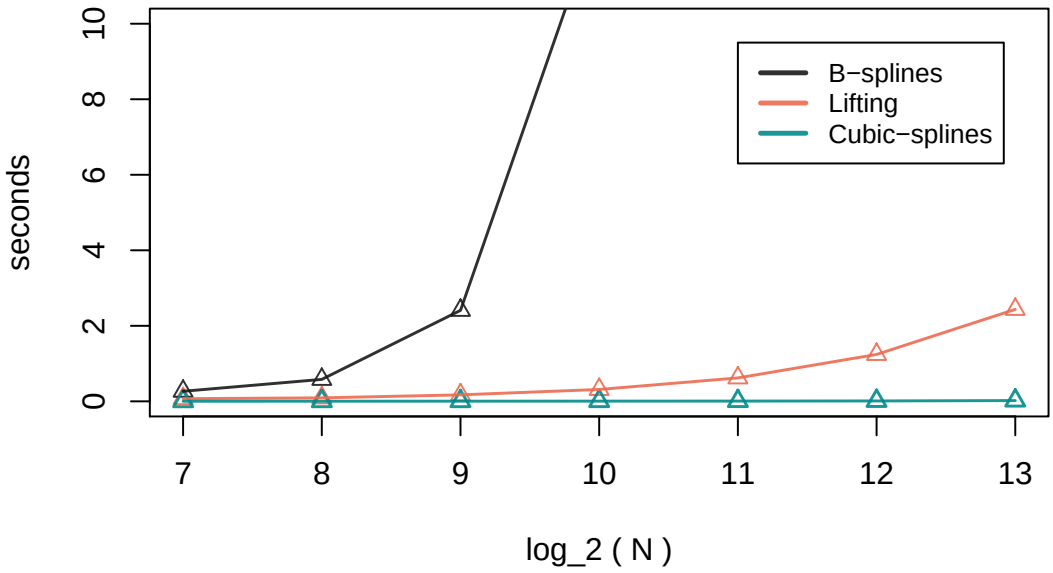

Figure 3.5: Computational time as a function of the size of the input.

Our primary goal of the preceding test was to determine if enlarging the set of approximations would lead to improved performance of the complexity coefficients. We assumed that the enlarged set would allow for improved approximation error and

a better estimation of the complexity coefficients. Here we measured performance by the ability of the complexity coefficients to discriminate between closely related simulations. Lower approximation errors did not correspond to better classification accuracy, however. The cubic spline method was the most computationally efficient and performed as well or better on the classification test as the other methods. Based on these results, the cubic spline approximation was used for the applied work in Chapters 4 and 5.

Chapter 4

Interpreting the Complexity Coefficients

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 how the complexity coefficients change as both the fixed Hölder-α class of a simulation and its fractal dimension is varied. For these simulations a single parameter controls both the Hölder class and the fractal dimension of the simulations. We find that the slope complexity coefficient B and a fractal estimator change linearly with this parameter for all but one of the simulations.

We begin by looking at 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 examples highlight some basic properties of the complexity coefficients.

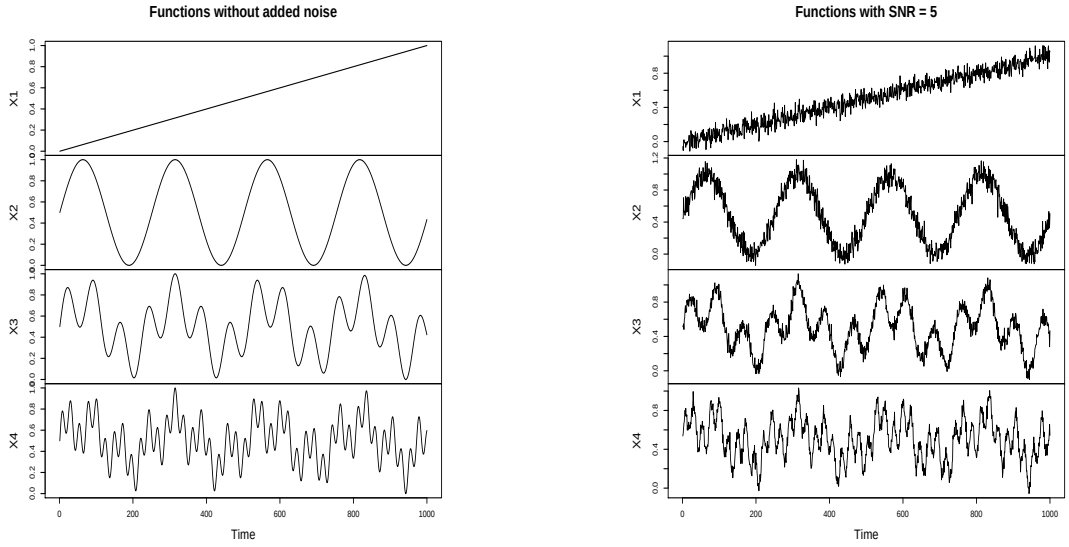

Figure 4.1: Linear and sinusoidal functions with and without added noise.

4.1 The Complexity Coefficients

The epsilon complexity coefficients A and B along with a variogram-based fractal dimension estimator which we denote $\hat{D}$, were estimated on the four functions depicted in in Figure 4.1. In addition to the original functions, A, B and $\hat{D}$ where computed on the functions with two levels of added noise. Figure 4.2 shows the $log - log$ regression of the errors ε_h on the fraction of samples kept S_h. The complexity coefficients for these fits are given in Table 4.1. While these functions do not reveal much about the behavior of the complexity coefficients for more complicated time series, the results demonstrate properties of the complexity coefficients and the sensitivity of the estimators to added noise.

The parameters of the linear regression in Figure 4.2 determine the complexity coefficients A, B. The $x-$axis is the log of S_h, the proportion of points used to approximate the original function at each step. The finest scale approximation is

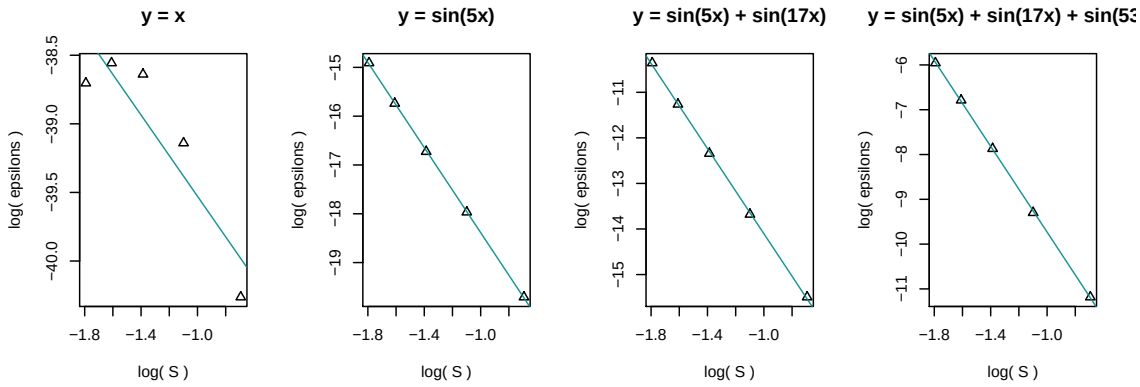

Figure 4.2: The log-log linear regression of approximation errors against the proportion of sampled S_h for simple linear and sinusoidal functions.

