DISSERTATION

ANALYSIS OF STRUCTURED DATA AND BIG DATA WITH APPLICATION TO NEUROSCIENCE

Submitted by

Ela Sienkiewicz

Department of Statistics

In partial fulfillment of the requirements

For the Degree of Doctor of Philosophy

Colorado State University

Fort Collins, Colorado

Summer 2015

Doctoral Committee:

Advisor: Haonan Wang

Mary Meyer
F. Jay Breidt
Stephen Hayne

ABSTRACT

ANALYSIS OF STRUCTURED DATA AND BIG DATA

WITH APPLICATION TO NEUROSCIENCE

Neuroscience research leads to a remarkable set of statistical challenges, many of them due to the complexity of the brain, its intricate structure and dynamical, non-linear, often non-stationary behavior. The challenge of modeling brain functions is magnified by the quantity and inhomogeneity of data produced by scientific studies. Here we show how to take advantage of advances in distributed and parallel computing to mitigate memory and processor constraints and develop models of neural components and neural dynamics.

First we consider the problem of function estimation and selection in time-series functional dynamical models. Our motivating application is on the point-process spiking activities recorded from the brain, which poses major computational challenges for modeling even moderately complex brain functionality. We present a big data approach to the identification of sparse nonlinear dynamical systems using generalized Volterra kernels and their approximation using B-spline basis functions. The performance of the proposed method is demonstrated in experimental studies.

We also consider a set of unlabeled tree objects with topological and geometric properties. For each data object, two curve representations are developed to characterize its topological and geometric aspects. We further define the notions of topological and geometric medians as well as quantiles based on both representations. In addition, we take a novel approach to define the Pareto medians and quantiles through a multi-objective optimization problem. In particular, we study two different objective functions which measure the topological variation and geometric variation respectively. Analytical solutions are provided for topological and geometric medians and

quantiles, and in general, for Pareto medians and quantiles the genetic algorithm is implemented. The proposed methods are applied to analyze a data set of pyramidal neurons.

ACKNOWLEDGEMENTS

First and foremost I would like to express my deepest appreciation and gratitude to my advisor Professor Haonan Wang for his invaluable mentorship and inspiration throughout our research collaboration. I am especially grateful for his limitless patience and advice in the process of writing this dissertation.

I would also like to thank my committee members Professors Mary Meyer and Jay Breidt for the important and insightful feedback they provided to me for my dissertation work, and for their guidance and advice during my years at Colorado State University.

Next, I would like to thank my outside committee member Dr. Stephen Hayne from the Department of Computer Information Systems and Dr. Christos Papadopoulos from the Department of Computer Science for providing me with access to their computer labs, without which parts of this dissertation could not have been written. In addition, I would like to thank Dr. Hayne for the valuable comments and suggestions he provided on this dissertation.

I would also like to thank Dr. Dong Song from the Center for Neural Engineering at the University of Southern California for his indispensable contributions to Chapter 2 of this dissertation, as well as his professional advice and valuable comments.

I would also like to express my gratitude to the Department of Statistics at Colorado State University for all of their support and for creating a friendly, stimulating environment.

Lastly, I would like to thank all of my family and friends for supporting me through these challenging five years.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Introduction

1.1 Background

Statistical analysis has been an integral part of neuroscience ever since the discovery of the stochastic nature of neural responses. Scientists were using electrodes to detect neural action potentials, also referred to as neural spikes, in the brains of animals as early as the 1950s. At the time, the dominating theories of neural activity were deterministic, but experiments showed variability in the response when the stimulus was held constant, and random spikes in the absence of any stimuli. See Stein [1965] for a theoretical analysis of that variability, and Harrison et al. [2013] for a recent review of statistical modeling of variability and its counterpart synchrony, detected in various parts of the nervous system.

Neuroscience research generates a great deal of public interest, from the dynamics of learning and memory formation to the recognition and diagnosis of brain diseases like autism, depression and Alzheimer's disease. These and many other goals were behind the recent BRAIN initiative (Brain Research through Advancing Innovative Neurotechnologies), a governmental program funding and encouraging interdisciplinary effort to provide measurable progress in brain science. Statistics plays an important part in that effort, which was clearly and eloquently discussed in an ASA white paper [ASA, 2014]. The primary statistical tasks will naturally include building models and quantifying uncertainty. Challenges are numerous, especially with regard to accounting for complex, structured data, spatio-temporal effects, missing or incomplete data, variations between species, genders, brain regions, etc. Because of the sheer quantity of data produced by neuroscience research, the analysis often enters the realm of *big data*, where many classical methods become impractical due to memory or time constraints; see Fan et al. [2014] for a recent review of big data challenges in statistics.

The brain is a source of *big data* due to its massive structure comprised of billions of neural cells and trillions of connections, but in reality, the brain's structure is not static. In fact, it is continuously evolving with changes driven by complex dynamic behavior. The behavior and the structure, two important aspects of the brain, are interconnected with one impacting the other. So far we are lacking sufficient quantities of data that would combine these characteristics, but the potential for future research is enormous.

The dynamic aspects of the brain include the exchange of electric signals (action potentials) between neural cells. Neurons typically consist of a soma, an axon and one or more types of dendrites; see Figure 3.1 for an example of a neuron and its components. Dendritic structures of neurons are responsible for receiving signals from other cells, whereas an axon carries the signal outside of the cell. The history of a single neuron's firings is often encoded as a spike train. Analysis of neural spike trains is at the core of much contemporary neuroscience research. Researchers model output neural spikes based on the inputs to understand the behavior of neurons in various regions of the brain, e.g., a frequently modeled brain region is the hippocampus, an area involved in learning and memory formation. The dependency between inputs and outputs is non-linear, temporal, and often non-stationary. In fact, the spatio-temporal patterns of neural firings are believed to be a part of information encoding in the brain.

There are many approaches to the analysis of spike trains, each with numerous examples in the literature, e.g., Bayesian analysis [Archer et al., 2013], firing intensity estimation with a Cox proportional hazard model [Brown et al., 2002, Plesser and Gerstner, 2000], Artificial Neural Networks based model [Maass, 1997], and a logistic model with generalized Volterra kernels [Song et al., 2007, 2013]. Each model has been useful in answering important research questions, but each also comes with challenges and shortcomings. One of the major challenges is dimensionality. Usually, output neural spikes depend not just on current but also past input observations, and system memory is large relative to the duration of a spike. These assumptions, together with

complex interactions between inputs, can lead to the estimation of thousands of parameters even for a system with a relatively small number of inputs. One of the contributions of this dissertation is presenting a strategy for fitting sparse dynamical models to neural data using appropriate optimization methods, approximations, and parallel computing.

The brain can be viewed as a complex network with a *small world* architecture. The connections between cells are not observed directly, but can be modeled based on the analysis of spike train firings. A cell's propensity to form these connections has also been of interest to scientists. The more complex the dendritic structure, the more signals a cell can potentially receive and process [Mel, 1999]. As early as the nineteenth century, scientists observed a correlation between the diversity of neural dendritic structures and the complexity of the entire organism; see Ramón y Cajal [1995] for a detailed study of the nervous systems of vertebrates. The heterogeneity of dendritic structures has also been associated with learning capabilities. Some early methods to estimate the density of dendritic arborizations included the counting of dendritic segments and the calculation of a *fractal dimension* [Fiala and Harris, 1999]. Dendritic structures can be modeled as trees, often as binary trees. Trees are complex, extremely non-Euclidean structures, and classical statistical tools are often not sufficient to describe and quantify them. Due to the heterogeneity encountered in the real biological tree-structured data, it is often desirable to compare not just the mean structures of samples, but also objects representing different quantiles, e.g., 10th or 90th, of the corresponding distributions. For such comparisons to be possible, a notion of a quantile of tree-structured data is required. A space of complex data lacks a natural order, just like multivariate and functional data. In the analysis of multivariate and functional data, the *simplicial depth* [Liu, 1990] and norm minimization [Serfling, 2002, Walter, 2011] have been applied to define analogs of the order statistic. The analysis of tree objects has been, until the last decade, focused on classification trees [Banks and Constantine, 1998, Phillips and Warnow, 1996] and phylogenetic trees [Billera et al., 2001, Nye et al., 2011]. For such tree objects, a mean

and a median tree are of particular interest, and popular approaches were built on the notion of the Fréchet median [Fréchet, 1948]. Only recently, with advances in medical imaging technologies, biological tree-structured data have become available to statistical analysis. In a study motivated by the analysis of the structure of brain arteries, Wang and Marron [2007] provided the first attempt to define a median of a set of trees with both topological and geometric properties. The approaches in that and many other papers [Aydın et al., 2009, Wang et al., 2012] were based on labeling of tree nodes for correspondence between branches. Such nodal correspondence may be natural in some applications. For instance, nodes in classification trees and edges in phylogenetic trees are uniquely identified. This is not always the case in biological data, as with plant roots, brain arteries and dendritic arborizations. Various choices of correspondence can potentially lead to different research conclusions, and artificially imposed labeling may result in biased estimates. For example, Skwerer et al. [2014] observed that labeling in trees serving as a model of human brain arteries can lead to a degenerate, star-tree median. For trees where labeling is not natural, it can be beneficial to consider quantiles that do not depend on assigned labels. The analysis of unlabeled tree-structured data has not been addressed before now and is a major contribution of this dissertation.

1.2 Scope

In this dissertation, we develop statistical models motivated by two data sets originating from neuroscience research, specifically, a large set of dynamic data consisting of spike trains recorded in animal studies, and a set of complex, structured objects extracted from digital reconstructions of neurons. Both data sets pose a *big data* challenge.

In Chapter 2, we propose a numerical implementation of sparse functional dynamical models with a large data set. Our motivating example comes in the form of spike train data collected from the hippocampus of a rodent. The data consists of 90-minute recordings of activities from eight

hippocampal neural clusters, and our goal is to develop a *Multiple-Input Single-Output* (MISO) system, which models an output neuron as a function of its current and past inputs. Our work builds on ideas proposed by Song et al. [2007, 2013], and it employs a probit model with a Volterra series. A Volterra series is a generic solution to nonlinear differential equations, and, similar to a Taylor series expansion, approximates a function with a desired accuracy on a compact set. Unfortunately, for most large data sets, only the first order model is computationally feasible, even if the model fit is not considered adequate. This is in part due to the massively large number of parameters in the model. Our contribution is a feasible and efficient computational strategy to fit second- and higher-order Volterra models using the maximum likelihood principle and its regularized analog. Our proposed strategy uses a smaller memory footprint and makes model fit feasible on a multi-core computer without paying a penalty for distributed computations.

In Chapter 3, we focus our attention on complex, non-Euclidean data, specifically tree-structured data, that can be characterized as highly *heterogeneous*. Most work in the literature focuses on the topological properties of trees based on pre-selected labeling systems. Here, we take both topological and geometric attributes of trees into account, and intend to develop statistical tools for unlabeled trees. Our goal is to define quantiles of such objects, which can be potentially used for comparison between samples of tree-structured data. In particular, we develop a concept of a quantile of tree-structured objects through a multi-objective optimization problem. We include criteria that quantify topological and geometric variations as objective functions in the optimization problem. We also develop a methodology to identify empirical quantiles for a sample of tree-structured data. Identification of empirical quantiles is a high-dimensional, combinatorial optimization problem. Such a problem can often be solved efficiently with a genetic algorithm, which has been shown to exhibit an *implicit parallelism*; see Goldberg [1989]. In Chapter 4, we discuss in details the genetic algorithm used in estimating empirical quantiles of unlabeled tree objects.

Finally, in Chapter 5, we discuss future work in the direction of one-sample and two-sample hypothesis testing of unlabeled tree-structured data, based on mixtures of topological tree distributions and simulations.

CHAPTER 2

Sparse Functional Dynamical Models[1]

2.1 Introduction

Dynamical system is time-dependent, i.e., its output is determined not only by its current input, but also its past inputs. Its complexity is often enhanced by the nonlinear interactions within its internal structure, in such a way that the resulting system is different from the sum of their parts. It has been shown that despite their complex nature, dynamical systems can often be described with a few low order ordinary differential equations; see Fuchs [2013] for a more recent review.

The first documented study of a nonlinear dynamical system involved two interacting species, predators and prey, and was conducted independently by Volterra and Lotka in the 1910s and 1920s. The interaction was modeled through nonlinear differential equations [Volterra, 1930]. The Volterra series has been demonstrated to be a general solution to such nonlinear differential equation [Flake, 1963, Karmakar, 1979, Volterra, 1930]. It represents the output of the system as a series of convolutions of its inputs and is often referred to as a Taylor series with memory, or a nonparametric representation of a nonlinear system. For a system with a single input $x(t)$ and a single output $y(t)$, the series can be expressed in the following form

$$y(t) = \kappa_0 + \sum_{p=1}^{\infty} \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} \kappa_p(\tau_1, \ldots, \tau_p) x(t-\tau_1) \cdots x(t-\tau_p) d\tau_1 \cdots d\tau_p \tag{2.1}$$

where κ_p is known as the p-th order Volterra kernel. In practice, all Volterra kernels have to be estimated in the process of system identification.

The Volterra theory of analytic functions was advanced by Wiener in the applications to electronic communications in the 1920s [Wiener, 1958], which resulted in a Wiener theory of nonlinear

[1] This Chapter is based on the early draft of paper "Sparse Functional Dynamic Models - A Big Data Approach."

systems. The key idea of the Wiener series was to orthogonalize the Volterra functionals, and thus facilitate model estimation. By the 1950s both the Volterra and Wiener models became accepted and widely-used methods of the analysis of complex dynamical systems with finite memory and time-invariance. Just like the Taylor series for nonlinear functions, the Volterra/Wiener series can suffer from convergence issues, but in 1910, Fréchet showed that the Weierstrass theorem can be extended to Volterra functions, essentially proving that any Volterra/Wiener system can be approximated on a compact set with a desired accuracy [Fréchet et al., 1936].

The nonlinear dynamical systems have been identified in areas as diverse as social science, natural science and engineering. In economics and psychology the system dynamics are related to human behavior and interactions. Engineering processes, although usually described by a set of known rules or laws, can exhibit unknown nonlinearities, especially under abnormal circumstances, e.g., in control systems. The systems analyzed by Wiener and Schetzen [Schetzen, 1981, Wiener, 1958] considered inputs like voltage, current, velocity, stress, temperature. A network monitoring system is concerned with traffic characteristics: volume, temporal and spatial patterns of messages, etc. Nonlinear interactions between inputs are modeled to describe and detect system failures, network intrusions, resource misuse, etc [Cannady, 1998, Debar et al., 1992].

Biological systems present a much greater challenge, since the rules governing their behavior are unknown or only partially understood. They are frequently modeled as black-boxes or gray-boxes with multiple inputs and multiple outputs. There are abundant examples of applying the Volterra or Wiener functional expansion in modeling physiological functions: cardiovascular, respiratory, renal [Marmarelis, 1993], functions of auditory peripherals and visual cortex [Wray and Green, 1994], human pupillary control, medical imaging [Korenberg and Hunter, 1996] and many more.

The particularly challenging examples of biological systems come from neuroscience, with the human brain considered to be the most complex nonlinear dynamical system of all. It is also a

good illustration of a system composed of subcomponents, in which understanding the functionality of simplest building blocks is not enough to understand and model the entire system. The behavior of neurons, each one consisting of a soma, an axon, and dendrites, has been fairly well understood at the individual level. Existing parametric models (e.g., Hodgkin-Huxley, Fitzhugh-Nugamo, Hindmarsh-Rose), consisting of sets of differential equations, can successfully describe the nonlinear dynamics of the membrane voltage at single neuron level [Fuchs, 2013]. However, these models cannot easily scale up to the level of *neural ensembles*, which are groups of interconnected and functionally similar neurons, due to numerous unknown biological mechanisms and processes within the neuronal networks. On the other hand, nonlinear dynamical model in the form of the Volterra/Wiener series, models the neural ensembles directly from their input/output data. It does not rely on the complete knowledge of the system and thus provides an appealing approach for modeling large-scale, complex nervous systems.

Specifically in this paper, we model the nonlinear dynamics of neural ensembles using spiking activities. Spikes, also termed action potentials, are brief electrical pulses generated in the soma and propagating along axons. Neurons receive spike inputs from other neurons through synapses and then transform these inputs into their spike outputs. This input-output transformation is determined by numerous mechanism/processes such as short-term synaptic plasticity, dendritic/-somatic integration, neuron morphology, and voltage-dependent ionic channels. Since spikes have stereotypical shapes, they can be simplified as point-process signals carrying information with the timings of the pulses. It is widely accepted that, in the brain, information is encoded in the spatio-temporal patterns of spikes. Thus, characterization of the point-process input-output properties of neurons using spiking activities is a crucial step for understanding how the neurons, as well as the neuronal networks, transmit and process information; see Song et al. [2007, 2013]. Within a neural ensemble, this procedure is essentially the identification of the functional connectivity between neurons.

In the previous studies (Song et al., 2007, 2009), Volterra kernel models were combined with generalized linear models to characterize the input-output properties of neurons. To facilitate model estimation, Volterra kernels are expanded with basis functions, e.g., Laguerre basis or B-spline basis. Model coefficients are estimated with a penalized likelihood method (Song et al., 2013). However, this model estimation imposes major computational challenge. The first-order Volterra series has been shown to be inadequate to capture the system dynamics [Song et al., 2007]. The second-order series, with even a moderate number of inputs and basis functions, requires an estimation of thousands of parameters. A large number of observations is critical, to account for sparsity in the inputs and to avoid over-smoothing of the basis functions; see Marmarelis [1993] and remarks by Korenberg and Hunter [1996].

With a large n and a large p, the system identification faces a *big data* challenge, which is characterized by both hardware and software issues; see Fan et al. [2014] for a recent review. The data may not fit in the memory of a single computer, and some statistical methods, if they can be adjusted to accommodate distributed data and computing, just take too long to be computationally feasible. As a result, none of the cited papers was successful in providing a viable and reproducible method of regularized estimation of Volterra kernels of higher orders. A regularization is key to achieve a functional sparsity at both local and global levels, necessitated by the sparsity in neural connectivity [Tu et al., 2012]. Song et al. [2013] provided a nice study of sparsity by comparing the effects of Laguerre basis and B-spline basis. Both types are adequate in representing the global sparsity, but only the B-spline basis successfully model the local sparsity. It is worth mentioning that, to achieve sparsity, Song et al. [2009] used the forward step-wise model selection and Song et al. [2013] implemented regularization based method, e.g., LASSO [Tibshirani, 1996].

Aside from the nonparametric modeling using Volterra kernels, other approaches are also available though they are not considered in this paper:

- Cascade or block structure modeling [Korenberg and Hunter, 1986]
- Basis representation of orthogonalized Wiener kernels [Marmarelis, 1993].
- Artificial neural networks (ANN) [Wray and Green, 1994], the authors show a correspondence between the Volterra kernel model and a neural network. Recurrent neural networks are a competing approach to the analysis of the nonlinear dynamical systems, and can perform both a knowledge based, and a black-box type system identification.

In this paper we address the aforementioned computational challenges by applying big data techniques to perform a system identification using a sparse basis expansion of the Volterra kernels of a second or higher order. The rest of this paper is organized as follows. Section 2.2 contains an overview of the map-reduce technique. Section 2.3 presents our proposed dynamical Multiple Input Single Output (MISO) model. Section 2.4 describes the computational issues and Section 3.2 covers the analysis of real data.

2.2 Map-reduce and Big Data

2.2.1 Computational Complexity Analysis

Given the disparity between our perception of time and the small scale of inter-neural communication, the analysis of brain functions is bound to face computational challenges. A short 15 minute recording of brain activity in an animal performing stimuli-induced actions can lead to close to a million observations if broken into 1 millisecond (ms) intervals, a timeframe in which a neural spike can be detected. The number of the input connections doesn't have to be large (in fact the neural network is believed to exhibit a *small-world* architecture: few direct connections, but only a few steps away from any other neuron [He, 2005]), but due to complex interactions between inputs, even a small number of them can lead to thousands of parameters in the model. A design matrix for the first-order Volterra system with several inputs and one second system

memory requires just close to 2 GB of storage, but for the second-order system that number grows to over 200 GB, and involves close to thirty thousand parameters. Many matrix operations are "embarrassingly parallel", so an infrastructure is needed to divide the matrix into chunks that can fit in the memory of a single computer, execute tasks in parallel and assemble the results. The division of tasks and reassembly of results are the foundation of the *map-reduce* approach [Dean and Ghemawat, 2008], an algorithmic concept that can be implemented on a single multicore system or in a cluster of servers. There exists a variety of multicore programming options, depending on the platform and the language of choice. The most common cluster implementations come from Apache Software Foundation projects, i.e., Hadoop and Spark, and are briefly described in the next section. Although 200 GB may not be considered big data by practitioners dealing with terabytes or more, it is still enough to cause computational issues in the statistical analysis that requires hundreds (and sometimes thousands) of such large matrix operations. Just the generation of a 200 GB matrix took five hours of a single-threaded processing (which could be parallelized). We found two viable computational approaches to the analysis we describe in details in the following sections. The first approach relies on placing the large matrix on the distributed file system, and using the distributed map-reduce framework, e.g., Hadoop or Spark, to perform computations. The cluster should be large enough to offset the penalty of creating and accessing the matrix, and of distributed computing. Without such a large cluster, one can still find a feasible in-memory approach. As will be explained in the following sections, one can fit a higher-order Volterra model with only a first-order model matrix physically existing, and generating higher order data on demand, when performing matrix vector operations. Such approach allowed us to use map-reduce on a single multicore server with very good results, which are detailed in Table 2.1. Gains from such approach can be measured in days of computations relative to a small distributed cluster.