Without Noise	A	B	$\hat{D}$
x	-41.45	-1.92	1.00
$\sin 5x$	-22.71	-4.33	1.00
$\sin 5x + \sin 17x$	-18.71	-4.61	1.00
$\sin 5x + \sin 17x + \sin 53x$	-14.54	-4.81	1.01
SNR = 20	A	B	$\hat{D}$
x	-4.04	-0.57	1.97
$\sin 5x$	-3.90	-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 estimates for linear and sinusoidal functions.

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 at this point correspond to accurate

approximations and less variability at the finest scale of the sampled function. In theory, the approximation error at zero should be an exact approximation of the function. Since we are computing the log of the approximation error ε, $\log(\varepsilon) \to -\infty$ as $\varepsilon \to 0$. For the three functions in Figure 4.1 the overall approximation error is low. Larger intercept should correspond to larger errors and, as expected, the intercept value A increases with increased function complexity and higher levels of noise.

The complexity coefficient B measures the rate of change of the approximation error as a function is approximated using fewer samples. The coefficient B increases rapidly as noise is added. Similarly, the fractal dimension estimator approaches its theoretical maximum 2 as Gaussian noise is added to the functions. In order to compare these results to estimates on pure Gaussian noise, we estimated the complexity coefficients and fractal dimension on 50 samples Gaussian noise. The mean estimate of the complexity coefficient B was -0.54, close to the estimates in Table 4.1 for functions with added noise. For both the fractal dimension estimator and the complexity coefficient B, relatively small amounts of added noise generate estimates similar to that of Gaussian noise.

	A	B	$\hat{D}$
White Noise	-2.84	-0.54	2

Table 4.2: Complexity coefficients and fractal dimension estimates for Gaussian noise.

4.2 Hölder Class and Fractal Dimension

Several of the simulations used in Chapter 3 to test the performance of approximation methods have parameters that determine both the Hölder class of a function and its fractal dimension. Our experiments show that for most of these simulations, the complexity slope coefficient B behaves similar to the variogram estimator of fractal dimension $\hat{D}$. For three of the simulations, the Weierstrass function, fractional Brownian motion(fBm), and the Cauchy process, the median values of the B and $\hat{D}$ change linearly with the parameters determining both fractal dimension and the Hölder exponent of the function or sample paths of the simulations. For these simulations, the complexity intercept coefficient A changed non-linearly as the parameter determining the Hölder class of a function changed. A different pattern was observed for the random-phase Weierstrass function. For samples of the random-phase Weierstrass function, the values for the complexity coefficient B and $\hat{D}$ were similar to white noise while the complexity coefficient A changed linearly with the parameter deterimining the Hölder exponent of the function.

The Weierstrass function is a self-similar deterministic function whose Hölder exponent and fractal dimension when written

$$W_\alpha(x) \quad = \sum_{n=0}^{\infty} b^{-n\alpha} \cos(b^n \pi x) \tag{4.2.1}$$

is determined by the single parameter α. The Hölder exponent is equal to α while the functions fractal dimension is $D = 2 - \alpha$. Since the function is deterministic, the complexity coefficients and fractal estimator were calculated on a single example of length 1000 for 20 parameters of α in the interval $(0, 1)$. The results in Figure 4.4

show that the complexity coefficient B and fractal dimension change linearly with α. The fractal dimension estimator tracks the theoretical fractal dimension closely, ranging from 2 to 1. The complexity coefficient B ranges -0.6 to -1.4. For both estimators, fractal dimension corresponding to 2 coincides with estimates close to those estimated for Gaussian noise.

The random-phase Weierstrass function adds a random phase to the periodic component of the Weierstrass function but its Hölder exponent and fractal dimension are determined by the α parameter in the same manner as for the deterministic Weierstrass function.The Weierstrass and random-phase Weierstrass functions are illustrated in Figure 4.3 for three different parameter values $\alpha = \{0.20, 0.42, 0.80\}$. Unlike the other results reported in this Chapter, neither the fractal estimator nor the complexity coefficient B varied with α. For each parameter the complexity coefficients and fractal dimension were estimated on 50 samples of length 1000. Figure 4.5 shows that the complexity coefficient B and fractal estimator $\hat{D}$ range around values associated with estimates for Gaussian noise. 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.

Fractional Brownian motion also has a single parameter α which is equal to the Hölder exponent and determines the fractal dimension of sample paths as $D = 2-\alpha$. For fBM the parameter α, sometimes denoted H, is also the Hurst parameter. The results of the estimators computed on 50 samples are shown in Figure 4.7 and are similar to the estimates for the deterministic Weierstrass function with the median of both the complexity coefficient B and $\hat{D}$ changing linearly with the parameter α. As was the case for the deterministic Weierstrass function, the complexity coefficient

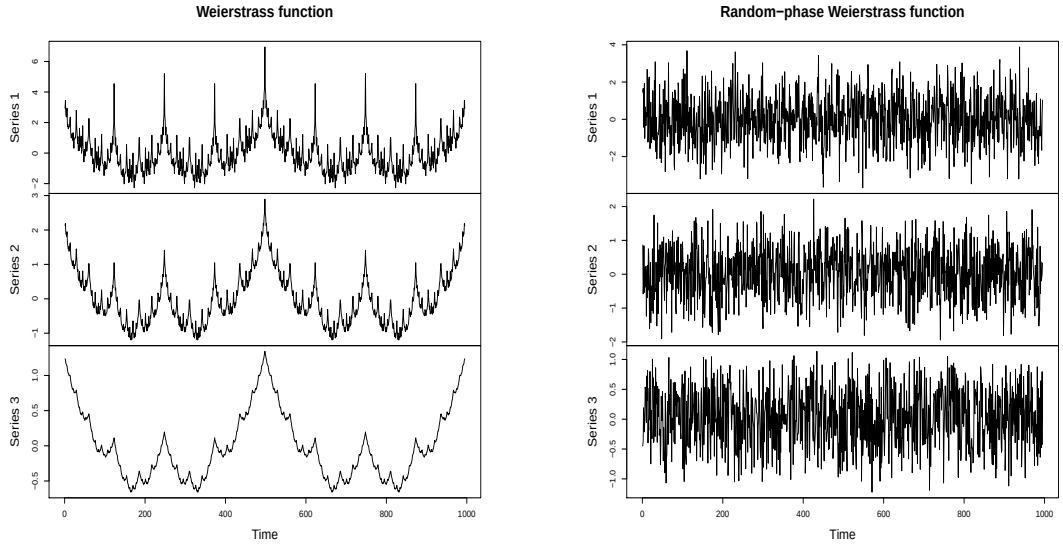

Figure 4.3: The Weierstrass and random-phase Weierstrass function for $\alpha = \{0.20, 0.42, 0.80\}$

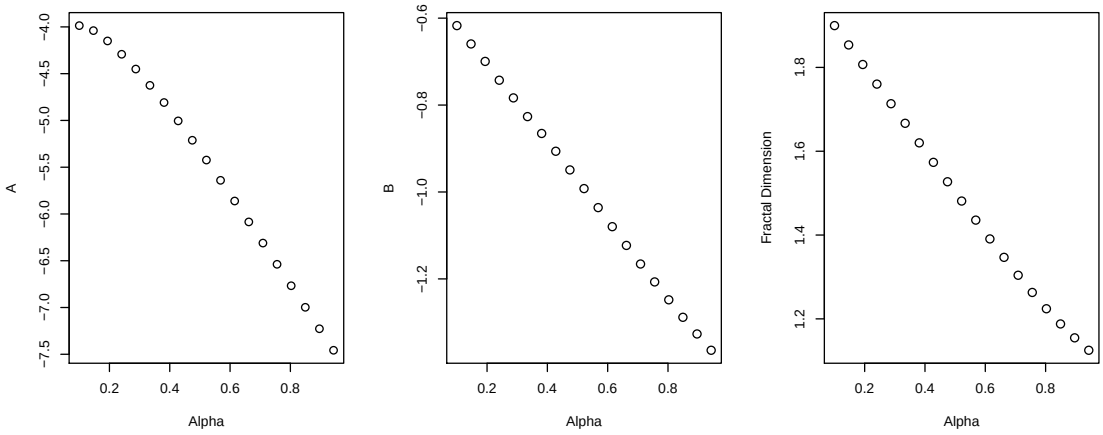

Figure 4.4: Complexity coefficients and fractal dimension for values of the Weierstrass α parameter.

A also changes non-linearly with α.

The Cauchy process was the final simulation we tested. The Cauchy process has

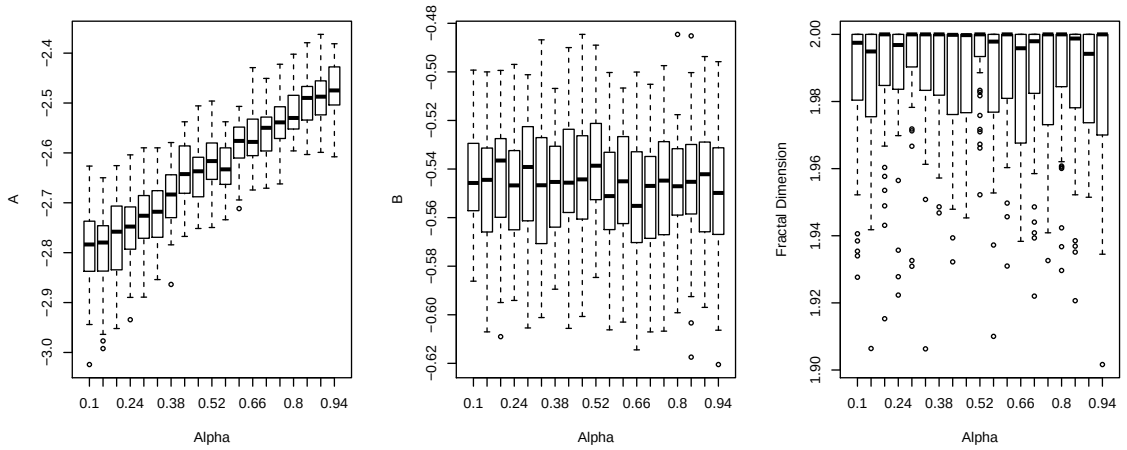

Figure 4.5: Complexity coefficients and fractal dimension for values of the random-phase Weierstrass α parameters.

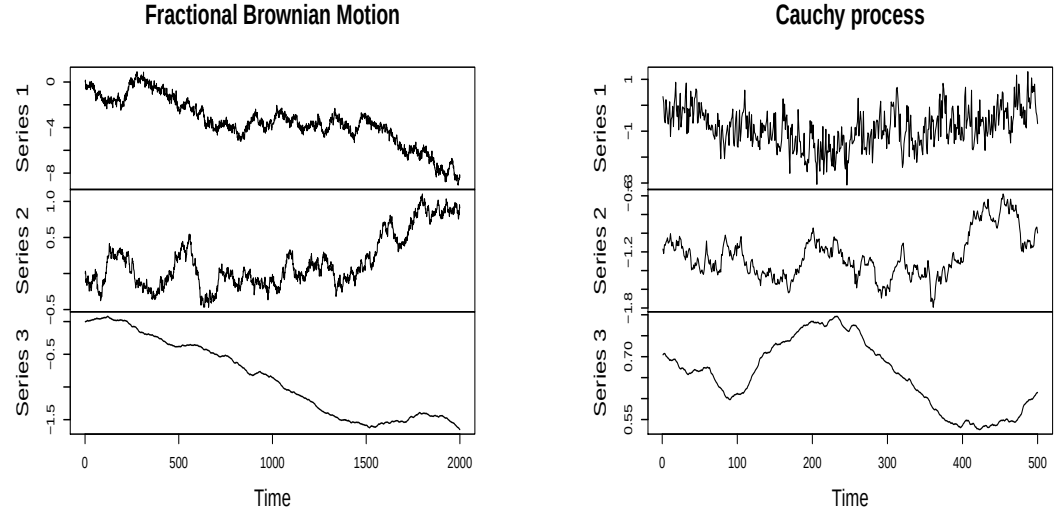

Figure 4.6: Fractional Brownian motion and the Cauchy process with $\alpha = \{0.20, 0.42, 0.80\}$ for a constant β parameter.