2.2.2 A Brief Review of Map-Reduce

The idea of map-reduce originated at Google [Dean and Ghemawat, 2008] where its original purpose was to accommodate large matrices in data analysis. The matrix and associated computations would be distributed across multiple servers configured into a cluster. Apache Hadoop is an open source implementation of a distributed data storage (DFS) and the map-reduce algorithm. It provides a necessary infrastructure to distribute the data across servers, and an application programming interface (API) for processing chunks of data and assembling the results. The concept of map-reduce was adopted in the industry largely due to the availability of the Apache implementation and the option to "lease" a Hadoop cluster from Amazon Web Services. Currently Apache Hadoop is one of the most popular platforms for the statistical analysis of big data, including both supervised and unsupervised learning. A competing map-reduce implementation comes from another Apache project, Spark, and is considered to be a better solution for statistical analysis based on iterative large matrix operations, e.g., Iteratively Re-weighted Least Square or Conjugate Gradient described in Section 2.4.

Figure 2.1 depicts the architecture of a generic map-reduce cluster. A client submits a job to the master node, with clearly specified tasks for a mapper and a reducer as well as the location and format of the input data. The master node starts a sufficient number of mappers to execute the job, passing each one a chunk of the input data. The functionality of mappers and reducers is application specific; in fact, the mappers and reducers can perform any tasks, bound only by the format of their input and output. In general, the mapper performs grouping, sorting or filtering functions, and the reducer calculates the summary of the data. Each mapper processes a chunk of data, independent from other mappers, with the results returned in the form of `<name,value>` pairs, also referred to as *keys*. A mapper can produce any number of keys as a result, recognizing that all keys with the same name will be processed by one reducer. A few examples of matrix

vector operations in map-reduce framework are provided in Section 2.2.3. Hadoop mappers and reducers are stateless, and this can become a limitation in an analysis that requires many iterative map-reduce operations. To solve this, Spark allows to execute many mappers and reducers in whichever order, and maintains a context between them. Mappers and reducers can also run within the same process in different threads or on different cores of a multicore server. Many big data applications, including classification and clustering, can benefit from map-reduce based computations, whether in a distributed environment or on a multi-core platform [Chu et al., 2007].

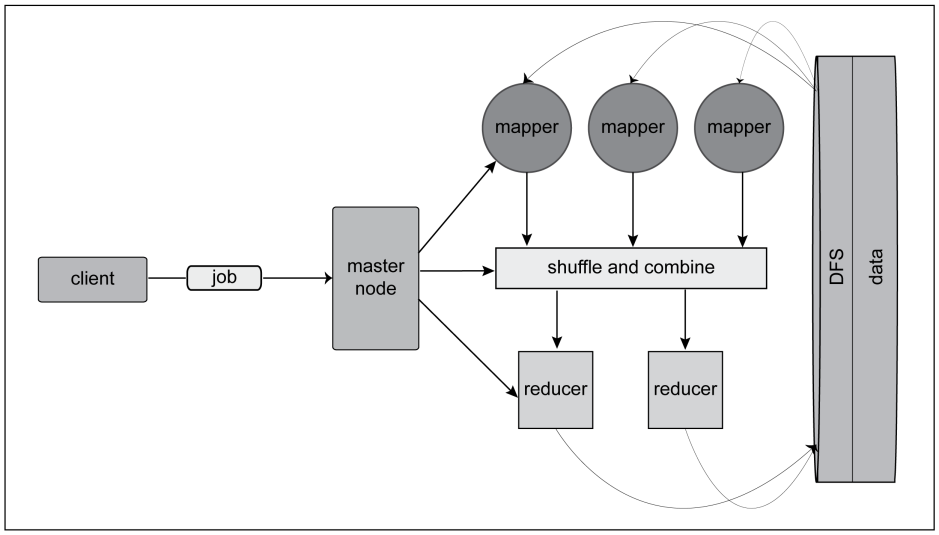

Figure 2.1: Map-Reduce architecture with mappers and reducers running on one or multiple servers.

2.2.3 Matrix processing

In classical statistical analysis, e.g., linear models and generalized linear models, one typically needs to calculate $X\beta$, X^TWy, X^TWX, for a design matrix X, a response vector y, a parameter vector β and a weight matrix W. All these operations can be performed using the map-reduce technique with a matrix X stored in a distributed file system. For better illustration, we consider three toy examples as depicted in Figures 2.2-2.4.

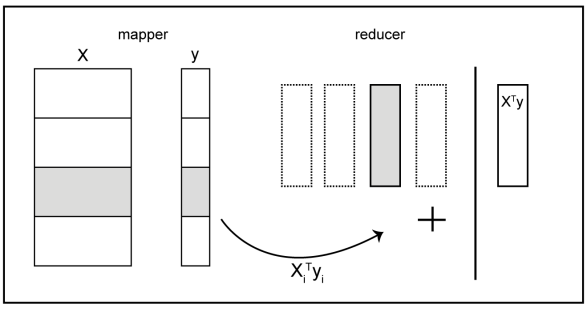

Figure 2.2: Mapper and reducer functionality for $X^T y$.

In Figure 2.2, we consider the application of the map-reduce technique to matrix multiplication of a matrix and a column vector, say $X^T y$. Here, X is an $n \times p$ matrix with $n >> p$, and y is a $n \times 1$ vector. In particular, a mapper receives a subset of rows of X, multiplies them by a corresponding subset of elements of a vector y and creates a resulting vector of size p. A reducer adds all output vectors created by the mappers together to produce the desired vector.

Figure 2.3 illustrates another widely used matrix operation in statistics. In order to compute $X^T X$ for a matrix X with large n, a mapper receives a subset of rows of X and creates a resulting square matrix of size $p \times p$. A reducer adds all matrices created by mappers. When p is small the resulting $p \times p$ matrix is easily invertible in memory.

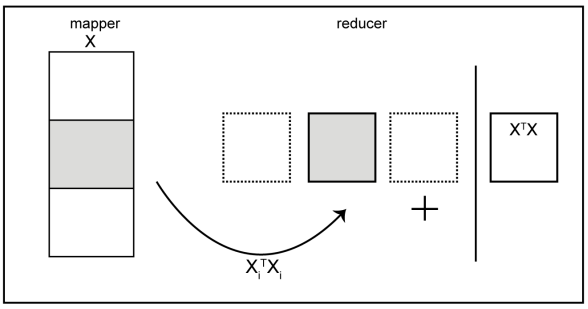

Figure 2.3: Mapper and reducer functionality for $X^T X$.

As indicated by Figures 2.2 and 2.3, the weights can be easily incorporated into the procedures to produce X^TWy and X^TWX. In fact, when n is large and p is small, the implementation of map-reduce technique is rather straightforward. However, for a large p, it becomes quite complicated. In this paper, within the scope of a *large* design matrix X of dimension $n \times p$, two detailed categories can be identified

(i) p is moderately large so that $p \times p$ square matrix is not easily invertible, but fits in memory;

(ii) p is extremely large, and $p \times p$ matrix does not fit in memory.

The category (ii) is more problematic, and may require alternative methods to circumvent matrix inversion; see Section 2.4.2 for further discussion. In the dataset under analysis, the response vector fits in-memory, so the operations X^Ty and $X\beta$ can be easily performed using the map-reduce technique, but it is the X^TWX that can be a source of computational challenges. The size of the parameters vector for the first-order Volterra model belongs to the category (i), but for the second-order and higher, it is in the category (ii). As will become evident in later sections, in many statistical problems the matrix X^TWX is only used in a matrix-vector product $X^TWX\beta$, so the large $p \times p$ matrix X^TWX does not need to be created at all. The matrix-vector product can be calculated as a single operation in the map-reduce framework. The simplified case with identity weights is visualized in Figure 2.4.

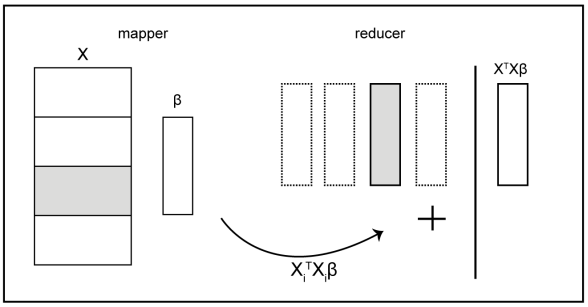

Figure 2.4: Mapper and reducer functionality for $X^TX\beta$.

2.3 Multiple-Input and Single-Output Model

2.3.1 Volterra Model

The output of a neuron or neural ensemble consists of a series of spikes, and it is convenient to model it in terms of a probability or a rate of action potentials. Let $y(t)$ be the output signal and let $x_1(t), \ldots, x_d(t)$ $(0 \leq t \leq T)$ be d binary input signals indicating the times at which the spikes occurred in a T seconds interval. In particular, both input and output signals are recorded at a fixed sampling frequency; that is, for each signal, observations are obtained at $t = \delta, 2\delta, \ldots, n\delta$ and $T = n\delta$, where δ is the length of a sampling time interval. At time t, if a spike is observed, then 1 is recorded and 0 otherwise. Although the neural membrane potential is a continuous function, identification of a spike is outside of the scope of this paper, and we will not make any assumptions about the duration of a spike.

Let $\boldsymbol{x}(t) = (x_1(t), \ldots, x_d(t))^T$ be the vector of input signals. The conditional probability of the action potential of the output signal, given the history of the input and output signals, can be written as

$$\theta(t) = P(y(t) = 1 | \mathcal{H}(t)),$$

where $\mathcal{H}(t) = (\boldsymbol{x}^T(t), \boldsymbol{x}^T(t-\delta), \ldots, \boldsymbol{x}^T(t-m\delta), y(t-\delta), \ldots, y(t-m\delta))^T$. Here $M = m\delta$ is the length of the system memory. An intuitive approach is to use the generalized functional additive model with an appropriate link function. There are two common choices: *logit* and *probit* link functions. In this paper, we will use the probit link for illustration purpose.

In Song et al. [2007], the authors proposed a discrete generalized Volterra model. The idea is to discretize each kernel function and represent it as a vector of coefficients. For simplicity, we

present their second-order model

$$\begin{aligned}\Phi^{-1}(\theta(t)) = \kappa_0 &+ \sum_{i=1}^{d}\sum_{k=0}^{m}\kappa_i^{(1)}(k)x_i(t-k\delta) \\ &+ \sum_{i_1=1}^{d}\sum_{i_2=1}^{d}\sum_{k_1=0}^{m}\sum_{k_2=0}^{m}\kappa_{i_1,i_2}^{(2)}(k_1,k_2)x_{i_1}(t-k_1\delta)x_{i_2}(t-k_2\delta) \\ &+ \sum_{k=1}^{m+1}h(k)y(t-k\delta)\end{aligned} \tag{2.2}$$

where $\kappa_0, \kappa_i^{(1)}(k), \kappa_{i_1,i_2}^{(2)}(k_1,k_2)$ are coefficients of the zeroth order, first-order and second-order feedforward kernels, and $h(k)$ are the coefficients of the feedback kernel. These authors implemented likelihood based estimation procedure. The drawback is that the numbers of parameters and observations are very large, which makes computation intractable and unstable. For instance, in our motivating example, the sampling time interval δ is 1 millisecond and the memory length M is one second, and thus, the total number of parameter is the order of 10^6. In addition, T is about 90 minutes and the total number of observations is over five million for each signal. This is a high dimensional and large sample size problem.

To circumvent this problem, Tu et al. [2012] considered a nonparametric approach for model fitting of (2.2) with only the zeroth order and first-order feedforward kernels. In particular, the authors assumed that $\kappa_i^{(1)}(k)$ take values from a smooth coefficient function. For the convenience of notation, such function is denoted by $\varphi_i^{(1)}(\cdot)$ and $\kappa_i^{(1)}(k) = \varphi_i^{(1)}(k\delta)$. These authors proposed to use B-splines to approximate all smooth coefficient functions of the first-order which greatly reduced the number of parameters. However, for the second and higher order models the number of parameters still creates a computational challenge, and so does the large sample size. Here our interest centers on computational aspects of handling the large n and the large p using the aforementioned map-reduce technique.

Next we will generalize model (2.2) by including the interactions between the input signals. To simplify our notation, let $x_{d+1}(t) = y(t-\delta)$, and we have

$$\begin{aligned}\Phi^{-1}(\theta(t)) = \kappa_0 &+ \sum_{i=1}^{d+1}\sum_{k=0}^{m}\kappa_i^{(1)}(k)x_i(t-k\delta) \\ &+ \sum_{i_1=1}^{d}\sum_{i_2=1}^{d}\sum_{k_1=0}^{m}\sum_{k_2=0}^{m}\kappa_{i_1,i_2}^{(2)}(k_1,k_2)x_{i_1}(t-k_1\delta)x_{i_2}(t-k_2\delta).\end{aligned} \tag{2.3}$$

Following the approach by Tu et al. [2012], let $\varphi_i(\tau)$ and $\varphi_{i_1,i_2}(\tau_1,\tau_2)$ be the smooth first-order and second-order kernel functions. Thus, (2.3) can be expressed as

$$\begin{aligned}\Phi^{-1}(\theta(t)) = \kappa_0 &+ \sum_{i=1}^{d+1}\int_0^M \varphi_i^{(1)}(\tau)dN_{x_i}(t-\tau) \\ &+ \sum_{i_1=1}^{d}\sum_{i_2=1}^{d}\int_0^M\int_0^M \varphi_{i_1,i_2}^{(2)}(\tau_1,\tau_2)dN_{x_{i_1}}(t-\tau_1)dN_{x_{i_2}}(t-\tau_2),\end{aligned} \tag{2.4}$$

where N_{x_i} is a counting measure based on the signal $x_i(t)$. Our proposed model architecture can be summarized in a diagram as depicted in Figure 2.5.

Assuming sufficient smoothness, all kernel functions can be approximated with basis functions. Laguerre and B-spline representations are two popular choices. Song et al. [2013] compared both basis functions and suggested that B-splines provide a better approximation when characterizing local sparsity.

2.3.2 Kernel Functions and Basis Functions

B-splines, as described in detail in Schumaker [1980] and de Boor [2001], are piecewise polynomial curves and are commonly used for smoothing purpose thanks to their flexibility (where the desired degree of smoothness can be achieved by the appropriate choice of order and placement of knots) and the availability of computationally stable, inexpensive algorithms. Let $B_1(\tau),\ldots,B_J(\tau)$ be a sequence of B-spline basis functions with certain degrees, say b, and fixed interior knots on $[0,M]$.

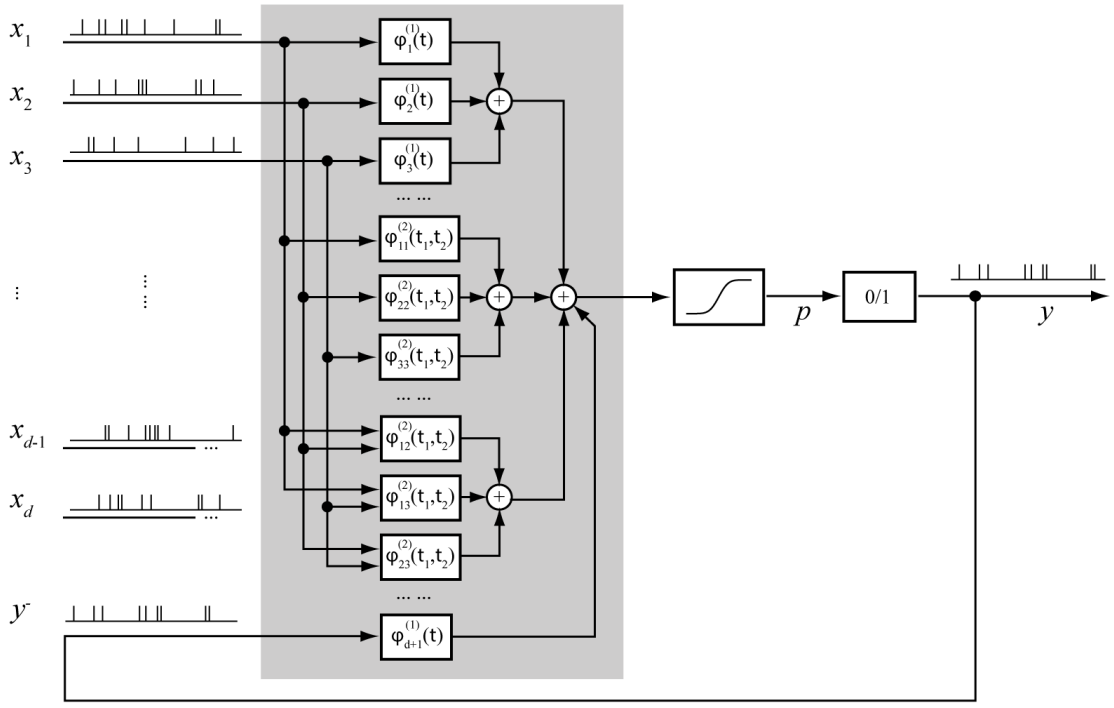

Figure 2.5: Second-order Volterra model with d inputs and one output. Inputs are convoluted with φ functionals of first- and second-order. The history of output y is treated as an additional input.

The kernel functions in (2.4) have the following approximation in terms of B-spline basis

$$\varphi_i^{(1)}(\tau) \approx \sum_{j=1}^{J} \alpha_j^{(i)} B_j(\tau) \tag{2.5}$$

$$\varphi_{i_1,i_2}^{(2)}(\tau_1,\tau_2) \approx \sum_{j_1=1}^{J}\sum_{j_2=1}^{J} \alpha_{j_1,j_2}^{(i_1,i_2)} B_{j_1,j_2}(\tau_1,\tau_2) \tag{2.6}$$

where $B_{j_1,j_2}(\tau_1,\tau_2) = B_{j_1}(\tau_1)B_{j_2}(\tau_2)$ is the product of two one-dimensional B-spline basis functions. Plugging both approximations (2.5) and (2.6) into (2.3) yields the following approximation for $\Phi^{-1}(\theta(t))$

$$\begin{aligned}\Phi^{-1}(\theta(t)) &= \alpha_0 + \sum_{i=1}^{d+1}\sum_{j=1}^{J} \alpha_j^{(i)} \int_0^M B_j(\tau) dN_{x_i}(t-\tau) \\ &+ \sum_{i_1=1}^{d}\sum_{i_2=1}^{d}\sum_{j_1=1}^{J}\sum_{j_2=1}^{J} \alpha_{j_1,j_2}^{(i_1,i_2)} \int_0^M \int_0^M B_{j_1,j_2}(\tau_1,\tau_2) dN_{x_{i_1}}(t-\tau_1) dN_{x_{i_2}}(t-\tau_2)\end{aligned} \tag{2.7}$$

where $\alpha_0 = \kappa_0$. We further write

$$\begin{aligned}\xi_j^{(i)}(t) &= \int_0^M B_j(\tau) dN_{x_i}(t-\tau) \\ \xi_{j_1,j_2}^{(i_1,i_2)}(t) &= \int_0^M \int_0^M B_{j_1,j_2}(\tau_1,\tau_2) dN_{x_{i_1}}(t-\tau_1) dN_{x_{i_2}}(t-\tau_2).\end{aligned}$$

Consequently, (2.7) can be written as

$$\Phi^{-1}(\theta(t)) = \alpha_0 + \boldsymbol{X}(t)^T \boldsymbol{\alpha} \tag{2.8}$$

where $\boldsymbol{X}(t) = (\xi_1^{(1)}(t), ..., \xi_J^{(1)}(t), ..., \xi_J^{(d+1)}(t), \xi_{1,1}^{(1,1)}(t), ..., \xi_{J,J}^{(d,d)}(t))^T$ is a row vector and $\boldsymbol{\alpha}$ is vector of the corresponding coefficients of $\boldsymbol{X}(t)$. Furthermore, $\boldsymbol{X}(t)$ can be decomposed into two subvectors, namely $\boldsymbol{X}_1(t)$ and $\boldsymbol{X}_2(t)$, where

$$\boldsymbol{X}_1(t) = (\xi_1^{(1)}(t), \ldots, \xi_J^{(1)}(t), \ldots, \xi_J^{(d+1)}(t))^T, \quad \boldsymbol{X}_2(t) = (\xi_{1,1}^{(1,1)}(t), \ldots, \xi_{J,J}^{(d,d)}(t))^T.$$

Note that $\boldsymbol{X}_1(t)$ and $\boldsymbol{X}_2(t)$ correspond to the first-order and second-order kernels respectively.