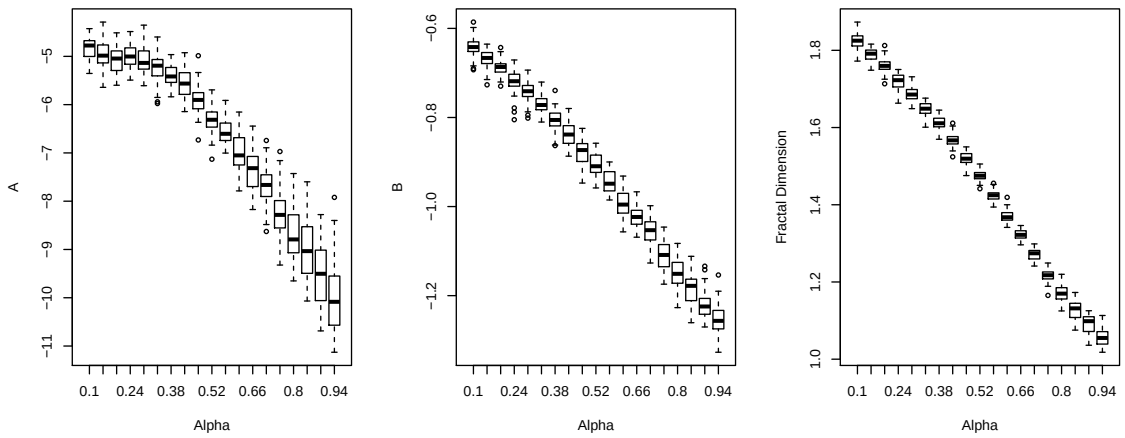

Figure 4.7: Complexity coefficients and fractal dimension for values of the α parameter of fractional Brownian motion.

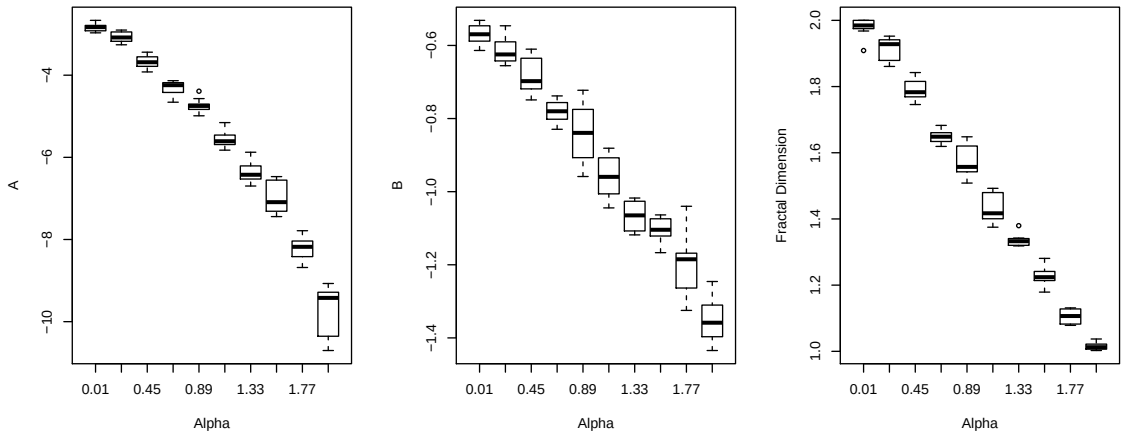

Figure 4.8: Complexity coefficients and fractal dimension for various α parameters for a constant β parameter of the Cauchy process.

two parameters: the parameter α determines fractal dimension and the parameter β determines the Hurst coefficient or long-range dependence of the sample paths. Simulations were generated on a grid of the parameters α, β for $0 < \alpha < 2$ and $0 < \beta < 1.5$ with 30 simulations generated for each point on the parameter grid. At a constant parameter β the median of the complexity coefficients and fractal estimator again show a linear relation to the α parameter, as shown in Figure 4.8. This relation holds even as β, the Hurst parameter, is varied. The mean value of each of the estimators for for a fixed value β and α is shown in Figure 4.9. Variation in the Hurst parameter has little affect on the either the complexity coefficients or the fractal dimension estimator.

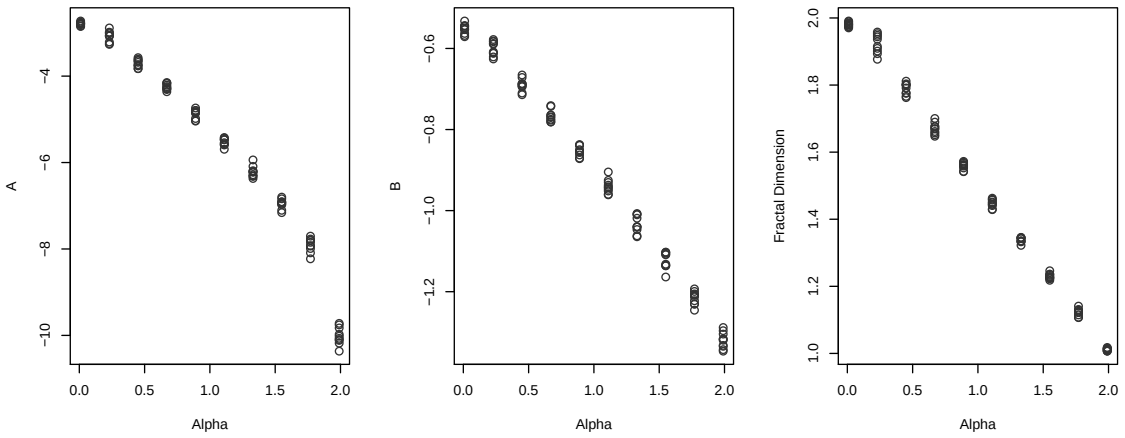

Figure 4.9: Mean value of the complexity coefficients and fractal dimension for each α value of the Cauchy process.

Figure 4.10 shows another view of the complexity coefficients plotted against changes the Cauchy process parameter β. The complexity coefficient B and fractal dimension $\hat{D}$, preserve the grid of the parameter space — for a fixed β, the parameter

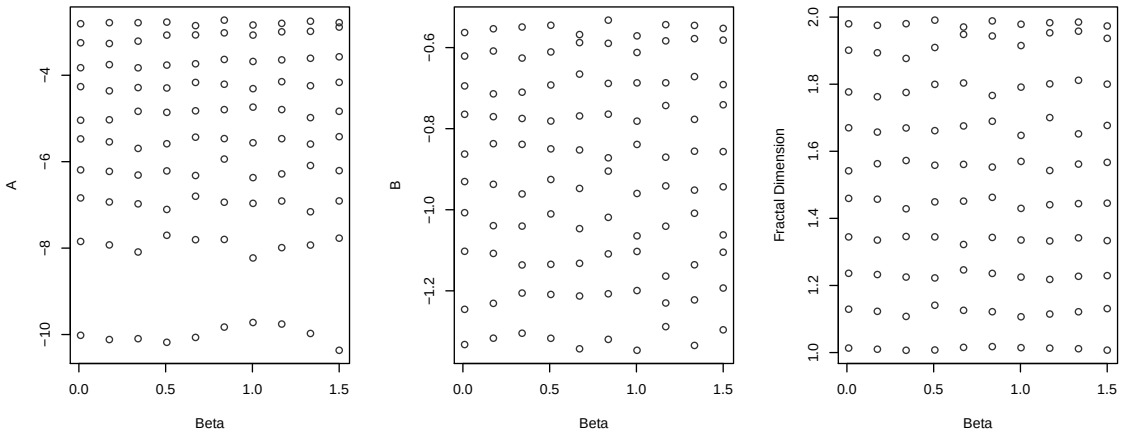

Figure 4.10: The mean value of the complexity coefficients and fractal dimension for each β value of the Cauchy process.

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, property of a function.

For all simulations but the random-phase Weierstrass function the, a similar pattern was observed for the complexity coefficients and the fractal dimension estimator. Both the fractal dimension estimator $\hat{D}$ and the complexity coefficient B changed linearly with the parameter determining the Hölder exponent and fractal dimension of the simulations. 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 whose fractal dimension and Hölder exponent are determined by a single paramater. A comparison of the complexity coefficient B and fractal estimators on a wider variety of time series might show whether the relation the

ε-complexity coefficients and fractal dimension hold for a wider range conditions.

Chapter 5

Seizure Prediction

Electroencephalograms(EEGs) capture the electromagnetic potential of aligned neurons firing in unison. Although a number of correlations between EEG and physiological phenomena have been found the relation between EEG and the dynamics of local neural circuitry are not well-understood [3]. One difficulty is relating in vitro behavior of individual neural cells or small clusters of cells with the more complex large scale variations recorded by the EEG signal. In the case of the epileptic neuropathology, the challenge is to relate better known local cellular mechanisms to the spread synchronized neuronal firing associated with epileptic seizures. Making this connection between the larger scale electromagnetic phenomena recorded by the EEG signal and local neural dynamics is not the goal of this Chapter. Instead, this was a preliminary study to determine whether a seizure response to a locally applied stimulus could be predicted based EEG recorded prior to the stimulus. An additional aim of this chapter is to compare classification models based on the segmentation of EEG features based on the ε-complexity coefficients.

Using a 4 minute window of EEG taken from an epilepsy prone mouse we predict whether a stimulus will induce a seizure response. A small set of spectral

and non-linear features are calculated on six EEG channels. Change points in the $\varepsilon-$complexity coefficients are used to segment the features and the results are compared to other segmentation methods. We also compare the features and channels predictive of seizure outcomes with the variation in feature distributions of trials with seizure and non-seizure outcomes. Because predictions are made on a channel-by-channel basis, we also able to compare how features predictive of seizure outcomes vary by channel and brain region.

5.1 Introduction

EEG are non-stationary signals marked by transient waveforms and apparent regime changes over relatively short periods of times. A common method of analyzing EEG is to compute some set of features on the raw signal EEG signal and use these features as input to one or more classifiers. These features often combine a decomposition of the signal, for example, spectral, wavelet, principal component(PCA) or independent component(ICA) decompositions, with some linear or non-linear features[1]. Signal decomposition and feature selection serve to reduce the high-dimensional EEG signal to a more interpretable feature set by averaging feature values over a sliding window. 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.

Here, we use a set of spectral features and non-linear features to predict seizure outcomes. However, instead of segmenting the signal on uniform partitions the segmentation is based on changes in the $\varepsilon-$complexity coefficients. The $\varepsilon-$complexity

coefficients were designed to capture discontinuities or regime changes in a signal. By partitioning the feature space at changes in the complexity coefficients we hope to capture features are more homogeneous segments.

We compare classification performance based on several segmentation schemes, using both the complexity coefficients and a set of uniform break points to partition the features. For each of these models, the prediction of seizure outcomes is based on features computed on single EEG channels. 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.

5.2 EEG Data

The EEG data were gathered from 4 mice with a genetic mutation in the voltage-gated sodium channel gene Scn1a. A similar mutation is responsible for the Dravet syndrome and the mutation results in early onset epileptic seizures [11]. The mice were equipped with 4 intracranial(iEEG) electrodes —intracranial sensors placed directly on the brain cortex — along with 2 local field potential (LFP) electrodes located in the thalamus. The mice were genetically modified to produce a light sensitive protein, an opsin. This allows for direct stimulation of neurons with a laser pulse train. The stimulus is designed to induce a seizure in the mouse.

Data was gathered from 13 distinct time periods during which the stimulus was applied between one to four times. The 4-minute segments preceding the stimulus was used to predict outcomes and there were 26 trials in total. The result of each trial was were coded as either a seizure or no seizure and we refer to a seizure as

a positive response. 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 iEEG and LFP signals 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.

5.3 EEG Features

The features consisted of windowed spectral band power estimates, variance and two non-linear 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

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

The frequency bins corresponding to the remaining the theta, alpha, beta and gamma frequency bands are $4 - 8Hz, 8 - 12Hz, 12 - 30Hz, 30 - 100Hz$. Relative band power, band power divided by total power, was used as the final feature.

A smaller set of non-linear features was chosen based on their ability to classify the simulation groups used in Chapter 3. Features that did not increase the performance of a random forest classifier were omitted from the final feature set. The initial set of features were sample entropy, and permutation entropy, spectral entropy, a Hurst parameter estimator, wavelet variance, a fractal dimension estimator and signal variance. Of these, variance, spectral entropy and the Hurst parameter

were used in our final classifier.

Spectral entropy is a measure of the distribution of power in the spectrum of a signal.

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

We have defined the Hurst parameter, a measure of long-range dependence, in Chapter 2. 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 to segment the features but were not included as features in the final set of classifiers. All features were computed on 2 second non-overlapping windows.