2.3.3 Likelihood and Parameter Estimation

In this section, we consider likelihood based approach for parameter estimation in (2.7). For the ease of our notation, all input signals are considered as fixed. For the binary output signal $y(t)$, observations are recorded at $t = \delta, 2\delta, \ldots, n\delta$. Here, under conditional independence, the likelihood function of $\boldsymbol{y} = (y((m+1)\delta), \ldots, y(n\delta))^T$ given $y(\delta), \ldots, y(m\delta)$ can be written as

$$\begin{aligned}L(\boldsymbol{y}|y(\delta), \ldots, y(m\delta)) &= \prod_{s=m+1}^{n} P(y(s\delta) = 1 | y((s-1)\delta), \ldots, y((s-m)\delta)) \\ &= \prod_{s=m+1}^{n} \theta(s\delta)^{y(s\delta)} (1 - \theta(s\delta))^{1-y(s\delta)},\end{aligned} \tag{2.9}$$

where $\theta(s\delta) = \Phi(\alpha_0 + \boldsymbol{X}(s\delta)^T \boldsymbol{\alpha})$. Note that the likelihood function is a function of $\theta(t)$ and, as a consequence of (2.8), is also a function of the unknown parameters α_0 and $\boldsymbol{\alpha}$. In addition, the log-likelihood function takes a form

$$l(\alpha_0, \boldsymbol{\alpha}) = \sum_{s=m+1}^{n} y(s\delta) \log(\theta(s\delta)) + (1 - y(s\delta)) \log(1 - \theta(s\delta)) \tag{2.10}$$

The maximum likelihood estimators are denoted by $(\widehat{\alpha}_0^{MLE}, \widehat{\boldsymbol{\alpha}}^{MLE})$. Maximizing the log-likelihood function with respect to the unknown parameters is a well studied convex optimization problem with an iterative solution based on the Taylor series approximation of (2.10) or equivalently, the Newton-Raphson method or Fisher scoring [Givens and Hoeting, 2012]. It is worth mentioning that, unlike the logistic regression, these two methods are not equivalent for the probit link. In fact, Newton-Raphson method provides a faster convergence but requires the calculation of the Hessian matrix at each step (compared to the Fisher information). More details are given in Section 2.4.2.

2.3.4 Functional Sparsity and Neural Connectivity

In neural network, the neural connectivity is believed to be sparse in nature [Berger et al., 2005, Song et al., 2007]. In Tu et al. [2012], the authors defined two different types of sparsity. In particular, given multiple input neurons, only a small subset of these may have any relevance to the signal recorded from the output neuron. This property is referred to as *global sparsity*, and it should lead to the estimation of functionals φ_i corresponding to irrelevant inputs as zero. Among the inputs that show relevance to the output signal, the impact will be limited in time, and the duration of the effect is often of interest. Outside of the interval in question, the functional will also be zero, which is referred to as *local sparsity*.

The classic maximum likelihood estimator (MLE) does not guarantee either sparsity, so a penalized approach enforcing it is preferable. For parametric regression models, penalized approach has been widely studied and gained its popularity due to simultaneous parameter estimation and variable selection. Commonly used penalty functions include LASSO [Tibshirani, 1996] and SCAD [Fan and Li, 2001], group LASSO [Yuan and Lin, 2006] and group bridge [Huang et al., 2009]. In Wang and Kai [2014], the authors considered the problem of detecting functional sparsity for univariate, nonparametric regression models, and examined the performance of penalized approach

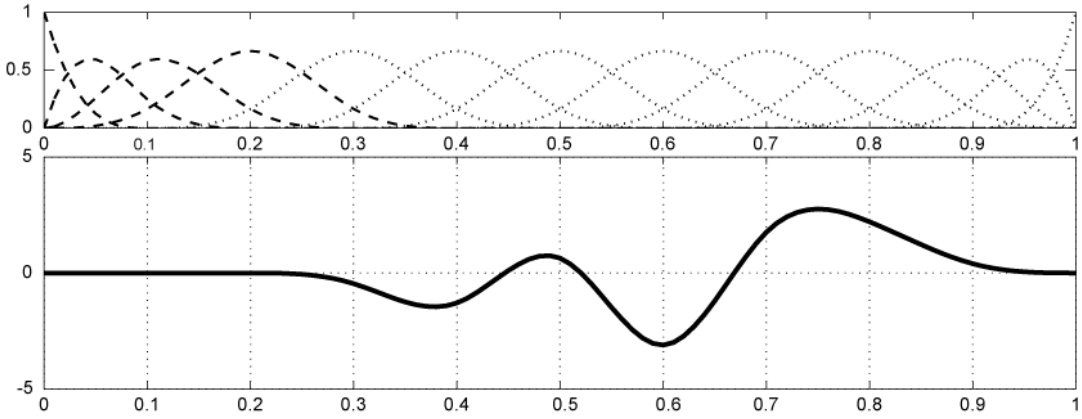

Figure 2.6: Top: A sequence of 13 cubic B-spline basis functions with 9 equally-spaced interior knots on the interval [0,1]. Bottom: An illustration of a function with local sparsity on interval [0,0.2]. There are 10 subintervals corresponding to 10 groups. On each subinterval, there are 4 basis functions coefficients taking positive values. For instance, to identify the sparsity of the function on $[0, 0.1]$, all four basis functions (dashed line type) need to have coefficients zero.

with aforementioned penalty functions. In addition, these authors incorporated the group bridge penalty with an intuitive grouping method for estimating functions with global or local sparsities. For demonstration purpose, consider the function, shown as thick, solid line type in the bottom panel of Figure 2.6. Note that, except for some singletons, this function is zero on $[0, 0.2]$. In the top panel, a collection of cubic B-spline basis function with 9 equally-spaced interior knots are depicted. Approximating with these cubic B-spline basis functions, for sparsity on $[0, 0.1]$, it is expected that the estimates for the coefficients of four basis functions, shown as dashed line type, are zero. Hence, it is natural to treat these four coefficients as a group. In general, for B-spline basis functions with degree b, each group consists of $b + 1$ coefficients.

In this paper, we adopt the grouping idea suggested by Wang and Kai [2014]. In fact, for the i-th first-order kernel $\varphi_i^{(1)}(\tau)$ in (2.7), its coefficients of B-spline approximation in (2.5) can be assigned to $J - b$ overlapping groups,

$$\{\alpha_1^{(i)}, \ldots, \alpha_{b+1}^{(i)}\}, \{\alpha_2^{(i)}, \ldots, \alpha_{b+2}^{(i)}\} \ldots, \{\alpha_{J-b}^{(i)}, \ldots, \alpha_J^{(i)}\}.$$

Similarly, for the second-order kernel $\varphi^{(2)}_{i_1,i_2}(\tau_1,\tau_2)$, the grouping becomes rather complicated. Over each small rectangle, spanned by two adjacent knots on τ_1 and τ_2, the sparsity can be determined by $(b+1)^2$ coefficients. To be more specific, its coefficients of B-spline approximation in (2.6) can be assigned to $(J-b)^2$ overlapping groups,

$$\left\{\begin{array}{ccc} \alpha^{(i_1,i_2)}_{1,1} & \dots & \alpha^{(i_1,i_2)}_{1,b+1} \\ \vdots & \ddots & \vdots \\ \alpha^{(i_1,i_2)}_{b+1,1} & \dots & \alpha^{(i_1,i_2)}_{b+1,b+1} \end{array}\right\}, \dots, \left\{\begin{array}{ccc} \alpha^{(i_1,i_2)}_{J-b,J-b} & \dots & \alpha^{(i_1,i_2)}_{J-b,J} \\ \vdots & \ddots & \vdots \\ \alpha^{(i_1,i_2)}_{J,J-b} & \dots & \alpha^{(i_1,i_2)}_{J,J} \end{array}\right\}$$

For our convenience, we write the group penalty as

$$P^{(\gamma)}_{\lambda}(\boldsymbol{\alpha}) = \lambda \sum_{g=1}^{G} |\boldsymbol{\alpha}_{A_g}|^{\gamma},$$

where G is the total number of groups and $\gamma \in (0,1)$ is a constant. Here $\boldsymbol{\alpha}_{A_1}, \dots, \boldsymbol{\alpha}_{A_G}$ represent the groups of coefficients as defined above.

Finally, the penalized log-likelihood criterion can be expressed as

$$l(\alpha_0, \boldsymbol{\alpha}) - P^{(\gamma)}_{\lambda}(\boldsymbol{\alpha}) \tag{2.11}$$

The penalized MLE is defined as the maximizer of this criterion. The computational aspects are covered in Section 2.4.4.

2.4 Computational Aspects with Big Data

2.4.1 Design Matrix

Write $\boldsymbol{\theta} = (\theta((m+1)\delta), \dots, \theta(n\delta))^T$. From (2.8), $\boldsymbol{\theta}$ can be expressed as

$$\Phi^{-1}(\boldsymbol{\theta}) = \alpha_0 + \mathbf{X}\boldsymbol{\alpha} \tag{2.12}$$

where the vector-valued function Φ^{-1} is defined by applying Φ^{-1} entry-wise. Here, $\mathbf{X}$ is the *design matrix* and can be written as $\mathbf{X} = (\boldsymbol{X}((m+1)\delta), \dots, \boldsymbol{X}(n\delta))^T$.

The design matrix $\mathbf{X}$ can be represented as $[\mathbf{X}_1, \mathbf{X}_2]$, where $\mathbf{X}_1$ and $\mathbf{X}_2$, based on $\boldsymbol{X}_1(t)$ and $\boldsymbol{X}_2(t)$, correspond to the parameters of the first-order kernels and the second-order kernels respectively. The elements of the matrix $\mathbf{X}_1$ can be calculated as

$$\xi_j^{(i)}(t) = \int_0^M B_j(\tau) dN_{x_i}(t-\tau) = \sum_{t_{ik} \in [t, t-M)} B_j(t - t_{ik}) \tag{2.13}$$

for $t = (m+1)\delta, \ldots, n\delta, j = 1, \ldots, J$ and $i = 1, \ldots, d+1$. Here, t_{ik} are timestamps such that input x_i is taking value 1. The elements of the matrix $\mathbf{X}_2$ are derived using the formula

$$\xi_{j_1,j_2}^{(i_1,i_2)}(t) = \int_0^M \int_0^M B_{j_1,j_2}(\tau_1, \tau_2) dN_{x_{i1}}(t-\tau_1) dN_{x_{i2}}(t-\tau_2) = \xi_{j_1}^{i_1}(t) * \xi_{j_2}^{i_2}(t) \tag{2.14}$$

Note that, the 2-dimensional B-spline basis functions are tensor products of the 1-dimensional linear basis. Consequently, the rows of $\mathbf{X}_2$ can be efficiently calculated as the Kronecker products of the rows of $\mathbf{X}_1$. This observation is rather important for big data computation; in fact, the matrix $\mathbf{X}_2$ does not have to physically exist, which largely alleviates storage and computational challenges in further calculations.

2.4.2 Maximum Likelihood Estimation with Conjugate Gradient

The iteratively reweighted least squares (IRLS) is a well established algorithm for finding a maximum likelihood estimate of a generalized linear model (2.12); see McCullagh and Nelder [1989] for more details. This iterative approach is based on the Taylor expansion of the log-likelihood function using either observed or expected information. At the $(k+1)$-st step, one finds the estimate of $\boldsymbol{\alpha}$, say $\boldsymbol{\alpha}^{(k+1)}$, based on the previously calculated $\boldsymbol{\alpha}^{(k)}$ by minimizing $\|\mathbf{W}_k^{\frac{1}{2}}(\boldsymbol{z}^{(k)} - \alpha_0 - \mathbf{X}\boldsymbol{\alpha})\|_2^2$. Here, $\boldsymbol{z}^{(k)}$ is a current response vector and $\mathbf{W}_k$ is a matrix of weights depending on $\boldsymbol{\alpha}^{(k)}$. In fact, $\mathbf{W}_k$ is a diagonal matrix with elements $w_j^{(k)}$,

$$w_j^{(k)} = \phi(\alpha_0^{(k)} + \mathbf{X}_j^T \boldsymbol{\alpha}^{(k)})^2 / (\theta_j^{(k)}(1 - \theta_j^{(k)})), \tag{2.15}$$

and $\boldsymbol{z}^{(k)}$ is a vector with elements $z_j^{(k)}$ defined as

$$z_j^{(k)} = \alpha_0^{(k)} + \mathbf{X}_j^T \boldsymbol{\alpha}^{(k)} + (y_j - \theta_j^{(k)}) / \phi(\alpha_0^{(k)} + \mathbf{X}_j^T \boldsymbol{\alpha}^{(k)}), \tag{2.16}$$

where $\mathbf{X}_j^T$ is the j-th row of $\mathbf{X}$, $\theta_j^{(k)} = \Phi(\alpha_0 + \mathbf{X}_j^T \boldsymbol{\alpha}^{(k)})$, and ϕ, Φ are the density and distribution functions of standard normal respectively. Moreover, the minimization problem above has a closed form solution,

$(\alpha_0^{(k+1)}, (\boldsymbol{\alpha}^{(k+1)})^T)^T = (\mathbf{Z}^T \boldsymbol{W}_k \mathbf{Z})^{-1} \mathbf{Z}^T \boldsymbol{W}_k \boldsymbol{z}^{(k)}$, where $\mathbf{Z} = [1, \mathbf{X}]$. In practice, for a moderately large matrix $\mathbf{X}$, the closed form solution at every iteration can be calculated using a map-reduce framework as described in Section 2.2.3. However, the operation of matrix inversion is not feasible with a massively large matrix $\mathbf{X}$. An alternative solution is to use the *conjugate gradient* method.

Conjugate gradient is an effective optimization method developed for solving quadratic optimization problems of the type: $\min_x \frac{1}{2} x^T \mathbf{H} x - x^T b$, but used also for more general nonlinear optimization. It is based on the notion of conjugate vectors, say $d_1, ..., d_p$ such that $d_i^T \mathbf{H} d_j = 0$, for $i \neq j$. The conjugate vectors are linearly independent and span the Euclidean space $\mathbb{R}^p$. The solution of the quadratic optimization problem can then be expressed as a linear combination of these vectors. The coefficients of that representation can be obtained iteratively in p steps. The conjugate gradient method has been successfully applied to classification in the big data context due to its scalable computational complexity; see Komarek [2004]. In fact, the number of operations for a convex quadratic problem is estimated to be $O(p)$. The implementation, which details can be found, among others, in Antoniou and Lu [2007], requires approximately p matrix-vector multiplications. To carry out the IRLS algorithm, at the $(k+1)$-st step, the minimization problem can be expressed in terms of $\mathbf{H} = 2\mathbf{Z}^T \boldsymbol{W}_k \mathbf{Z}$ and $b = 2\mathbf{Z}^T \boldsymbol{W}_k \boldsymbol{z}^{(k)}$, where $\boldsymbol{W}_k$ and $\boldsymbol{z}^{(k)}$ are defined as in (2.15) and (2.16). Note that, in big data computation, the matrix $\mathbf{H}$ is never present in memory since it is only used in a matrix-vector multiplication, which is computable in a map-reduce framework; see Section 2.2.3. The vector b is of size $p \times 1$ and it can be computed with map-reduce and stored in memory.

It is worth mentioning that conjugate gradient method is also flexible when additional linear constraints are present. For instance, in model (2.4), it is reasonable to assume that $\varphi_i^{(1)}(M) =$

$\varphi_i^{(1)}(m\delta) = 0$ for all first-order kernels. This can be effectively written as linear equality constraints in terms of the vector of coefficients. As pointed out by Antoniou and Lu [2007], the equality constraints result in the problem of dimension reduction, and the reduced parameter vector can be estimated following the same iterative procedure.

2.4.3 Penalized Maximum Likelihood Estimation

The MLE obtained using IRLS in Section 2.4.2 does not provide any sparsity. One way to enforce sparsity of the identified model is to consider minimizing the penalized log-likelihood as expressed in (2.11). The minimization problem poses computational challenge since it is non-convex. A popular approach is to approximate the log-likelihood function by a quadratic function. Let $l_0(\boldsymbol{\alpha}) = l(\widehat{\alpha}_0^{MLE}, \boldsymbol{\alpha})$ be the likelihood function after plugging-in $\widehat{\alpha}_0^{MLE}$. The approximation of $l_0(\boldsymbol{\alpha})$ can be obtained via a Taylor series expansion at the maximum likelihood estimation $\widehat{\boldsymbol{\alpha}}^{MLE}$; that is,

$$l_0(\boldsymbol{\alpha}) \approx l_0(\widehat{\boldsymbol{\alpha}}^{MLE}) + \frac{1}{2}(\boldsymbol{\alpha} - \widehat{\boldsymbol{\alpha}}^{MLE})^T \nabla^2 l_0(\widehat{\alpha}^{MLE})(\boldsymbol{\alpha} - \widehat{\boldsymbol{\alpha}}^{MLE})$$

where $\nabla^2 l_0(\widehat{\boldsymbol{\alpha}}^{MLE}) = -\mathbf{Z}^T\mathbf{W}\mathbf{Z}$ is the Hessian of the likelihood function evaluated at the $\widehat{\boldsymbol{\alpha}}^{MLE}$, $\mathbf{Z} = [1, \mathbf{X}]$ and $\mathbf{W}$ is a diagonal weight matrix. This quadratic approximation can be expressed as

$$l_0(\boldsymbol{\alpha}) \approx l_0(\widehat{\boldsymbol{\alpha}}^{MLE}) - \frac{1}{2}\|\mathbf{W}^{\frac{1}{2}}(\boldsymbol{z} - \mathbf{X}\boldsymbol{\alpha})\|_2^2 \tag{2.17}$$

where $\boldsymbol{z} = \mathbf{X}\widehat{\boldsymbol{\alpha}}^{MLE}$. Thus, the penalized criterion (2.11) can be expressed as

$$\underset{\boldsymbol{\alpha}}{\operatorname{argmin}} \frac{1}{2}\|\mathbf{W}^{\frac{1}{2}}(\boldsymbol{z} - \mathbf{X}\boldsymbol{\alpha})\|_2^2 + P_\lambda^{(\gamma)}(\boldsymbol{\alpha}). \tag{2.18}$$

In the special case of $\gamma = 1$, the penalty is essentially the LASSO penalty [Tibshirani, 1996]. For $\gamma \in (0, 1)$, the optimization problem is non-convex, and Huang et al. [2009] provided an iterative solution that can be summarized as follows

1. For a given λ_n, calculate $\tau_n = \lambda_n^{1/1-\gamma}\gamma^{\gamma/1-\gamma}(1-\gamma)$

2. For $s = 1, 2, \ldots$, until convergence, compute

$$\boldsymbol{\alpha}^{(s)} = \underset{\alpha}{\operatorname{argmin}} \tfrac{1}{2}\|\mathbf{W}^{\frac{1}{2}}(z - \mathbf{X}\boldsymbol{\alpha})\|_2^2 + \sum_{g=1}^{G} (\zeta_g^{(s)})^{1-1/\gamma} \sum_{i:\alpha_i \in A_g} |\alpha_i|$$

where

$$\zeta_g^{(s)} = c_g(\tfrac{1-\gamma}{\tau_n \gamma})^{\gamma} \sum_{i:\alpha_i \in A_g} |\alpha_i^{(s-1)}|^{\gamma}, \text{ for } g = 1, .., G$$

Note that each stage of the iterative procedure can be reduced to the minimization of a quadratic form with L_1 penalty applied to the coefficients $\boldsymbol{\alpha}$. For small data set, it can be efficiently solved by the LARS algorithm [Efron et al., 2004]. However, in the big data context, this algorithm may not be adaptable. Finding the solution to such a problem applicable to a large data set is the subject of the next section.