5.4 Classification Procedure

The classification procedure takes place in four main steps. For each channel, the features are computed on 2 second intervals. The resulting set of features is segmented based on changes in the complexity coefficients. The weighted averages of

the features on these segments is then used as training set for the classifier. Classifier performance was evaluated using 5-fold cross-validation and positive responses were even spread across the 5 test sets.

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. Using the held-out test set, compute the class probability of each segment of the test trial using the trained classifier.
8. For each trial the sum of the prediction probabilities of each segment, weighted by segment length, determine 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[7]. 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 [2].

The effect of individual features on the classification outcomes can be less transparent for non-linear classifiers like random forest when compared to regression-based classifiers. However, the importance of the variable in discriminating classes can be recovered from the classification trees. We report the mean variable importance of the random forest classifiers determined by the mean decrease in the Gini index across all branches [2]. The Gini index measures node purity, or the homogeneity of a class in a given node resulting from some division of the feature space. Informally, for binary classification variable importance measures how well a feature divides two classes into homogeneous cells.

5.5 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

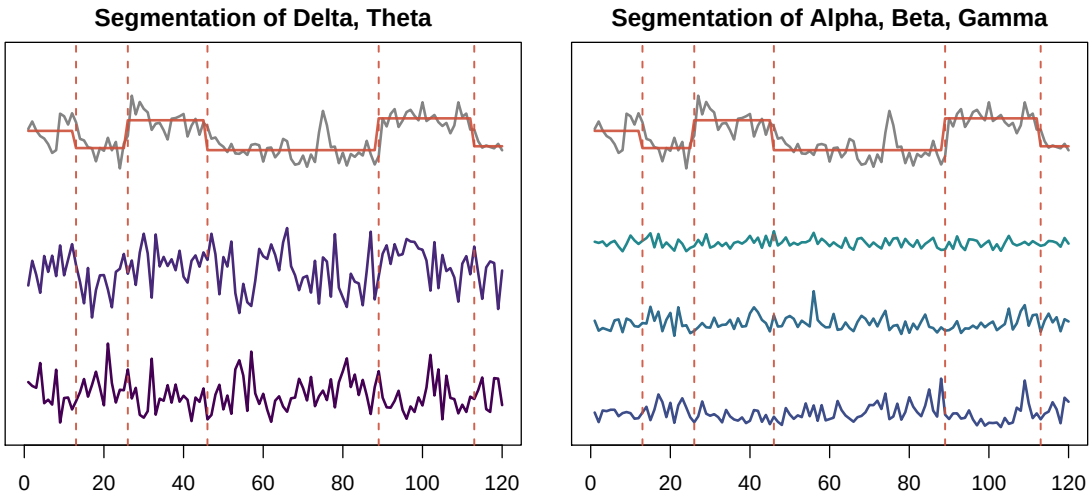

Figure 5.1: Spectral features segmented on changes in the B complexity coefficient.

of true positives to total positives

$$\frac{\text{True positives}}{\text{Total Positives}}$$

and *specificity* is the total of true negatives to total negatives:

$$\frac{\text{True Negatives}}{\text{Total Negatives}}.$$

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 5.5 for an example ROC curve from two of the classifiers tested. The 'curve' is the plot of the sensitivity against the false positive rate or 1−specificity and a higher AUC corresponds to a

better performing classifier. An AUC of 1 indicates the correct classification of all observations.

A baseline classification model was built using 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. This baseline model classifies the over represented class, the negative or no-seizure responses, with 93% accuracy but classification of a seizure response was no better than random.

	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 label the models $A, B, A+B$, where the label corresponds to the complexity coefficient or coefficients whose change points determined the partition. For example, the $A+B$ model partitions features based on the union of the change points of both complexity coefficients A and B. For three other models we partitioned features into regular segments of length $8, 15$ or 30. We refer to these models by the number of partitions.

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 50 bootstrap samples and the confidence intervals give the range of the middle 95% of the model samples. In general, all models performed best on the LFP channels, here labeled channel 1 and 2. The best performing model was model

8 followed by the B model.

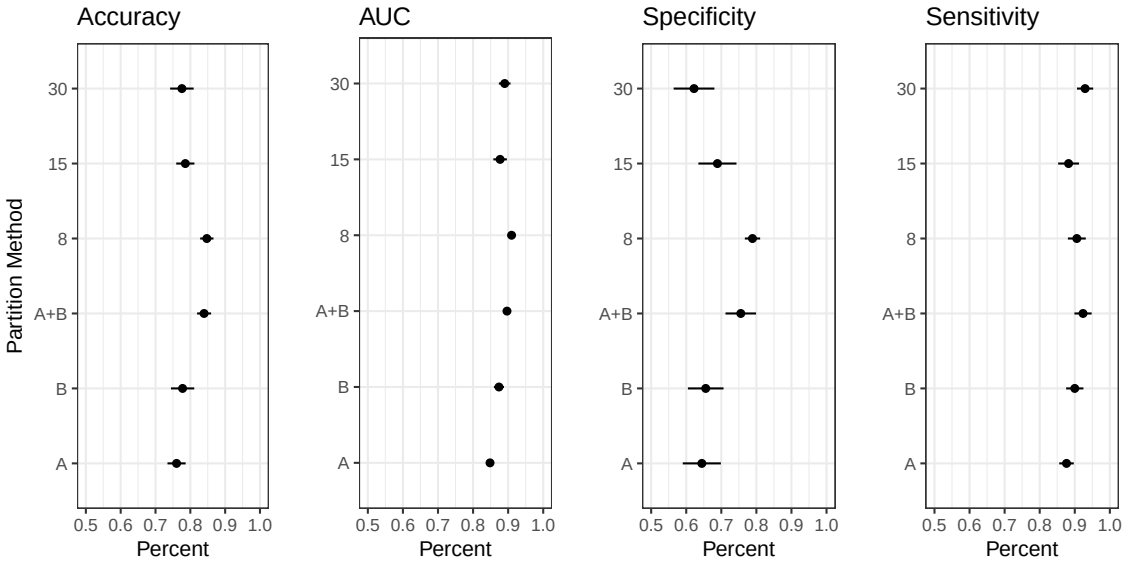

Figure 5.2: Classification diagnostic values for each classifier using features LFP channel 1.

The mean and 95% confidence intervals for the performance measures for each classifier using features computed on channel are shown in Figure 5.2. All methods classified non-response trials well but the two better performing models — model 8 and model B — had relatively high accuracy in classifying seizure responses: 76% for model B and 79% for model 8.

The balanced accuracy of all models for each channel is reported 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 iEEG channels 3-6. Accurate classification of seizures tailed off sharply for all iEEG channels as seen in Table 5.3. In particular, specificity was greater than 70% for only a few models, all of which used channels 1 or 2. The ROC and smoothed

	CH1	(sd1)	CH2	(sd2)	CH3	(sd3)	CH4	(sd4)	CH5	(sd5)	CH6	(sd6)
A	0.82	(0.04)	0.77	(0.04)	0.69	(0.02)	0.71	(0.04)	0.57	(0.04)	0.63	(0.03)
B	0.83	(0.03)	0.77	(0.04)	0.71	(0.03)	0.71	(0.03)	0.61	(0.05)	0.65	(0.06)
A+B	0.81	(0.03)	0.73	(0.03)	0.70	(0.04)	0.72	(0.03)	0.57	(0.09)	0.63	(0.07)
8	0.83	(0.05)	0.82	(0.04)	0.74	(0.04)	0.72	(0.02)	0.75	(0.05)	0.74	(0.06)
15	0.82	(0.05)	0.84	(0.04)	0.73	(0.04)	0.73	(0.02)	0.77	(0.05)	0.78	(0.06)
30	0.79	(0.07)	0.77	(0.05)	0.76	(0.04)	0.76	(0.03)	0.73	(0.08)	0.75	(0.04)

Table 5.2: Mean and s.d. of balanced accuracy for all models.

	CH1	(sd1)	CH2	(sd2)	CH3	(sd3)	CH4	(sd4)	CH5	(sd5)	CH6	(sd6)
A	0.73	(0.06)	0.60	(0.08)	0.46	(0.04)	0.52	(0.05)	0.26	(0.07)	0.33	(0.05)
B	0.76	(0.07)	0.59	(0.07)	0.51	(0.06)	0.53	(0.05)	0.37	(0.09)	0.38	(0.11)
A+B	0.69	(0.07)	0.51	(0.06)	0.49	(0.08)	0.53	(0.05)	0.23	(0.17)	0.32	(0.12)
8	0.77	(0.11)	0.70	(0.07)	0.59	(0.07)	0.54	(0.04)	0.62	(0.11)	0.57	(0.11)
15	0.71	(0.09)	0.73	(0.08)	0.57	(0.06)	0.56	(0.00)	0.68	(0.12)	0.62	(0.12)
30	0.64	(0.14)	0.60	(0.09)	0.60	(0.08)	0.61	(0.06)	0.59	(0.15)	0.58	(0.09)

Table 5.3: Mean and s.d. of specificity for all models.

ROC curves in Figure 5.5 for are similar for both the model $A + B$ and model 8.

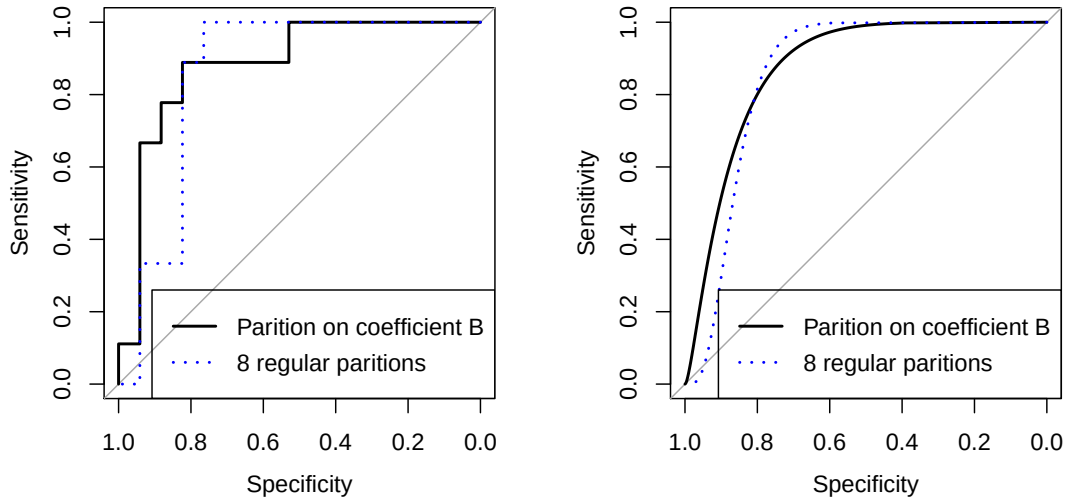

Figure 5.3: Best ROC curves for classifiers B and 8.

5.6 Feature Importance

The classification results show that seizures could with a relatively high degree of accuracy for some model and channel combinations. Apart from the baseline model, each of these models used features calculated on individual channels. This allows us to look at the combination of feature and channel associated with accurate predictions. Here we compare the variable importance of the more successful models to the difference in feature distributions between trials with a seizure and non-seizure response. Random forests build multiple decision trees that divide up the feature space in complex ways and the final classification is tallied from the votes of individual trees. This means there may be now clear relation between feature

distribution and random forest variable importance. Nevertheless, we find that features associated with the best predictors were associated with clear differences in distributions between trial classes.

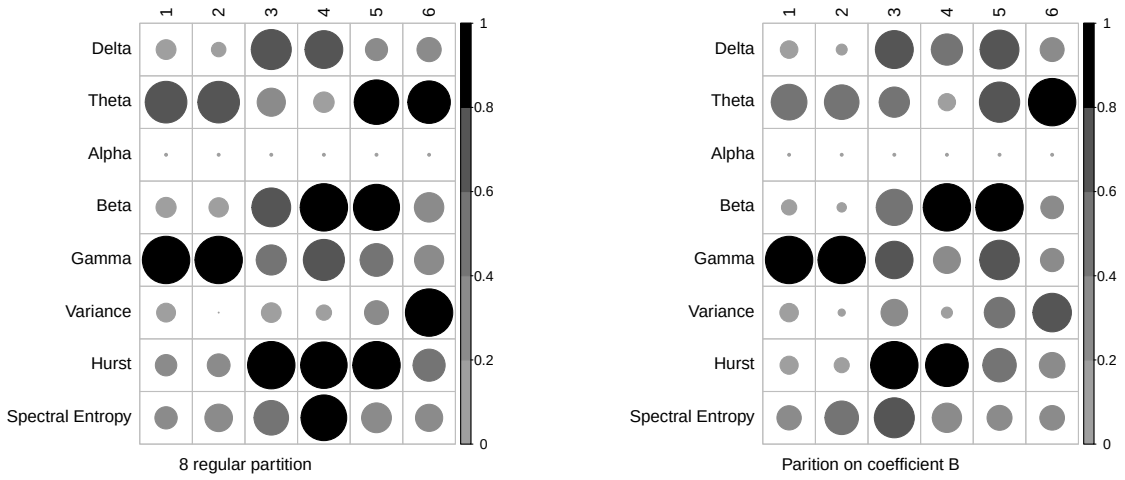

Figure 5.4: Normalized variable importance for each partition method and channel.