It is important to mention that the group bridge penalty described above can encounter stability issues in calculations [Percival et al., 2012]. An alternative approach is to use the SCAD penalty [Fan and Li, 2001], which can be written as

$$p_\lambda(|\alpha_j|) \approx p_\lambda(|\widehat{\alpha}^{MLE}|) + p_\lambda^{'}(|\widehat{\alpha}^{MLE}|)(|\alpha_j| - |\widehat{\alpha}^{MLE}|) \tag{2.19}$$

An analog of (2.11), equipped with SCAD penalty, can be expressed as

$$\frac{1}{2}\|\mathbf{W}^{\frac{1}{2}}(\boldsymbol{z} - \mathbf{X}\boldsymbol{\alpha})\|_2^2 + \sum_{g=1}^{G} p_\lambda(\|\boldsymbol{\alpha}_{A_g}\|_1).$$

Zou and Li [2008] proposed a one-step estimation procedure, which is based on a linear approximation of the penalty function. Thus, (2.19) can be written as a standard LASSO problem; that is,

$$\frac{1}{2}\|\mathbf{W}^{\frac{1}{2}}(\boldsymbol{z} - \mathbf{X}\boldsymbol{\alpha})\|_2^2 + \sum_{g=1}^{G} \eta_g \|\boldsymbol{\alpha}_{A_g}\|_1,$$

where $\eta_g = p'(\|\widehat{\boldsymbol{\alpha}}_{A_g}^{(MLE)}\|_1)$.

As will be seen in Section 3.2, in our application, the estimates using the group bridge penalty and the SCAD penalty are comparable.

2.4.4 L_1 Regularization via Coordinate Descent Algorithm

As outlined in Section 2.4.3, parameter estimation with the group bridge penalty and SCAD penalty can be reduced to the LASSO problem. LASSO has an efficient implementation using a variation of the LARS algorithm [Efron et al., 2004], which is not well tuned for operating on the design matrix of high dimension. Another implementation was suggested by Friedman et al. [2007] and Friedman et al. [2010]. The elastic-net penalty, of which LASSO is a special case, can be computed by a coordinate descent and soft thresholding. The algorithm is applicable to linear models, or generalized linear models, in which case it can be integrated with the IRLS steps.

The idea of soft thresholding was proposed by Donoho and Johnstone [1995], and it was designed to recover a signal in the noisy environment by providing an adaptable threshold. This approach is motivated by a simple optimization problem

$$\min_t \frac{1}{2}(t - t_0)^2 + \lambda|t|,$$

where $\lambda > 0$ and t_0 is a fixed constant. The solution can be written as $t_{\min} = S(t_0, \lambda) \equiv \text{sign}(t_0)(|t_0| - \lambda)_+$. This observation motivated the gradient descent algorithm for the LASSO solution; see Friedman et al. [2010] for more details. In particular, starting from any given parameter values, one parameter is updated using the soft thresholding. The detailed algorithm is described in Friedman et al. [2010]. When implementing this algorithm, matrix-vector products are necessary and can be carried out efficiently under the map-reduce framework.

2.4.5 Tuning Parameters

The performance of parameter estimation relies on the choice of λ and γ. Here, the parameter γ is fixed at 0.5 following recommendations in Huang et al. [2009]. The common procedure for selecting λ is to proceed with model identification for multiple values of the coefficient and choose one according to a selected criterion, e.g., the Akaike's Information Criterion [AIC; Akaike,

1973], the Schwarz' Bayesian Criterion [BIC; Schwarz, 1978], or, especially for larger models, the extended BIC [EBIC; Chen and Chen, 2008]. Such criterion-based methods are suitable for the selection in the first-order model, but more problematic for the second-order model, or for any large data in general. The selection of λ for the second-order model requires multiple identification efforts, but time cost, a major concern for big data analytics, does not allow an exhaustive search. In this paper, for the second-order model, we suggest to search for the tunning parameter among the predetermined increasing sequence of candidates starting from the value selected for the first-order model.

2.5 Data Analysis

2.5.1 Analysis of Neural Spikes

The data analyzed in this section comes from a public repository `crcns.org` and consists of neural recordings performed on a rat exposed to a maze; for details see Mizuseki et al. [2009a,b]. The data set contains timestamps of neural spikes (action potentials) recorded in eight neuron cells within a 90 minute timeframe. A portion of the data set, recorded over three seconds, is shown in Figure 2.7. All neurons come from CA1 region of hippocampus. We consider seven of them as inputs and one as output.

For each neuron, it is assumed that one observation, either 0 or 1, is recorded every $\delta = 1$ millisecond (ms), which yields over five million observations. In addition, the memory length is chosen to be $M = 400$ milliseconds. Based on earlier studies [e.g., Song et al., 2013], the memory length is generally less than 1 second. In our data analysis, results from $M = 400$ and $M = 1000$ are very similar. For illustration purpose, we present only the results with $M = 400$ milliseconds. All the observations are used to fit the first- and second-order models described in the following subsections.

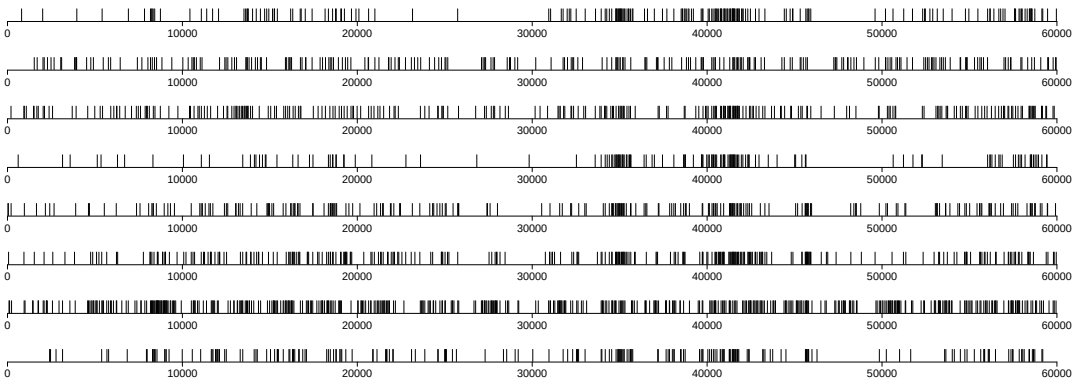

Figure 2.7: Three-second recording (out of 90 minutes) of seven inputs $x_1, \ldots, x_7$ (top rows) and output y (bottom row).

2.5.2 First-Order Model

In this section we model the probability of neural spikes as a GLM defined in (2.8) with a matrix $\mathbf{Z} = [1 \quad \mathbf{X}_1]$ and its elements defined in (2.13). Even though the number of observations n is over five million, the matrix $\mathbf{X}_1$ can be computed and stored in memory for all calculations. Here we assume that $\varphi(m\delta) = 0$. We also consider a constraint $\varphi(0) = 0$ because the impact of the input spike is expected to have a positive delay. However, the delay might not be detectable in the data if the sampling rate is not high enough, in which case the value of the $\varphi(0)$ may correspond to the maximum value of the functional. In fact, this turns out to be the case for all seven input neurons. Figures 2.8 and 2.9 show the results of the first-order model fit, with the MLE estimate depicted with a solid line, and three different penalized fits including group bridge (dashed), group SCAD (dotted) and LASSO (dash-dotted). The zoom-in view in each image shows the differences in sparsity estimation for all four models. The group bridge fit exhibits local sparsity in four out of seven neurons. The group SCAD and LASSO fits do not achieve the local sparsity in the examined range. The group bridge estimate was calculated with $\lambda_n = 40$ selected using the EBIC criterion. The group bridge fit has a lower EBIC value than those of the group SCAD fit and LASSO fit.

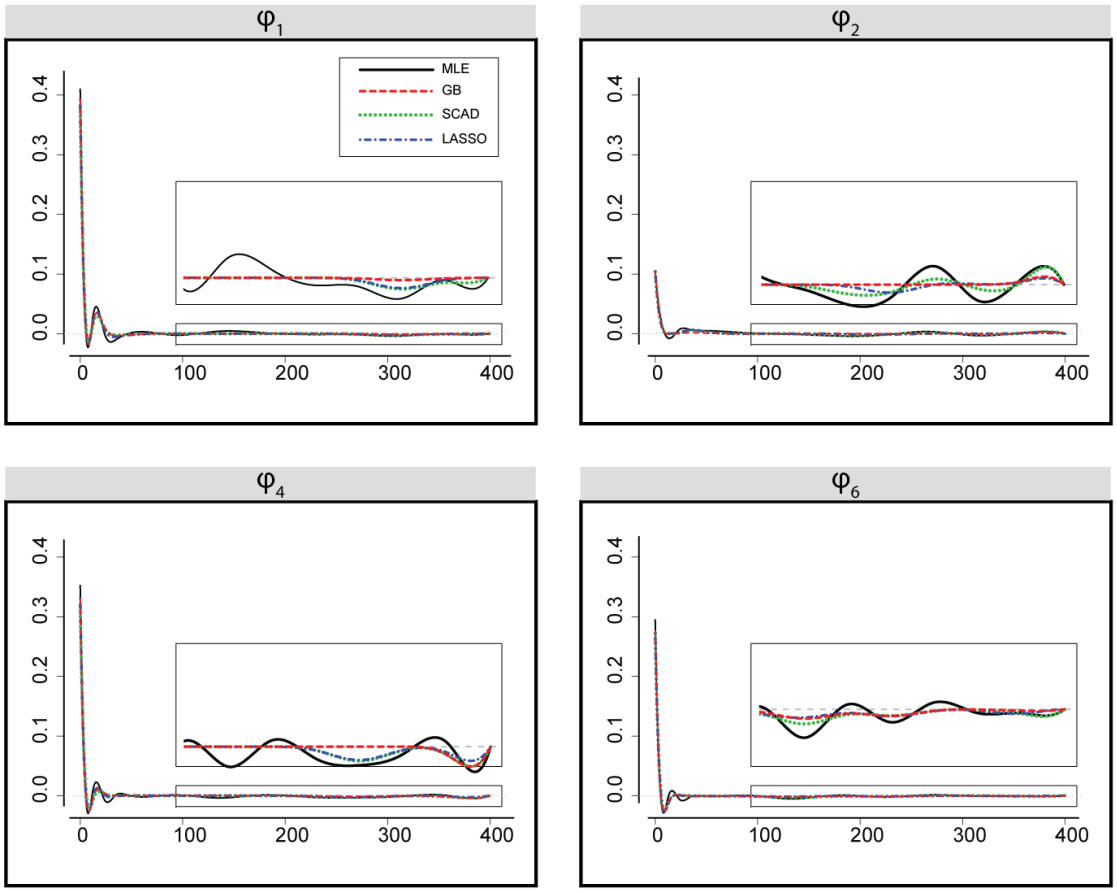

Figure 2.8: Maximum likelihood estimation (solid) and penalized maximum likelihood estimation using a group bridge penalty (dashed), SCAD (dotted) and LASSO (dash-dotted) of φ functionals in the first-order model corresponding to the input neurons: x_1, x_2, x_4, x_6. The estimates are comparable, and only a zoom-in view highlights differences in sparsity estimation.

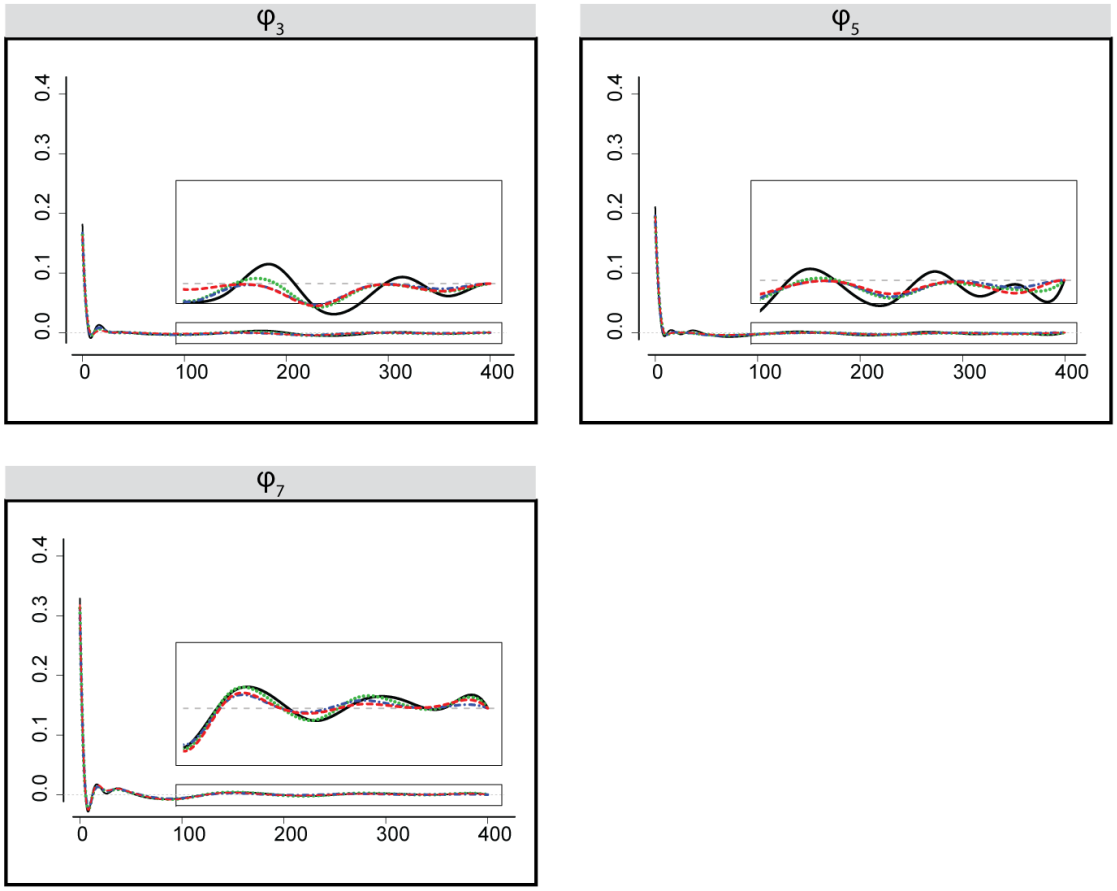

Figure 2.9: Maximum likelihood estimation (solid) and penalized maximum likelihood estimation using a group bridge penalty (dashed), SCAD (dotted) and LASSO (dash-dotted) of φ functionals in the first-order model corresponding to the input neurons: x_3, x_5, x_7. No sparsity is detected.

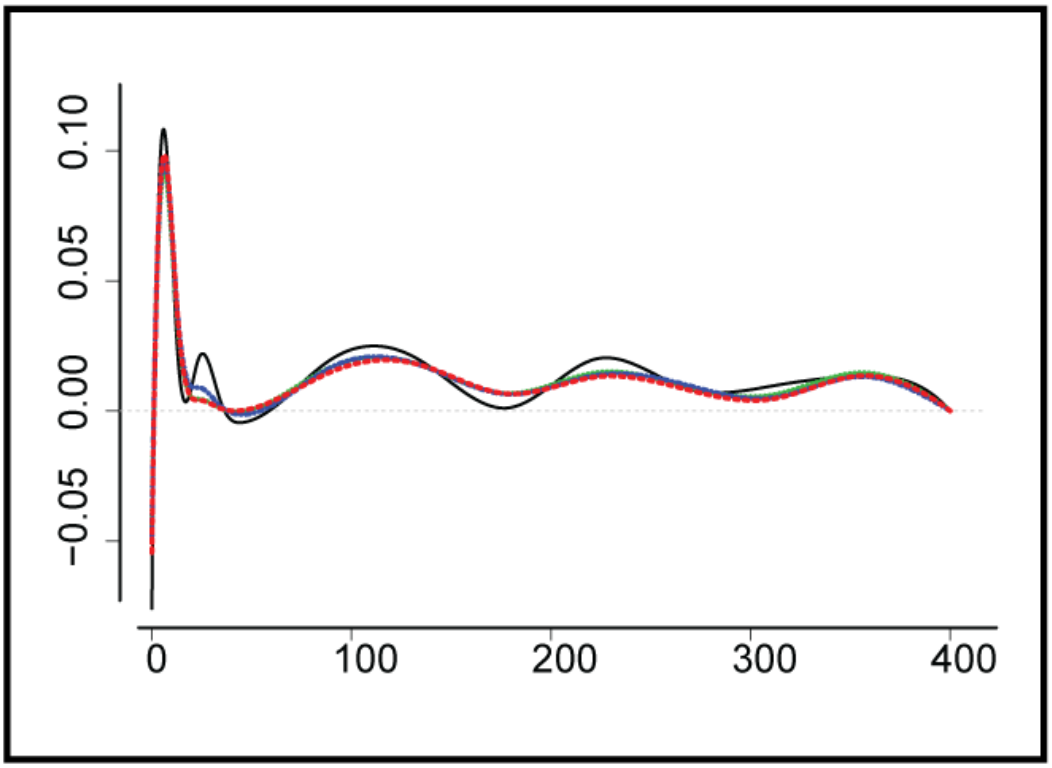

Figure 2.10: Maximum likelihood estimation (solid) and a group bridge estimation (dashed) of a φ functional in the first-order model corresponding to the output neuron. Local sparsity is not detected.

Figure 2.10 depicts the maximum likelihood and a group bridge estimation of the kernel corresponding to the feedback kernel. The negative value at 0 confirms the expected inhibitory effect of a recent spike on the probability of the immediate succeeding firing. The positive value close to 0 could be an indication of a clustering effect in neural spikes, which can also be observed in Figure 2.7. Local sparsity is not detected by any estimator.

2.5.3 Second-Order Model

In this section we model the probability of neural spikes as a GLM defined in (2.8) with a matrix $\mathbf{Z} = [1 \quad \mathbf{X}_1 \quad \mathbf{X}_2]$ and its elements defined in (2.14). Only the matrix $\mathbf{X}_1$ is computed, and the matrix $\mathbf{X}_2$ is created dynamically to perform necessary matrix operations, as explained in previous sections. Due to time constraints we compute only two estimates, the maximum likelihood estimate and the penalized maximum likelihood estimate using the group bridge penalty. Recall that this penalty function outperforms others in the first-order analysis based on the EBIC criterion.

In Figure 2.11, the functional estimates of the first-order kernels $\varphi_i^{(1)}$ are very close to the first-order counterparts as depicted in Figures 2.8-2.10. Note that the local sparsity in the second-order model is more pronounced. The estimated interaction components, i.e., the second-order kernels, are presented in Figure 2.12. Only four second-order kernels are shown, φ_{11}, φ_{37}, φ_{57} and φ_{67}. The remaining kernels exhibit global sparsity, which suggests no interaction exists between corresponding input neurons or within the same input neurons at different time lags.

Table 2.1: Performance results for the second-order model with map-reduce for different hardware configurations: a multicore server with 1, 4, 8, 24, 40 cores, a hadoop cluster with 4 servers and a spark cluster running on top of the hadoop distributed file system (HDFS). The tests are performed for the first-order matrix with dimensions $n = 10^6, p_1 = 120$. The dimensions of the second-order matrix are $n = 10^6, p_2 = 7380$. The operations listed in the first column are the only big data operations used in our proposed method. The last three columns show the approximate number of big data operations per method used in this paper: Conjugate Gradient (CG), Iteratively Reweighted Least Squares (IRLS) and Coordinate Descent (CD).

	multicore [s]					hadoop [s]	spark [s]	method [count]		
	1	4	8	24	40	4	4	CG	IRLS	CD
$X^T y$	58.47	16.34	9.6	7.75	3.55	220	130	0	50	$50p_2$
$X\alpha$	50.8	14.8	8.91	6.89	3.22	210	106	0	50	0
$X^T W X\alpha$	98.31	26.39	14.71	10.12	5.11	280	188	$p_2/8$	$6p_2$	0

To examine the computational performance in big data context, we implement our proposed method in different hardware environments. Computing times for second-order model fit are

presented in Table 2.1. The results are rather unfavorable for the Hadoop cluster. This may partially be due to the fact that the performance test was conducted in `R`, which is not a very efficient computing platform. The Spark test uses `Python`, which is known to be more efficient at matrix operations, but even those results cannot match the in-memory multicore operations, which are executed using `R` and `C`. As the results make clear, the penalty for distributing data across multiple servers is very severe. The computational cost of recreating the data (e.g., recreating the matrix columns that constitute a Kronecker product of the existing columns) is much smaller than the cost of distributed communication. It should be noted, however, that the clusters we used are very small, larger clusters will drive the performance numbers down. On the other hand, multicore servers with 250 and more cores are becoming available, and that should make the in-memory identification of second- and higher-order models faster or feasible.

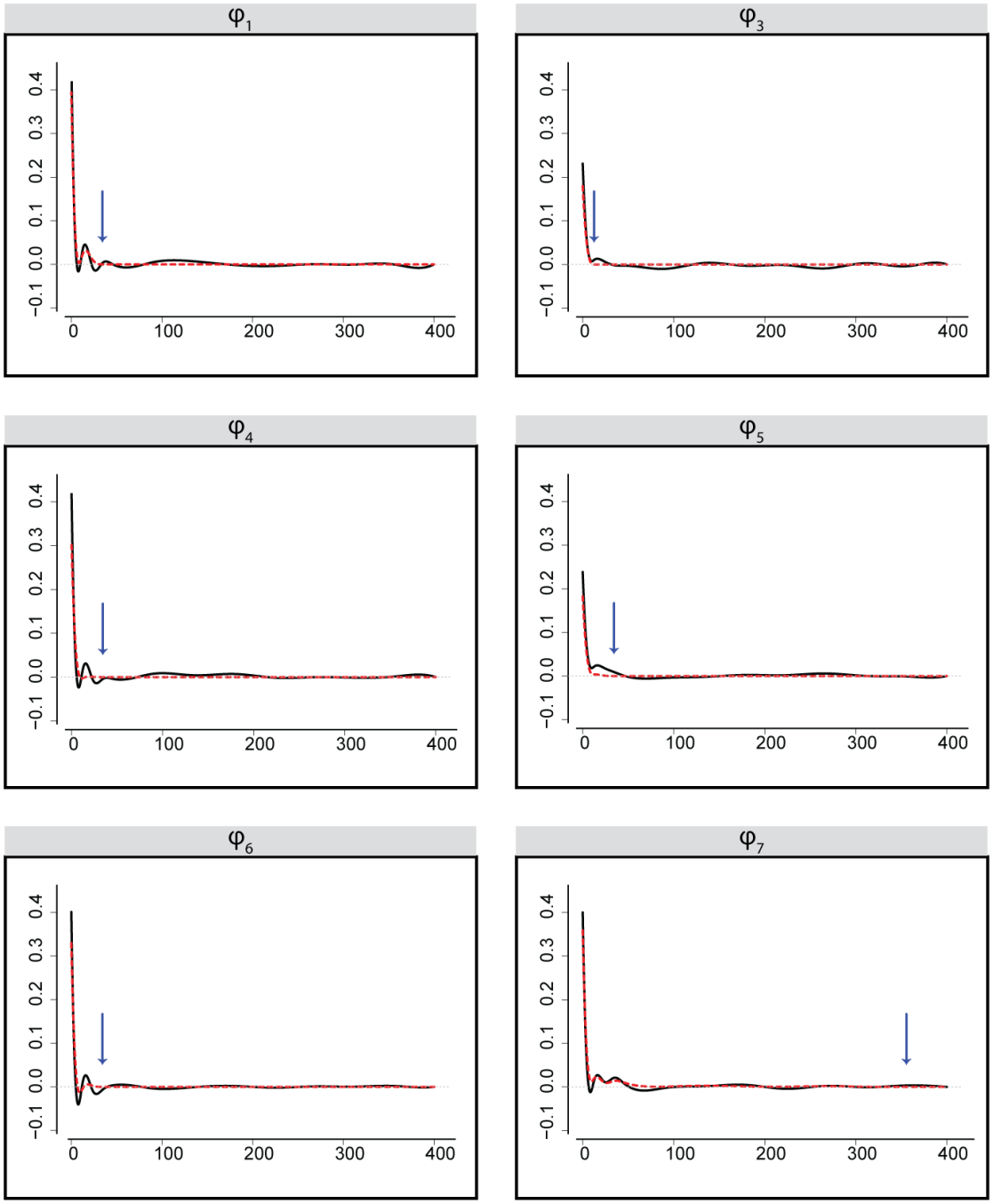

Figure 2.11: Maximum likelihood estimation (solid) and a group bridge estimation (dashed) of φ functionals in the second-order model corresponding to the input neurons x_1, x_3, x_4, x_5, x_6, x_7. Local sparsity visible for the penalized version in all neurons, the start of sparsity is marked with an arrow. Neuron x_2 exhibits global sparsity.

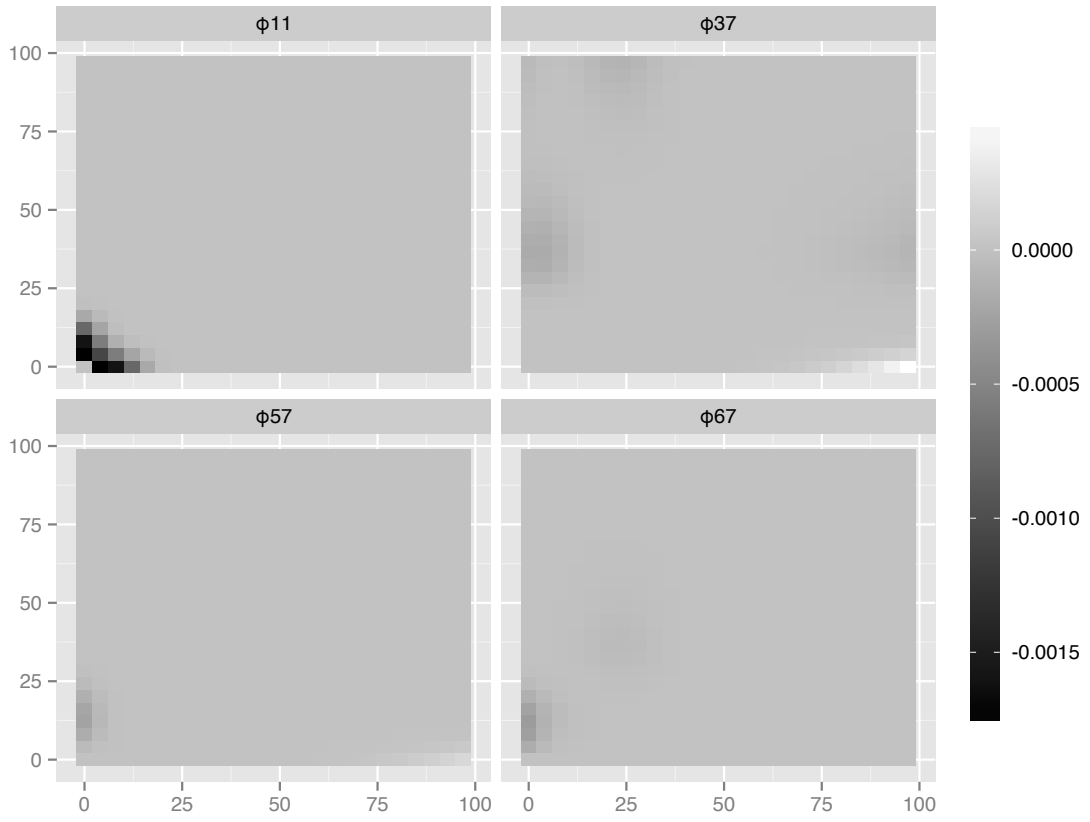

Figure 2.12: Penalized estimation of interaction components in the second-order model. Only non-zero components are displayed.

2.5.4 Model Validation

Model validation for neuron spike train data in functional dynamical settings is a challenging task and is still an area of active research. Brown et al. [2002] proposed to use a time-rescaling theorem to evaluate a goodness-of-fit of a neural spike model. The method relies on the intensity function for continuous time. The intensity function, integrated between two adjacent observed output spikes, is assumed to follow an exponential distribution with a unit rate. Thus, after an appropriate transformation to a uniform distribution, the Kolmogorov-Smirnov (KS) test can be used to test the goodness-of-fit. Haslinger et al. [2010] focused on a discretized model for neuron spikes, a more practical approach that is characterized by dividing time variable into short intervals or bins, and encoding a spike train as a binary sequence of 0's and 1's, corresponding to whether a spike was observed in a given bin. Those authors pointed out that the KS test in Brown et al. [2002] may exhibit biases for discretized approach. The biases are caused by physical and numerical constraints. For instance, neural spikes are not instantaneous events, and in fact are believed to last about 1 millisecond. In addition, the bin length creates a lower bound on the inter-spike interval. For the discretization approach, the distribution of data is geometric rather than exponential, and the exponential approximation may not be valid for large firing probabilities. Haslinger et al. [2010] further suggested an improved rescaling of estimated firing probabilities for bias correction. In our analysis, we adopt the discrete time rescaling theorem [Haslinger et al., 2010]. The resulting KS plot is shown in Figure 2.13. It can be seen that the overall model fit is satisfactory. Small bias is visible for low probabilities, which is expected due to the discretization of the time interval and the dependence on the spike history. This phenomenon was previously reported in Haslinger et al. [2010]. In fact, for the KS plot generated from the true model, such biases are also visible.

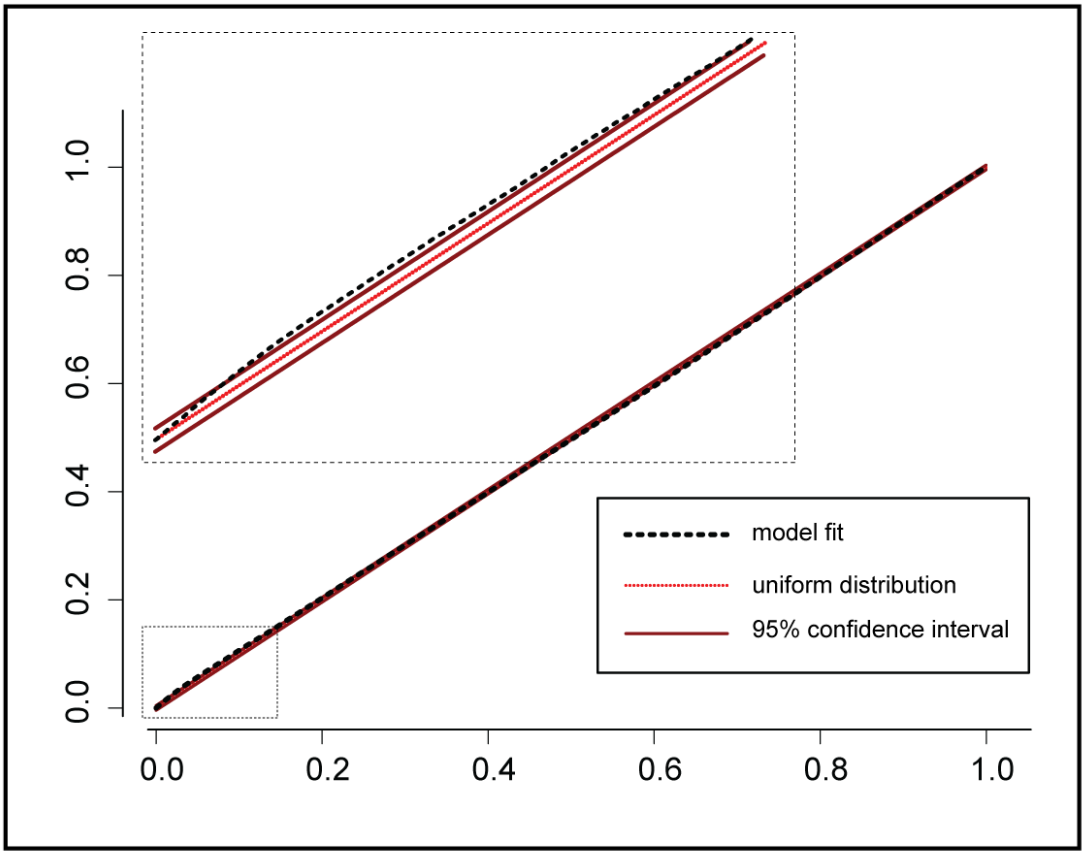

Figure 2.13: KS plot for rescaled firing probabilities against a reference distribution (uniform). Rescaled firing probabilities are depicted with a thick dashed line, reference uniform distribution with a dotted line, and a 95% confidence interval with a thin solid line. Small bias is visible for low probabilities, more evident in the zoom-in subplot.

CHAPTER 3

Pareto Quantiles of Unlabeled Tree Objects[1]

3.1 Introduction

When studying functional aspects of the brain, the hippocampus region is of particular interest. It is associated with long term memory and learning, and it is highly sensitive to pathological changes (e.g., disease, brain injuries). There is an ongoing effort to understand the dynamic behavior of hippocampal neuron cells, specifically their connectivity and firing activity (also known as spike trains). The information transmission between two regions of hippocampus, CA3 (input) and CA1 (output), has been extensively modeled in an effort to develop, among others, a neural prosthesis [**?**]. Less is known about the topological aspects of neurons in these two regions. Pyramidal neurons from the hippocampus typically consist of a soma, an axon and two types of dendrites (see Figure 3.1). The tree-like dendritic structures, also referred to as arborizations, are commonly associated with the functional complexity of the brain. The current "synaptotropic hypothesis", as stated in Cline and Haas [2008], describes the growth of dendritic branches as "dynamic and exploratory". The branches can live for as short as 10 minutes, as they "sample the environment to detect the appropriate cells" [Cline and Haas, 2008]. This dynamic process cannot be directly observed, and the data available for analysis only provide one snapshot in the lifetime of a neuron. However, given a set of static reconstructions of neural cells at different stages of maturity, our goal is to characterize the process governing the growth.

In statistical modeling, each neuron can be regarded as a *data object*, a complex entity that is generally outside the scope of classical statistics. The class of data objects can include images, trees, graphs, and often curves; see Marron and Alonso [2014] for a recent review of objects

[1]This Chapter is based on the early draft of paper "Pareto Quantiles of Unlabeled Tree Objects."

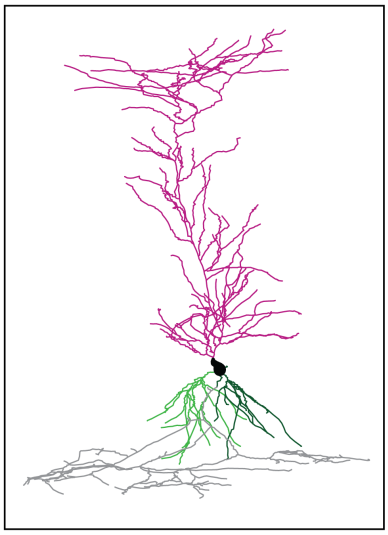

Figure 3.1: Graphical display of a pyramidal neuron cell, named after its pyramid-like shape. All arborizations grow out of the *soma*, which is depicted in black. Other components include the *axon* shown in grey, *apical dendrites* shown in magenta and *basal dendrites* shown in green (two shades of green are used to depict two disjoint arborizations). The basal dendrites often form a forest of several disjoint binary trees. The axon is ignored in our analysis.

and related statistical methods. The term *Object Oriented Data Analysis*, a class of tools for the analysis of complex data objects, was introduced to statistics by Wang and Marron [2007]. Since then, there has been a great deal of research to extend traditional statistical methods, e.g., regression and principal component analysis, to the space of complex data objects [Chang et al., 2011, Shen et al., 2014, Wang et al., 2012].

In classical statistics, descriptive measures, such as mean and deviation from the mean, have been widely used to describe and summarize information from data. But those statistics may not be sufficient to highlight the characteristics of complex data objects. For instance, in neuroscience, neurons from different brain regions exhibit topological "heterogeneity". The multitude of shapes, sizes and branching patterns observed in neural cells, calls for a more comprehensive depiction of the population distribution. For a univariate random variable, a comprehensive characterization can be established through a quantile function, which provides an intuitive, probabilistic way to

measure centrality, dispersion, skewness, and the tail behavior of the distribution. In particular, the quantile function is defined through the cumulative distribution function. However, the definition of quantiles becomes non-trivial for a multivariate random vector due to the lack of a natural order in high-dimensional space. Liu [1990] introduced the notion of *simplicial depth* and showed that it can be used as an analog of multivariate order statistics. Serfling [2002] provided a survey of different approaches to multivariate quantile definitions and useful criteria for their evaluation. The most notable methods are based on depth function and norm minimization. Functional data provide even more challenges, because standard approaches for a finite-dimension do not translate well to a functional space. Walter [2011] offered a thorough study of the properties of functional quantiles and their empirical analogs backed up by a case study of financial data. In that study, the author employed point-wise quantiles which are biased estimators of population quantiles, but they are consistent under some weak conditions.

The challenges increase even more for complex data objects, such as tree data, which can be characterized as extremely non-Euclidean; see Wang and Marron [2007]. There have been previous attempts to define a median of a population of such objects. Some examples come from the work on classification trees [Banks and Constantine, 1998, Phillips and Warnow, 1996]. The tree-structured data objects discussed in these papers are of a binary form, and their nodes can be uniquely labeled for correspondence between trees. The median tree is thus defined as a majority tree, i.e., a tree consisting of nodes found in the majority of trees in the set. Node labels are important and natural for classification trees, or phylogenetic trees [Billera et al., 2001], but for some tree-structured objects, e.g., brain arteries, neural dendrites, there is no established labeling scheme. The labeling of nodes can be crucial in answering many important research questions, but different labeling choices could lead to different results; see Aydın et al. [2009] for a discussion on thickness correspondence and descendant correspondence between brain artery systems. Node

labeling may also create a potential bias when estimating the density of branches. This point will be demonstrated in Section 3.4.

In this paper we propose a novel approach to evaluate quantiles of tree objects that does not rely on labeling of nodes or edges. We base our approach on a stochastic process view of a tree, which can be interpreted as a birth and death process. The connection between a tree and a stochastic process has been examined before. For instance, Harris [1952] studied curves generated by the depth-first traversal of trees. Such curves were instrumental in producing asymptotic results, e.g., related to a convergence of a stochastic process, but are not very well suited for comparing trees [Shen et al., 2014].

As noted by Wang and Marron [2007], a tree-structured data object can have both topological and geometric attributes. Topological attributes can be described generally as branching patterns, e.g., the number of nodes at any specific level. Geometric attributes could include distances between nodes, radiuses of edges, or angles between edges. In this paper we focus our attention on the length of edges. Here, we propose two functional representations of each tree-structured object encompassing its topological and geometric properties respectively. We define a quantile of tree objects by taking both properties into account; in particular, the quantile can be formulated as a solution of a multi-objective optimization problem. We also find empirical quantiles of tree distributions using a genetic algorithm.

This paper is organized as follows. In Section 3.2, we introduce two new functional representations for each unlabeled tree-structured object, which summarize the topological and geometric properties. In Section 3.3.1, we define a topological and a geometric median tree as a solution to the optimization problem. In Section 3.3.2 we introduce a novel notion of Pareto median tree object as a solution to the multi-objective optimization problem. Next, in Section 3.3.3 we extend this idea to define Pareto quantiles of tree objects. Section 3.4 provides a case study of a set of

neurons using our proposed methods. Finally, in Section 3.5, we examine our proposed method through a simulation study.

3.2 Data Object and its Curve Representation

3.2.1 Data

In this paper, our motivating example is a set of neuron cells from the brains of rodents. The original dataset consists of digital reconstructions of neurons obtained from an online inventory site `neuromorpho.org` [Ascoli et al., 2007] which includes more than 8000 neurons from various brain regions. For details on the data and data collection process, see Pyapali et al. [1998], Pyapali and Turner [1994, 1996]. Our primary interest centers on pyramidal neurons from two areas of hippocampus, regions CA1 and CA3. Here a set of $n = 187$ pyramidal neurons, including 119 and 68 from CA1 and CA3 regions respectively, is used. It is known that neurons from CA3 region receive input signals from other cells in the brain, while neurons from CA1 region form the output from the hippocampus.

In Figure 3.2, each subplot depicts a pyramidal neuron which has three major components, apical dendrites (colored in magenta), basal dendrites (colored in green), and a soma (colored in black) in between. The soma can be a single point or a line, and it is very small compared with apical and basal dendrites. In addition, the top row of Figure 3.2 shows three neurons from CA1, and the bottom row shows three neurons from CA3. From all six subplots, the basal dendrites seem to be shorter than the apical dendrites; whereas the difference in branching of both groups of neurons is apparent. In particular, the initial segments of apical and basal dendrites are shorter for CA1 neurons than those of CA3 neurons, and basal structures of CA3 neurons are larger than those in CA1 neurons.

In Section 3.2.2, we will discuss a tree representation of dendritic structures and further propose two new curve representations in Section 3.2.3.