In Figure 5.4 we show the variable importance for the two best performing models 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. However, no model was able to accurately predict seizures using these channels in isolation. The maximum

specificty among all models trained on these channels was 68% and for the two models depicted in 5.4 this number was 59%.

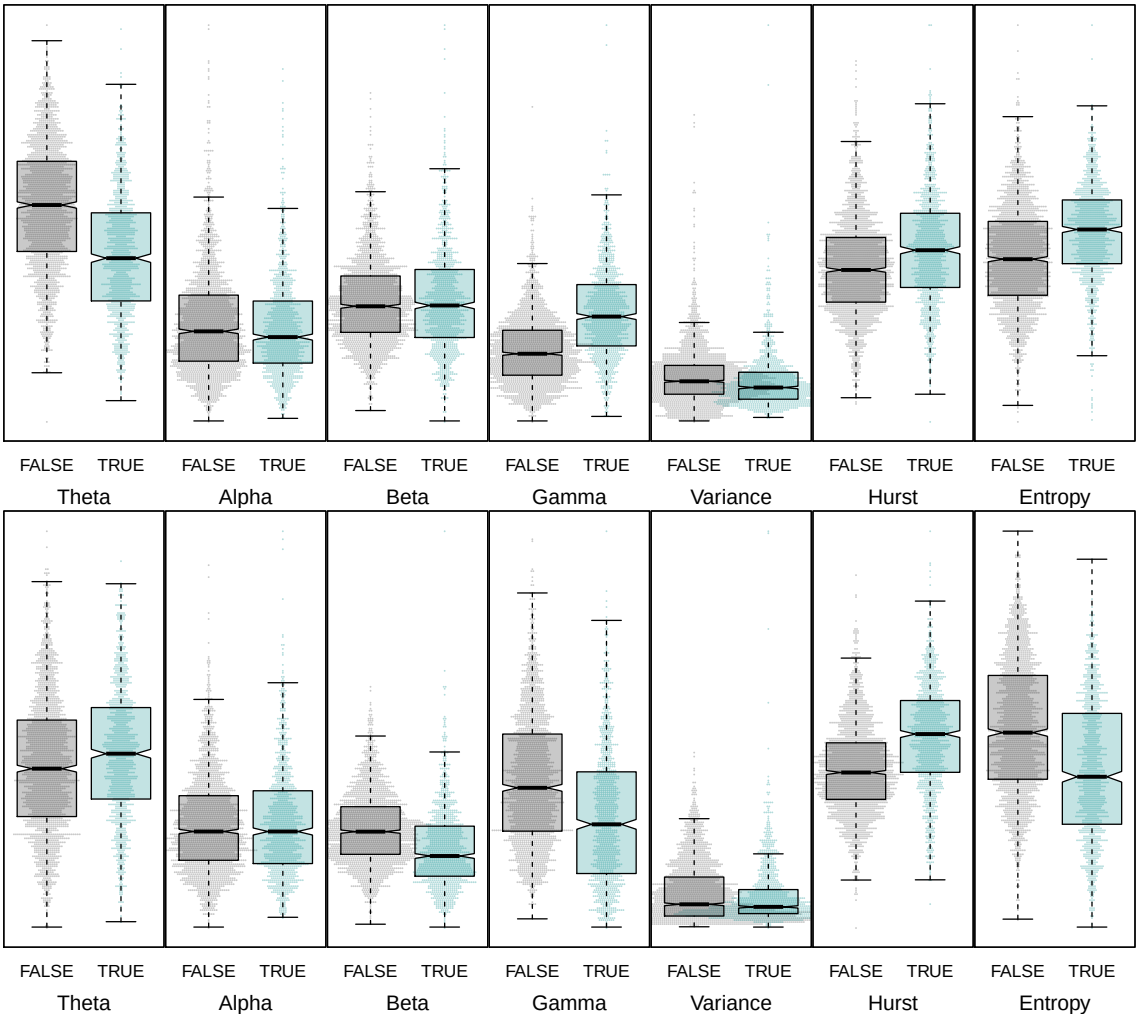

Figure 5.5: Feature distribution for channels 1 and 3.

Variable importance as measured by the random forest classifiers was also reflected in distributional differences in the features. The box plots 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 the unpaired Wilcox rank-sum test, a nonparametric test of the difference between two distributions. 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: P-values fo Wilcox rank sum test for difference of distribution

5.7 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. There were 4 mice used in the experiments but the trials resulting in seizures came from a single mouse. Data from other mice would be required to know whether the features associated with seizures held more generally. In addition, most of the seizures came from

stimuli applied within two trials. Several of the seizures responses, then, occurred after a previous seizure. Therefore, features found predictive of a seizure may be conflated with features that are a result of a seizure. Due to the small number of trials, we did not have a hold-out data set. The lab from which produced this set of data is collecting additional data. Testing the predicitive models described here against new data would better indicate whether the predictive power generalize well.

The segmentation model based on the complexity coefficients performed as the models with regular partitions for some channels. However, the regular partition models tended to perform more consistently across channels. The models built using channels the iEEG electrodes, channels 3-6, the models using features partitioned at regular intervals performed better than the models partitioned on change points in $\varepsilon-$complexity. The algorithm used to detect changes points in the complexity coefficients was sensitive to the density at which the features were computed. For example, computing the features, including the complexity coefficients on an overlapping set of windows would increase the density at which features were calculated. A finer resolution may improve the detection of change points the complexity coefficients. On average, the uniform partitions created divided the trials into more segments. It may also simply be that a somewhat finer segmentation than that given by the complexity coefficients is what improved the consistency of the models with uniform partitions. Tests of this segmentation 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] Turkey N Alotaiby, Saleh A Alshebeili, Tariq Alshawi, Ishtiaq Ahmad, and Fathi E Abd El-Samie, *Eeg seizure detection and prediction algorithms: a survey*, EURASIP Journal on Advances in Signal Processing **2014** (2014), no. 1, 183.

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

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

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

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

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

[7] Jerome Friedman, Trevor Hastie, and Robert Tibshirani, *The elements of statistical learning*, vol. 1, Springer series in statistics New York, 2001.

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

[9] 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.

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

[11] 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.

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

[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] Weixiao Shen, *Hausdorff dimension of the graphs of the classical weierstrass functions*, arXiv preprint arXiv:1505.03986 (2015).

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

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

[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.