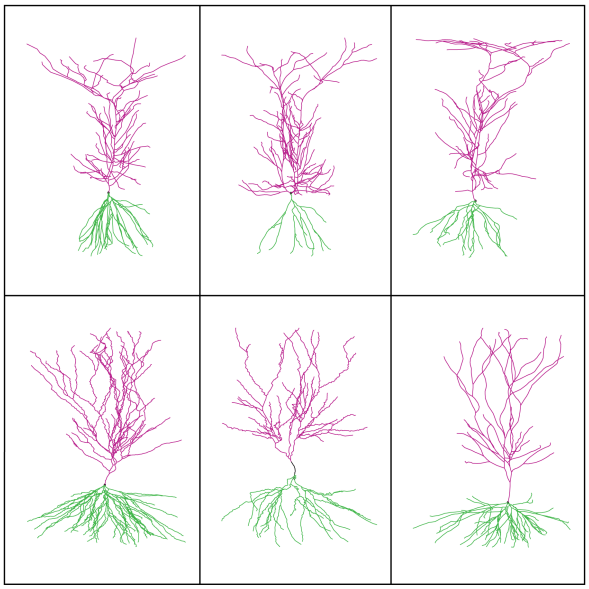

Figure 3.2: Graphical display of six neurons. Top row: three neurons from CA1 region of hippocampus; Bottom row: three neurons from CA3 region of hippocampus. In each subplot, apical dendrites are shown in magenta and basal dendrites are shown in green.

3.2.2 Graph as a Data Object

In mathematical graph theory, a *tree* is a simple graph with a set of nodes and edges, and there is a unique sequence of edges between any two nodes. A *forest* is a collection of trees. For any tree, the *root* is a specific node which can be designated based on the application. The level of a node is the number of edges of the path to the root node. For any two adjacent nodes connected by an edge, the node that is closer to the root node is called a parent node, and the other node is a child node. The node with no children is called a leaf node or a terminal node. For a tree object, if each node has at most two children, namely left child and right child, it is called a *binary tree*, and, if it has exactly two children, it is called a *full binary tree*.

In many scientific applications, binary trees have been used to model tree-structured objects. For instance, Wang and Marron [2007] proposed to use binary trees to represent human brain blood vessel systems. In our study, as can be seen in Figure 3.1, the apical dendrites emerge

from the apex of a soma, and branch like a single tree. Basal dendrites are somewhat different; in general, several dendritic trees grow out of the base of a soma, and form the basal dendrites. Here we model the apical dendrites as a binary tree and the basal dendrites as a forest of binary trees. In this paper, the term "forest" is referring to a disjoint union of binary trees.

We use a similar procedure, as discussed in Wang and Marron [2007], to construct a (binary) tree-structured *data object* for the apical dendrites. In particular, each dendrite segment between any two adjacent splits is denoted by a node. The initial dendrite segment till the first split point is the root node of the tree-structured object. Two nodes are connected by an edge if one dendrite segment branches off the other segment. The tree constructed in this way is a full binary tree. It is also worth mentioning that the resulting tree characterizes the *topological* property of the apical dendrites. More information about each dendrite segment (i.e., node) is also available including the thickness of the dendrite segment and the 3-dimensional coordinates of voxels along the dendrite. As suggested by Wang and Marron [2007], such information is the *geometric* property of a node, i.e., nodal attribute. For simplicity, we only consider the length of the dendrite segment as the nodal attribute, which is the basis of the functional representation established in the next section. We note for completeness that such binary trees with edge-lengths are known in the literature as binary Galton-Watson trees [Pitman, 2006] and their combinatorial and asymptotic properties have been extensively studied.

For the basal dendrites, each dendritic component can be represented as a tree-structured object using the aforementioned procedure. Consequently, the entire basal dendrites become a disjoint union of such binary trees, and indeed form a forest.

The procedure for constructing each binary tree is straightforward, but ambiguity may arise when identifying the left and right child nodes. Most recent work on tree-structured objects focused on sets of labeled trees. The term "labeled tree" is referring to a tree in which each node has a well specified label. In practice, as suggested by Aydın et al. [2009] and followed by Wang

et al. [2012], two approaches can be considered to establish labeling system, namely, *thickness correspondence* and *descendant correspondence*. In the first approach, at each split point, the thicker dendrite segment is denoted as the left child node of its parent node. As a contrast, in the second approach, the dendrite segment with more subsequent segments is denoted as the left child node. In Figure 3.3, a graphical comparison between two types of correspondence is provided. In panel (A), a synthetic tree is shown, and the corresponding dyadic tree representations using the thickness correspondence and descendant correspondence are shown in panel (B) and panel (C) respectively. The numbers associated with both trees are the level-order indices; see Wang and Marron [2007] for more details. It can be seen that, for the same tree, two types of correspondence

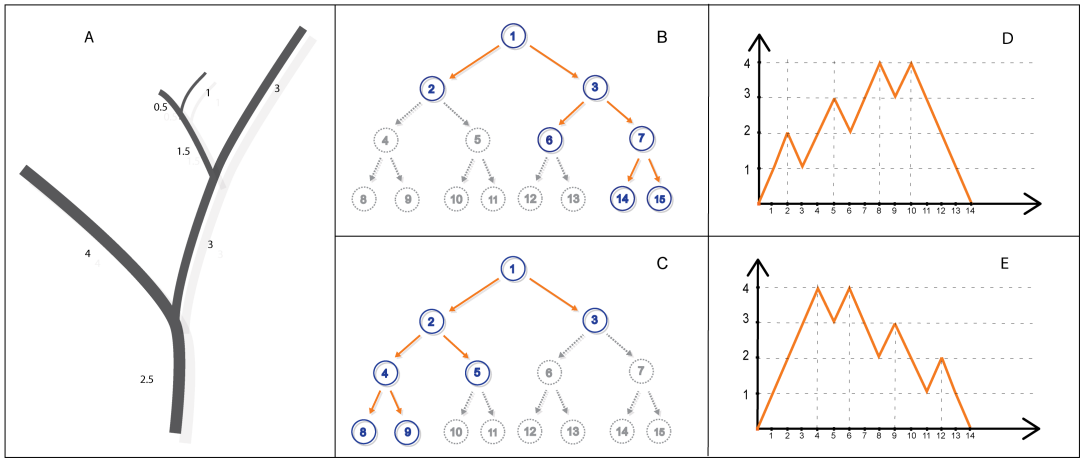

Figure 3.3: (A) Graphical illustration of a synthetic tree; (B) Dyadic representation using thickness correspondence; (C) Dyadic representation using descendant correspondence; (D) Harris path for the tree in (B); (E) Harris path for the tree in (C).

may result in quite different tree-structured objects with different labeling. Most existing methods for analyzing tree-structured data may reach different conclusions. In this paper, our main focus is a set of unlabeled trees or forests which has not been tackled before.

3.2.3 Curve Representations for Unlabeled Trees and Forests

For labeled binary trees, dyadic tree representation provides an intuitive way to visualize the topological property; see Figure 3.3. In practice, such representation is not suitable to depict a sample of tree-structured objects due to space limitation. In probability literature, tree-structured objects are usually modeled as branching processes. Harris [1952] established a correspondence, called *Harris correspondence*, between trees and random walks. For example, for the tree shown in panel (C) of Figure 3.3, a Harris path to visit all seven nodes is

$$① \to ② \to ④ \to ⑧ \to ④ \to ⑨ \to ④ \to ② \to ⑤ \to ② \to ① \to ③ \to ①$$

Note that this path is a *depth-first* search path of all nodes. It can be further illustrated as a curve. In particular, the number of steps required to visit each node along the Harris path is shown as the horizontal axis, and the level of the node is shown as the vertical axis. Two random walks are depicted in panel (D) and panel (E) of Figure 3.3. Note that, for the same tree-structured object, thickness correspondence and descendant correspondence may result in quite different random walks. Recently, Shen et al. [2014] used descendant correspondence to pre-process each tree and then obtained a corresponding Harris path.

The Harris path provides insightful information regarding the topological property of a single tree-structured object. The alignment issue arises when comparing Harris paths obtained from a set of trees. In Shen et al. [2014], the authors proposed a modified Harris path (a.k.a. Dyck path) to overcome this problem. In particular, they introduced a pre-specified (tree) parameter which controls the depth of search. The effect of such parameter remains unclear. Shen et al. [2014] further proposed a *branch length representation*. Each node is represented by the branch number and the length of the segment. The authors have conducted principal component analysis on the set of Dyck paths and the set of branch length curves, and certain important scientific findings have been reported. The success of their approach relies on the descendant correspondence and

the corresponding labeling system of binary trees. However, we might reach different conclusions using the same data and different types of correspondence. This issue becomes even more serious when the data objects are forests. When comparing two forests with different numbers of tree components, a well-defined order is usually not available. To circumvent this problem, we propose a new tree/forest representation which is independent of the choice of correspondence and the labeling system. Moreover, certain nodal attribute, e.g., the length of each segment, can also be incorporated in this new representation.

For a tree, we introduce a function $g(x), x \in [0, \infty)$ defined as the number of distinct points at distance x from the root. An illustrative example is given in the panel (C) of Figure 3.4. Such function $g(x)$ provides a *geometric curve representation* of a tree-structured object. Note that $g(x)$ is a piecewise constant function with $g(0) = 0$ and $g(\infty) = 0$. In particular, $g(x)$ is left continuous on $(0, \infty)$. One can also notice that the number of jumps in the range $(0, \infty)$ represents the number of nodes in the tree, a positive jump corresponds to an internal node, a negative jump corresponds to a leaf. It is worth mentioning that, for the tree with topology only, we can also obtain its *topological curve representation*, denoted here by $\ell(x)$, by assuming all segments have length equal to 1; see panel (B) of Figure 3.4. Here, the number of nodes in the tree can be retrieved as $\sum_{i=1}^{\infty} \ell(i)$, which is always an odd number. An equivalent definition of the curve representation is the number of intersections of a ball with "radius" x and the tree itself. In contrast to Harris path, the tree curve mimics the *breadth-first search* algorithm in graph theory in the sense that we would like to count the number of branches at any given radius x.

In general, for a forest with k distinct trees, the curve representation is defined as the sum of $g^{(1)}(x), \ldots, g^{(k)}(x)$, where $g^{(i)}(x)$ is the curve representation associated with the ith tree.

In our study, we will represent both apical and basal dendrites using tree curves. For our convenience, we will display a *joint tree curve* for both apical and basal dendrites. Specifically, we show a tree curve for the apical dendrites and the mirror-view of a tree curve for the basal

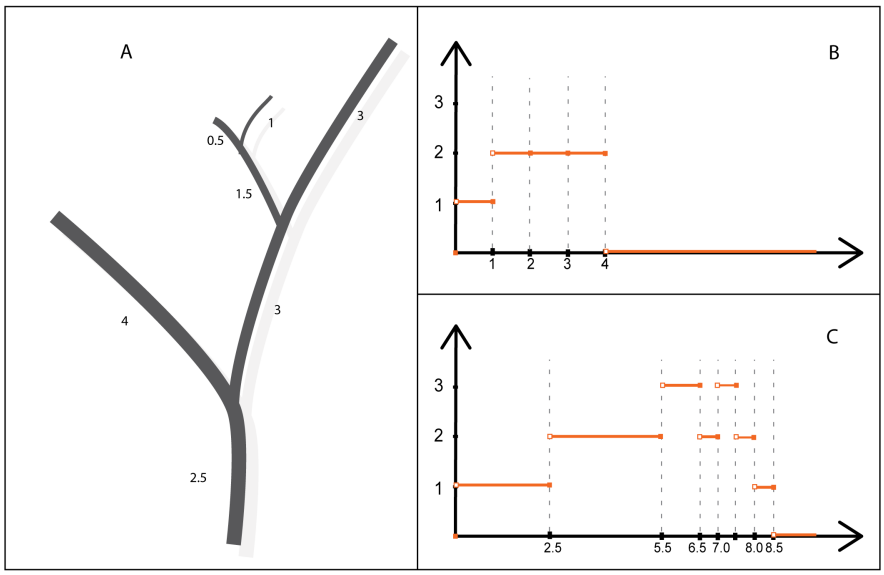

Figure 3.4: An example of tree (left) and its curve representations (right): the topological curve (B), the geometric curve (C).

dendrites in one plot. An example of joint tree curves is given in Figure 3.5. Here the raw data is depicted in the upper panel, the corresponding (joint) geometric and topological tree curves are depicted in the middle and the bottom panels.

3.2.4 Equivalence Classes of Topology and Geometry

For each tree or forest, the geometric tree curve provides a functional representation which, in fact, is *not* a one-to-one mapping from the space of binary trees or forests to the space of (piecewise constant) functions. In Figure 3.4, given a tree curve $g(x)$ in panel (C), we can reconstruct a tree; however, such reconstruction is generally not unique. Two trees, say t_1 and t_2, are geometrically *equivalent* if they have the same geometric tree curve, and hence can be written as $t_1 \overset{\mathtt{G}}{\sim} t_2$. The geometric equivalence class of tree t is the set of trees that are equivalent to t and is denoted by $[t]_{\mathtt{G}}$. Analogously, we define the topological equivalence class of a tree t, and denote it by $[t]_{\mathtt{T}}$. All trees in $[t]_{\mathtt{G}}$ and $[t]_{\mathtt{T}}$ have the same number of nodes, which equals to $2m_t + 1$, including m_t internal nodes and $m_t + 1$ leaves.

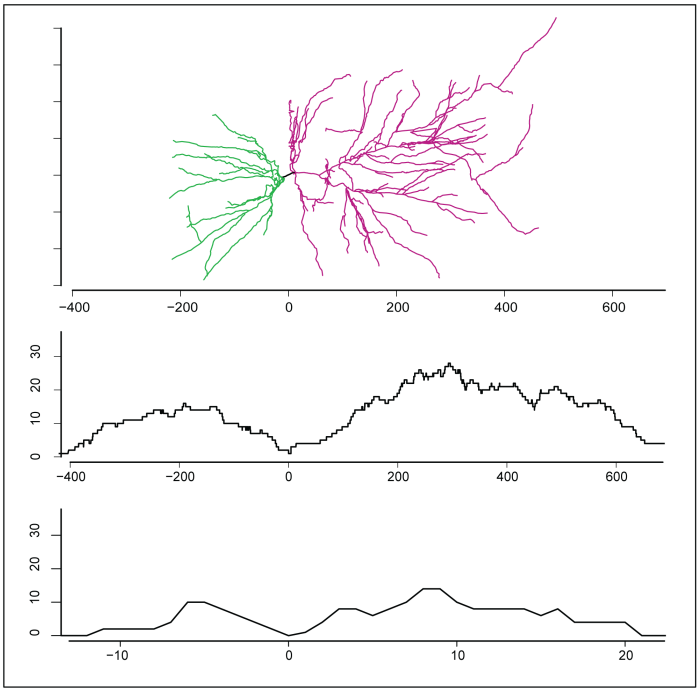

Figure 3.5: A graphical display of a joint geometric tree curve (middle) and topological (bottom) for the corresponding neuron object (top) with apical dendrites (colored in magenta) and basal dendrites (colored in green).

Next, we will define an operation, called *implant*, for trees and forests. In particular, for any tree (or forest) t, an *implant* of t is defined by swapping any two subtrees at the same distance from the root. Note that, for topological equivalence, the level plays a role of a distance. It can be seen that two trees (forests) are equivalent if and only if one tree (forest) can be obtained by a sequence of implant operations from the other. Thus, there is not a unique tree reconstruction from a tree curve, or even from geometric and topological curves combined. In this paper, we often reconstruct a tree with the procedure as described in Section 3.5.3.

3.3 Methodology

3.3.1 Median Trees and L_1 Distance

The notion of *median* tree has been previously studied by Phillips and Warnow [1996], Banks and Constantine [1998] and Wang and Marron [2007]. In Phillips and Warnow [1996] and Banks and Constantine [1998], the authors developed median trees for a set of classification trees. Wang and Marron [2007] took a first step to consider a set of tree-structured objects motivated by medical imaging analysis. In particular, for a sample of binary trees, $t_1, \ldots, t_n$, the authors proposed a (topological) median as the minimizer tree of

$$\min_t \sum_{i=1}^{n} d_I(t, t_i), \tag{3.1}$$

where d_I is the *integer tree metric*, defined in (3.1) of Wang and Marron [2007], for labeled binary trees. This notion of center point in tree space can be viewed as a special case of Fréchet median [Wang et al., 2012]. For general metric space, Fréchet [1948] proposed to define the center point, namely, Fréchet median, as the minimizer of (3.1) for any given metric.

In Section 3.2.3, two curve representations, topological and geometric tree curves, have been proposed for unlabeled trees. Consequently, an intuitive idea to measure the distance between two unlabeled trees is to use the L_1 metric between the corresponding curves. Note that each equivalence class has a unique curve representation. Thus, the L_1 metric between tree curves in fact provides a distance between equivalence classes of trees.

First, we will consider topological tree curves. For any two trees s and t with topological tree curves $\ell_s(x)$ and $\ell_t(x)$, the distance between the equivalence classes $[s]_{\mathtt{T}}$ and $[t]_{\mathtt{T}}$ is defined as

$$d([s]_{\mathtt{T}}, [t]_{\mathtt{T}}) = ||\ell_s(x) - \ell_t(x)||_1 \equiv \int_0^\infty |\ell_s(x) - \ell_t(x)| dx. \tag{3.2}$$

Theorem 1 establishes the connection between the L_1 distance in (3.2) and the integer tree metric of Wang and Marron [2007].

Theorem 1. For any two trees s and t, we have

$$d([s]_{\mathrm{T}}, [t]_{\mathrm{T}}) = \min_{s' \in [s]_{\mathrm{T}}, t' \in [t]_{\mathrm{T}}} d_I(s', t'). \tag{3.3}$$

From now on, let $\{t_1, \ldots, t_n\}$ be a random sample of trees, and let $\ell_i(x)$ be the topological curve representation of t_i. Similar to (3.1), we can formulate the median tree through an optimization problem described as

$$\min_{\ell(x)} \sum_{i=1}^{n} ||\ell(x) - \ell_i(x)||_1, \tag{3.4}$$

where $\ell(x)$ runs over the collection of topological curves.

If we relax the constraint in (3.4) and consider all possible functions $\ell(x)$, the solution is the pointwise median function, i.e., $m_0(x) = \text{median}\{\ell_1(x), \ldots, \ell_n(x)\}$. When n is odd, such pointwise median function is always unique. When n is even, the pointwise median function may not be unique for some x, and $m_0(x)$ takes the smallest value to break the tie. In Theorem 2, we will prove that such pointwise median function $m_0(x)$ corresponds to an equivalence class in which all elements are called the topological median trees.

Theorem 2. Assume that $\{t_1, \ldots, t_n\}$ is a sample of trees with finite levels. Let $\ell_i(x)$ be the topological curve representation of t_i. The pointwise median $m_0(x)$ corresponds to an equivalence class of trees, and hence is the minimizer of (3.4).

Our primary interest is a sample of trees with nodal attributes, e.g., the lengths of dendritic segments. In literature, for trees with nodal attributes, Wang and Marron [2007] proposed a median-mean tree, whose topology is determined by the topological median and nodal attributes can be obtained by averaging corresponding nodal attributes. For a set of unlabeled trees, their notion of "median-mean" cannot be generalized. In this paper, enlightened by (3.4), for a sample of trees $t_1, \ldots, t_n$ with geometric tree curves $g_1(x), \ldots, g_n(x)$ respectively, the geometric median tree can be defined through

$$\operatorname*{argmin}_{g(x)} \sum_{i=1}^{n} \|g_i(x) - g(x)\|_1, \tag{3.5}$$

where $g(x)$ runs over all possible geometric tree curves. Similar to Theorem 2, we will show that the pointwise median, denoted as $m_1(x)$, is a geometric tree curve.

Theorem 3. A pointwise geometric median of a finite sample of piecewise constant functions $g_i(x)$ represents a valid tree class.

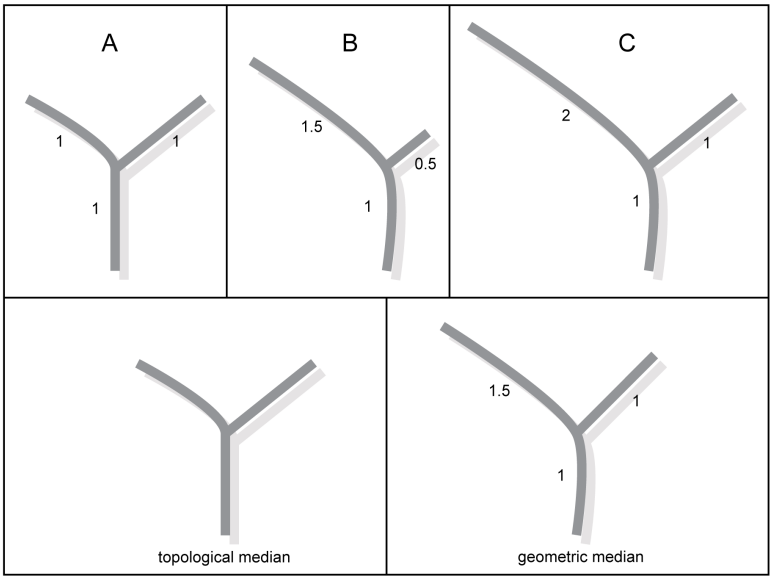

Figure 3.6: A graphical display of the topological median (lower-left panel) and geometric median (lower-right panel) of a sample of three tree-structured objects (top row). The number associated with each branch segment is the segment length, and is referred to as a geometric attribute. Here both median trees have the same topological structure with three branch segments.

To better illustrate the topological and geometric median trees, we will consider two examples, as shown in Figures 3.6 and 3.7. In each figure, a sample of three tree-structured objects are depicted in the top row (panels A-C). In Figure 3.6, three trees have the same topological structure, including one root segment and two offspring segments among which one is relatively longer than the other one. The topological and geometric median trees are displayed in the lower-left and lower-right panels. It can be seen that the topological median also has the same topology as all three trees. From the geometric median tree, it can be seen that two offspring segments have unequal length. In Figure 3.7, trees A and B have the same topology, and tree C has more

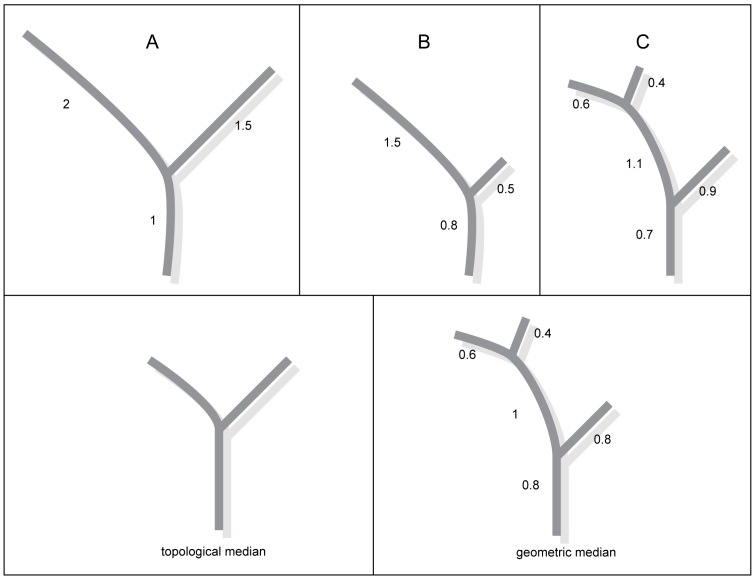

Figure 3.7: A graphical display of the topological median and geometric median of a sample of three tree-structured objects. Topological Median is the same as shown in Figure 3.6.

segments than the other two trees. Surprisingly, the topological median and geometric median have different tree structures. The reason is that the topological median only characterizes the centrality of topological properties, while the geometric median tree is influenced by the length of segments.

In the next section, we will introduce a new notion of median, called Pareto median, which will take both topological and geometric information into consideration.

3.3.2 Pareto Median Trees — A Multi-Objective Approach

We continue to let $\{t_1, \ldots, t_n\}$ be a random sample of trees. Let $\ell_i(x)$ and $g_i(x)$ be the topological and geometric curve representations of a tree t_i, respectively.

In Figure 3.8, a sample of 21 trees is depicted. All trees have the same simple topology, a trunk and two branches, and randomly-generated geometric attributes. The topological and geometric median trees are shown in panels (B) and (C). It can be seen that the geometric median has a more complex topological structure than the topological median. The complexity of the geometric

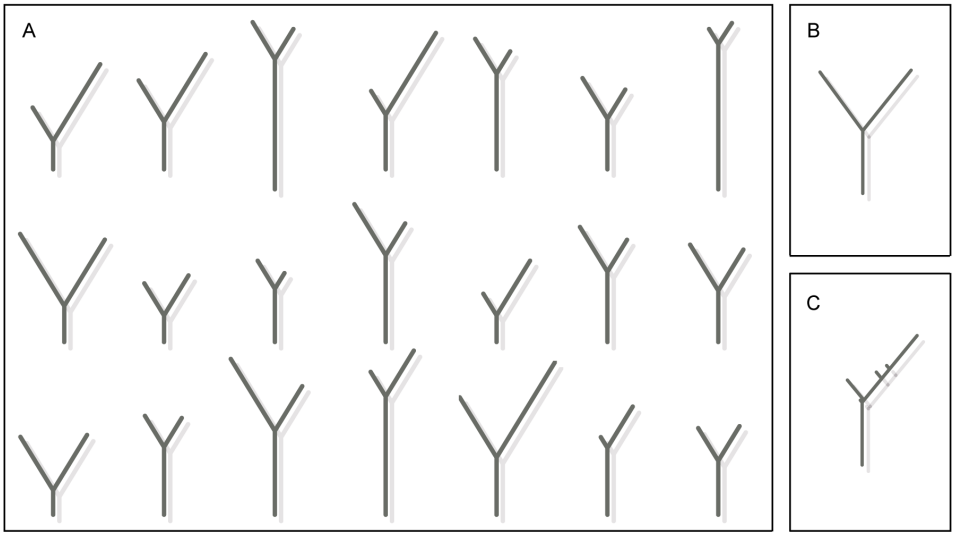

Figure 3.8: (A) Graphical display of a sample of 21 simulated trees with the same topology and different geometric attributes; (B) Topological median tree; (C) Geometric median tree.

median reflects the diversity of geometric attributes of the tree set. By contrast, the topological median is the manifestation of the topological homogeneity of the data. Preferably, we would like to find a median tree that takes into account both topological and geometric attributes together. In other words, we would like to find a tree to minimize both (3.4) and (3.5) simultaneously, which, in fact, is a multi-objective optimization problem. Mathematically, it can be formulated as

$$\min_t(T_n(t), G_n(t)) \tag{3.6}$$

where t runs over the space of binary trees,

$$T_n(t) = \sum_{i=1}^{n} ||\ell_t(x) - \ell_i(x)||_1 \quad \text{and} \quad G_n(t) = \sum_{i=1}^{n} ||g_t(x) - g_i(x)||_1$$

Here, $\ell_t(x)$ and $g_t(x)$ are the topological and geometric tree curves of t, respectively.

In multi-objective optimization, there is no guarantee of the existence of a solution which minimizes both $T_n(t)$ and $G_n(t)$. An alternative is the Pareto optimum; see Coello et al. [2007] for a formal definition. Pareto set contains all feasible solutions such that there is no other solution

that improves one of the criteria without worsening another. In other words, a Pareto optimal set is a set of feasible solutions which are not dominated by any other solution.

In our problem considered here, for any two trees s and s', s' is *dominated* by s if $T_n(s) \leq T_n(s')$ and $G_n(s) \leq G_n(s')$, and at least one inequality is strict. In addition, a tree is *Pareto optimal* or *Pareto median tree*, if it is not dominated by any other trees. Let $\mathcal{P}$ be the collection of all Pareto median trees. There are two *trivial* Pareto median trees, namely topological Pareto median and geometric Pareto median. For the topological Pareto median, we minimize $G_n(t)$ among the subclass of trees whose topological tree curves minimize $T_n(t)$. On the other hand, for the geometric Pareto median, we minimize $T_n(t)$ among the subclass of minimizer trees of $G_n(t)$. If a common solution exists, it is called an *ideal* tree.

Recall that, in Section 3.3.1, the topological median tree and the geometric median tree are defined by minimizing $T_n(t)$ and $G_n(t)$ respectively. It is worth mentioning that the geometric median and the geometric Pareto median have the same geometric curve; however, in terms of topology, the former is less restrictive than the latter. This is due to the fact that the geometric Pareto median minimizes T_n within a subclass of trees possessing the same geometric tree curve. Similarly, the topological median and the topological Pareto median have the same topology, but the former can have any geometric properties or attributes, and the latter has the geometry that minimizes G_n in a subclass of trees possessing the same topological curve.

For the example in Figure 3.8, there are four elements in the Pareto optimal set. All four Pareto median trees are shown in Figure 3.9. In particular, the first tree is the geometric Pareto median, and the last tree is the topological Pareto median. In addition, for each Pareto median tree, the corresponding values of T_n and G_n are depicted as a point in Figure 3.9. In this example, there is a unique tree corresponding to each pair of T_n and G_n, but it is not the case in general.

The geometric Pareto median can be found efficiently using convex optimization techniques; see Antoniou and Lu [2007]. However, the number of topologies grows with the number of nodes

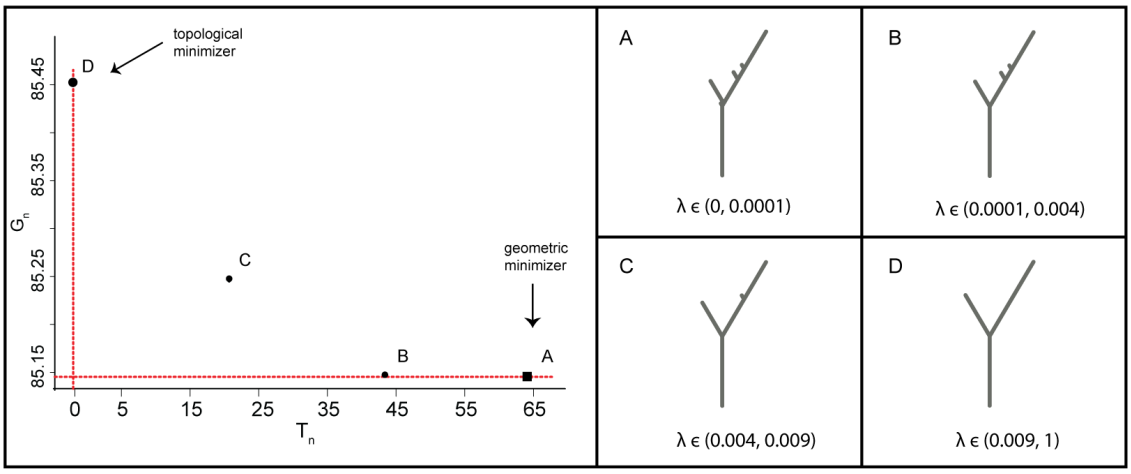

Figure 3.9: A graphical display of the T_n-values (horizontal axis) and G_n-values (vertical axis) for all Pareto median trees. The solutions form a *Pareto front*. In particular, the tree objects corresponding to those values are shown in panels (A)-(D).

in the tree. Finding an optimal topology can be viewed as a combinatorial problem of high dimension, which can be efficiently solved using genetic algorithm. In general, computation of a multi-objective optimization problem (3.6) can be very complicated. However, as will be seen in Section 3.3.4, after appropriate modification, genetic algorithm can be used to find all Pareto solutions.

It is worth mentioning that, in the literature, a widely-used approach to multi-objective optimization is the *weighted-sum* method [Coello et al., 2007]. Specifically, consider the following optimization problem, for $0 < \lambda < 1$,

$$\min_t \lambda T_n(t) + (1-\lambda) G_n(t) \tag{3.7}$$

This criterion is a linear combination of the T_n and G_n. It is expected that its solution will take both topology and geometric information into consideration. Moreover, note that the solutions of (3.7) are Pareto optimal of (3.6), hence are Pareto median trees. For a known λ, (3.7) is a single-objective optimization problem, which can be efficiently solved using the standard genetic

algorithm. In addition, a range of λ may correspond to a single Pareto optimal solution. This can be observed in Figure 3.9, where for each of four Pareto trees, the corresponding range of parameter λ is specified. In particular, the topological Pareto median corresponds to the largest λ, and the geometric Pareto median corresponds to the smallest λ. However, the multi-objective optimization in (3.6) is not equivalent to the single-objective optimization in (3.7); that is, for some Pareto optimal trees, there is no corresponding λ such that (3.7) holds. In literature, given a preselected λ, the weighted sum method is often referred to as an *a priori* method, in contrast to the *a posteriori* method that finds many Pareto solutions, and selects the best solution after the search is completed [Coello et al., 2007]. In this paper, our computational approach will yield all or most of the Pareto solutions.

3.3.3 Pareto Quantiles of Unlabeled Trees

In this section, we will extend the notion of Pareto median trees to *Pareto quantile trees*. To motivate our discussion, we fist consider a random variable X. Finding the sample quantile of X, based on a random sample $\{X_1, \ldots, X_n\}$, can be formulated as an optimization problem

$$\underset{x}{\operatorname{argmin}} \sum_{i=1}^{n} \rho_\tau(X_i - x),$$

where $\rho_\tau(z) = z(\tau - I(z < 0))$ and $\tau \in (0, 1)$. See Koenker and Hallock [2001] for more details.

Here, we consider a set of random tree objects rather than random variables. Enlightened by the problem above, we can generalize the formulation in (3.6) and define the Pareto quantiles through a multi-objective optimization problem; that is,

$$\min_t (T_n^\tau(t), G_n^\tau(t)) \tag{3.8}$$

where

$$T_n^\tau(t) = \sum_{i=1}^{n} \int_X \rho_\tau(\ell_i(x) - \ell_t(x))dx \quad \text{and} \quad G_n^\tau(t) = \sum_{i=1}^{n} \int_X \rho_\tau(g_i(x) - g_t(x))dx.$$

In the special case of $\tau = 0.5$, this problem is equivalent to (3.6), and yields the Pareto median trees.

Similar to a topological and geometric median, we first minimize $T_n^\tau(t)$ and $G_n^\tau(t)$ individually, to obtain topological and geometric quantiles. The analogs of Theorems 2 and 3 also hold, and are stated as follows. Both theorems play essential roles in the identification of Pareto quantiles.

Theorem 4. Assume that $\{t_1, \ldots, t_n\}$ is a sample of trees with finite levels. Let $\ell_i(x)$ be the topological curve representation of t_i. The pointwise topological quantile of $\ell_i(x)$ represents a valid tree.

Theorem 5. A pointwise geometric quantile of a finite sample of piecewise constant functions $g_i(x)$ represents a valid tree.

In general, for a sample of forests, it can be shown that a pointwise quantile represents a valid forest. Next, same as (3.6), the existence of the minimizer of (3.8) is not guarenteed. Here, we intend to find the Pareto optimal set for (3.8). Each element in this Pareto optimal set is called a 100τ-th Pareto quantile. Using the similar idea of the topological and geometric Pareto medians, there are two trivial elements in the Pareto optimal set of (3.8), namely, topological Pareto quantile and geometric Pareto quantile. For illustration, the Pareto optimal sets for the 25th and 75th quantiles for the example from Figure 3.8 are depicted in Figure 3.10. Both sets consist of just two solutions, the geometric Pareto quantile and the topological Pareto quantile. Note that, for this toy example, all solutions can be obtained using the weighted sum method, in a similar fashion as defined in (3.7).

3.3.4 Genetic Algorithm

The genetic algorithm provides a useful tool to solve combinatorial problems that do not have an analytic solution. By imitating the mechanism of a genetic selection acting on chromosomes

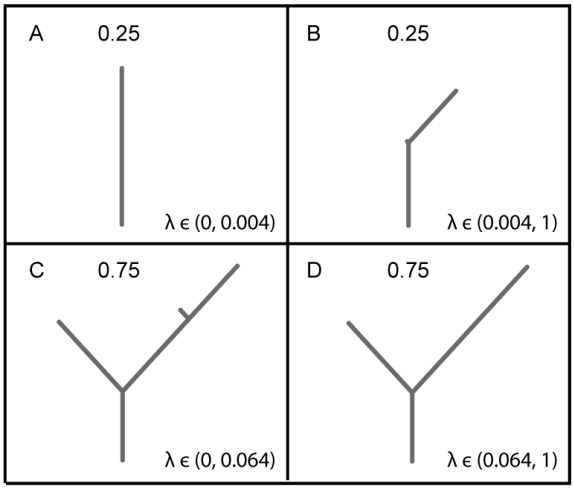

Figure 3.10: Top row: Pareto set for the 25th quantile; bottom row: Pareto set for the 75th quantile. Trees A and C are geometric Pareto quantiles, and trees B and D are topological Pareto quantiles.

and genes, the algorithm finds the *fittest* elements in the population. The interpretation of fitness depends on the optimization goal. In single-objective optimization, there is generally one best element. In multi-objective optimization, there is a set of best elements defined through non-dominance as discussed in Section 3.3.2. The algorithm starts with an initial population of trees, designed to provide sufficient *genetic diversity* for the natural selection to work, and creates new individuals stochastically via random *cross-overs* and *mutations* applied to the fittest (and occasionally less fit) elements of the previous generation. Theoretical results regarding the convergence of genetic algorithm are based on the schema theory [Goldberg, 1989]. The practical advice on the algorithm design is available in Sivanandam and Deepa [2007]. In general, the genetic algorithm performs better than a random search, and it does so by exploiting accumulated information about the features that improve overall capabilities of the organisms.

The implementation of the genetic algorithm to our multi-objective optimization problems is non-trivial and poses a significant challenge. It requires some important elements of the design, including the encoding of the population of geometric tree curves as *chromosome*-like strings, and

the definition of cross-over and mutation over those tree curves. Note that a mutation could create, delete or move a single branch, and a cross-over might merge two subtrees from two parents. The algorithm can also operate on geometric curves representing forests without significant changes.

In Section 3.3.2, for a known λ, we consider a single-objective optimization problem (3.7) instead of a multi-objective optimization problem (3.6). From a practical viewpoint, there is no additional complexity in designing a multi-objective algorithm compared to a single-objective algorithm, except for a potentially very large set of optimal solutions. For the toy example in Figure 3.8, the genetic algorithm finds the entire Pareto set of median trees (Figure 3.9) as well as 25th and 75th quantile trees (Figure 3.10). For the real data, the problem is much more complex. Figure 3.11 shows a collection of solutions found in a single run of the algorithm.

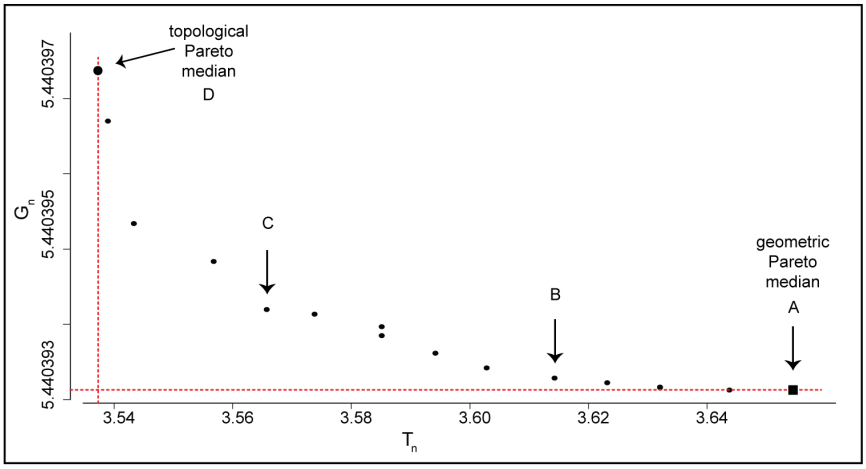

Figure 3.11: A graphical display of the Pareto set for the median of apical dendritic trees from CA1. The solutions form a Pareto front. The geometric Pareto median (A) and topological Pareto median (D) as well as two other Pareto medians, (B) and (C), are highlighted. All four are depicted in Figure 3.14. Here both axes are shown in log-scale.

3.4 Real Data Analysis

In this section, our proposed methods are applied to a set of pyramidal neurons as described in Section 3.2.1. This data set consists of 119 digital reconstructions of neurons from CA1 and

68 from CA3 of the hippocampus. In general, each neuron consists of approximately $3,000$ interconnected voxels, and each voxel is associated with a type (soma, axon, basal dendrite, apical dendrite) and a radius (not used in this study). The first step of the analysis involves the extraction of the tree object for both dendritic structures of every neuron. The geometric properties of each branch can be extracted in multiple ways. Ascoli and Krichmar [2000] provided a comprehensive survey of studies of the relationship between branch length, branch radius at bifurcation points and branching angles. The authors pointed out that approximating a branch length by a straight line between the bifurcation points leads to a much smaller tree. An alternative approach is to approximate the branch length by summing the distances between voxels in each branch, which could potentially lead to a larger tree. Here, we take the latter approach. The topological and geometric curve representations can be constructed based on the digitally reconstructed neurons.

Figure 3.12 shows both geometric (left column) and topological (right column) curve representations for all neurons in CA1 (top row) and CA3 (bottom row) regions. In each panel, joint tree curves, as defined in Section 3.2.3, are depicted. It can be observed that neurons from the CA3 region have a much more developed basal section (left portions of the tree curves) than neurons from CA1. The apical sections of neurons from both regions differ substantially. In panels (A) and (C), the geometric apical tree curves from CA1 (the right portions of the curves) are longer than the ones form CA3. Specifically, many CA1 tree curves are longer than 1000 (micrometers), whereas most CA3 tree curves are less than that. In addition, the largest branch counts for tree curves from CA1, on y-axis, are bigger than the branch counts for tree curves from CA3. The topological curves of the apical trees, in panels (B) and (D), indicate that apical trees from CA1 are taller than those from CA3. In fact, many CA1 curves reach levels 30 or higher, while all apical trees from CA3 end before level 20.

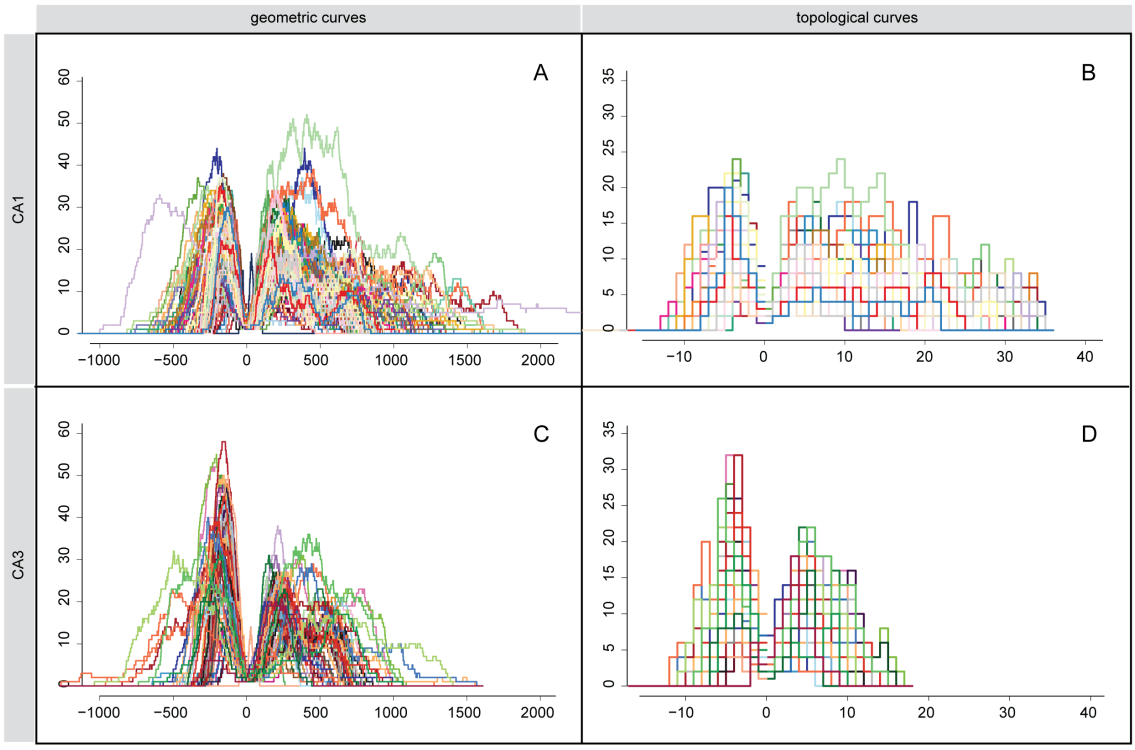

Figure 3.12: (A) Joint geometric curves for all neurons from CA1 region; (B) Joint topological curves for all neurons from CA1 region; (C) Joint geometric curves for all neurons from CA3 region; (D) Joint topological curves for all neurons from CA3 region.

For each choice of τ, we implement the genetic algorithm, as discussed in Section 3.3.4, to obtain quantiles of apical and basal dendritic trees. For instance, in Figure 3.11, the Pareto set for the median apical dendritic trees is depicted. Each element in the Pareto set will correspond to a Pareto median tree. In particular, the topological Pareto median and geometric Pareto median are highlighted in Figure 3.11, their corresponding tree representations are included in Figure 3.13.

Recall that each Pareto solution consists of two curves, a topological curve and a geometric curve, and for each pair, a tree can be reconstructed following the procedure outlined in Section 3.5.3. The geometric curves corresponding to both Pareto medians, in panels (B) and (F) of Figure 3.13, are very similar. The topological curves, in panels (A) and (E), reveal some topo-

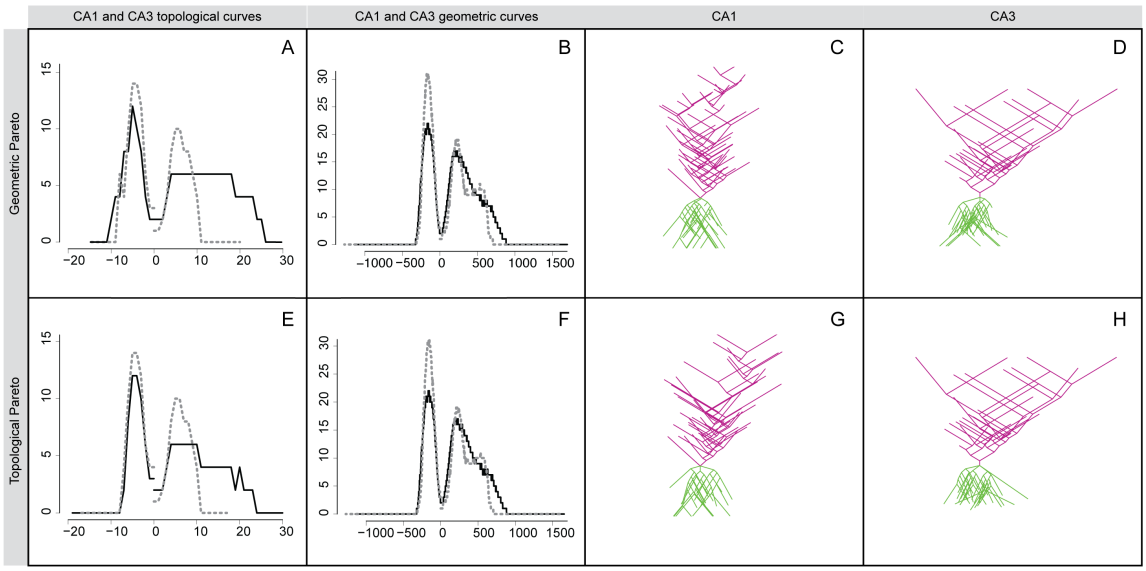

Figure 3.13: Graphical display of topological and geometric tree curves, as well as corresponding tree objects, for geometric Pareto median (top row) and topological Pareto median (bottom row). (A) topological tree curves for geometric Pareto median from CA1 (solid) and CA3 (dashed); (B) geometric curves for geometric Pareto median from CA1 (solid) and CA3 (dashed); (C) tree object corresponding to CA1 geometric Pareto median; (D) tree object corresponding to CA3 geometric Pareto median; (E) topological tree curves for topological Pareto median from CA1 (solid) and CA3 (dashed); (F) geometric curves for topological Pareto median from CA1 (solid) and CA3 (dashed); (G) tree object corresponding to CA1 topological Pareto median; (H) tree object corresponding to CA3 topological Pareto median.

logical differences between Pareto medians; in particular, around level 10, the geometric Pareto median tree tends to have more branches than the topological Pareto median.

In panel (B), the joint curves for geometric Pareto median trees from CA1 (solid line) and CA3 (dashed line) are compared. It can be seen that, for neurons from CA1 and CA3 regions, apical dendrites are longer than basal dendrites, but basal dendrites have higher branching maxima than apical dendrites. Furthermore, the median basal dendrites from both regions are of the same overall length, but basal dendrites from CA3 have substantially more branches in the middle section. In fact, the maximal number of branches for both basal dendrites occurs roughly at the same distance from the root. However, for apical dendrites, different lessons are learned. The median geometric curves, positive portions in panel (B), are closer in maximum branch count

(y-axis) but not in tree height (x-axis). The CA1 median tree is slightly longer, and the maximal numbers of branches for both apical dendrites are aligned at the same distance. Panels (C) and (D) contain simplified depictions of geometric Pareto median trees from CA1 and CA3.

In panel (E), the topological Pareto median curves from CA1 (solid line) and CA3 (dashed line) are compared. The basal topological median trees have the same height (x-axis), but CA3 maxima are larger than CA1 maxima. In fact, the median basal dendrites for both CA1 and CA3 regions are forests with three and four trees respectively. In contrast, the apical dendrites differ considerably in topology. The CA3 apical median tree is much shorter, and it reaches the maximum number of branches at about level 5, and from there, the number drops steeply. The CA1 median tree exhibits a different growth pattern. The branch maximum is lower than that of CA3, but the number diminishes slowly, which results in a much higher tree. Panels (G) and (H) contain simplified depictions of topological Pareto median trees from CA1 and CA3. Figure 3.13 shows just two members of the Pareto optimal set, the geometric Pareto median and the topological Pareto median. Figure 3.14 shows two additional reconstructed Pareto median trees from region CA1. All trees are indeed similar.

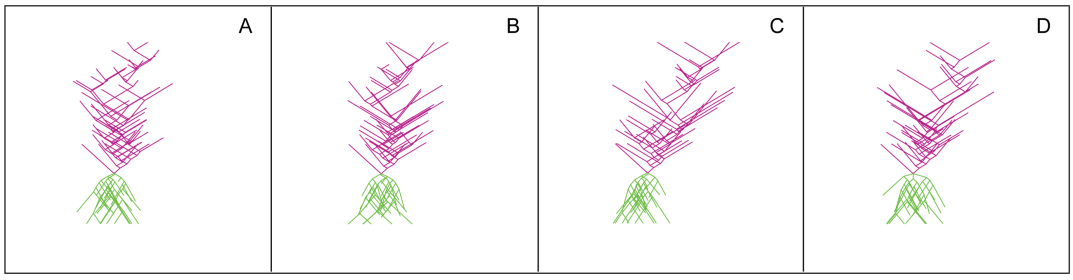

Figure 3.14: Pareto median trees from CA1, including the geometric Pareto median (panel A), the topological Pareto median (panel D), and two other Pareto median trees (panels B and C). The T_n and G_n values for those four median trees are highlighted in Figure 3.11.

As a comparison, we consider two alternative methods for obtaining the median tree based on node labeling using descendant correspondence [Shen et al., 2014]. In Figure 3.15, three

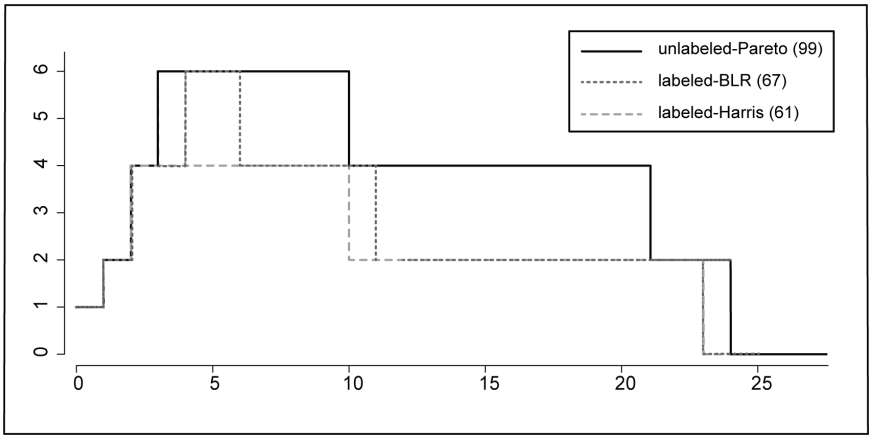

Figure 3.15: Topology for the median apical dendrite from CA1: pointwise true value (solid line) and two labeled methods (dotted and dashed lines). Overall numbers of branches are shown in brackets. Labeled methods underestimate the number of dendritic segments.

different topological medians for apical dendritic trees from CA1 region are depicted, including our proposed method (solid line), BLR (dotted line) and modified Harris paths (dashed line). The plot shows the number of dendritic branches (y-axis) at each level (x-axis) and the overall number of branches. It can be seen that both BLR and modified Harris paths for labeled trees yield trees with fewer dendritic segments, which essentially underestimates the topological complexity.

Next we implement our proposed method to compute both topological and geometric Pareto quantiles. Fgure 3.16 shows the 10th Pareto quantiles of neurons from CA1 and CA3 regions. The 10th quantile trees are much smaller and simpler, but they exhibit some characteristics that we observed in the median trees. In particular, the basal trees are shorter than apical trees, and have higher branching maxima. In addition, the basal trees from CA1 and CA3 are similar, both in geometry (panels B and F) and topology (panels A and E), but the basal trees from CA3 have slightly larger branching maxima than CA1 basal trees. The apical quantiles reveal more conspicuous differences between CA1 and CA3 regions. In geometry, depicted in panel (B and F), the apical tree from CA1 is slightly more complex than the apical tree from CA3; specifically it is longer and has more branches at all distances. In topology, depicted in panels (A) and (E), the

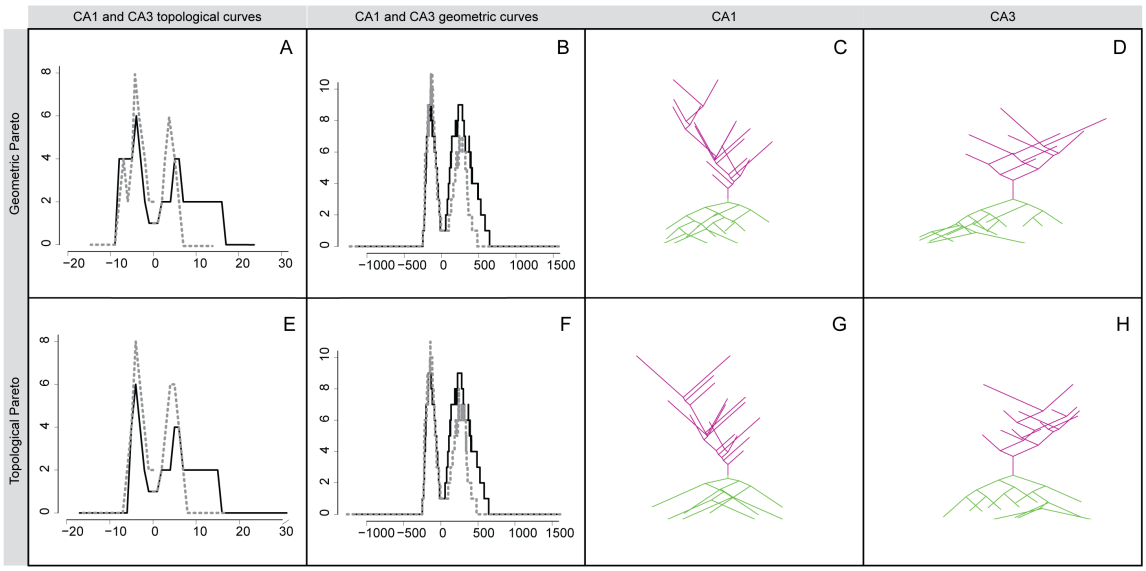

Figure 3.16: Graphical display of topological and geometric tree curves, as well as corresponding tree objects, for geometric 10th Pareto quantile (top row) and topological 10th Pareto quantile (bottom row). (A) topological tree curves for geometric 10th Pareto quantile from CA1 (solid) and CA3 (dashed); (B) geometric curves for geometric 10th Pareto quantile from CA1 (solid) and CA3 (dashed); (C) tree object corresponding to CA1 geometric 10th Pareto quantile; (D) tree object corresponding to CA3 geometric 10th Pareto quantile; (E) topological tree curves for topological 10th Pareto quantile from CA1 (solid) and CA3 (dashed); (F) geometric curves for topological 10th Pareto quantile from CA1 (solid) and CA3 (dashed); (G) tree object corresponding to CA1 topological 10th Pareto quantile; (H) tree object corresponding to CA3 topological 10th Pareto quantile.

branching pattern of CA1 apical trees is different from that of CA3 apical trees. Although apical trees from CA1 have more branches at any distance from the root, as shown in geometric curves in (B) and (F), these branches are distributed differently, and more of them are at higher levels of trees, and fewer at lower levels. The simplified depictions of corresponding trees, in panels (G) and (H), confirm these observations.

These characteristics are even more pronounced in the 90th quantile trees; see Figure 3.17. Comparing the 90th quantile to lower quantiles, one can observe that CA1 apical trees gained most branches further from the root and on higher levels, while CA3 apical trees gain most branches

closer to the root and on lower levels. In contrast, basal 90th quantile trees grew more in width (higher branching maxima) than in length or depth.

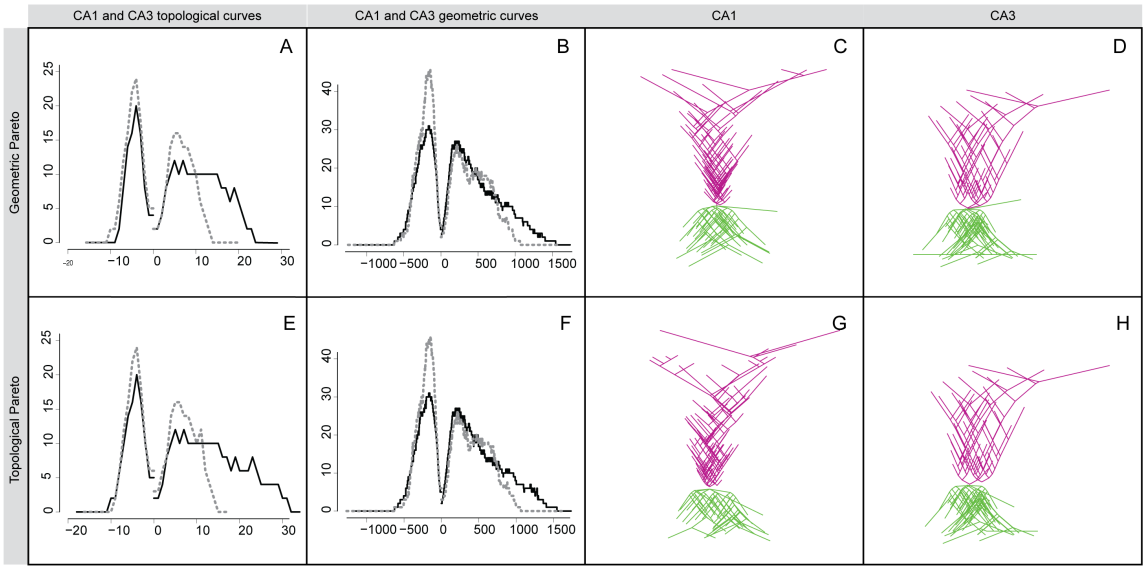

Figure 3.17: Graphical display of topological and geometric tree curves, as well as corresponding tree objects, for geometric 90th Pareto quantile (top row) and topological 90th Pareto quantile (bottom row). (A) topological tree curves for geometric 90th Pareto quantile from CA1 (solid) and CA3 (dashed); (B) geometric curves for geometric 90th Pareto quantile from CA1 (solid) and CA3 (dashed); (C) tree object corresponding to CA1 geometric 90th Pareto quantile; (D) tree object corresponding to CA3 geometric 90th Pareto quantile; (E) topological tree curves for topological 90th Pareto quantile from CA1 (solid) and CA3 (dashed); (F) geometric curves for topological 90th Pareto quantile from CA1 (solid) and CA3 (dashed); (G) tree object corresponding to CA1 topological 90th Pareto quantile; (H) tree object corresponding to CA3 topological 90th Pareto quantile.

To summarize, the geometric and topological differences between dendritic trees from CA1 and CA3 regions can be observed by analyzing the 10th, 50th, and 90th Pareto quantiles of both regions. The basal dendrites from both regions are very similar geometrically and topologically, from very small and simple trees to larger and more complex trees. The basal trees from CA3 appear to be forests with more component trees than basal forests from CA1. The apical parts reveal bigger differences between two regions. Apical trees from CA1 region are slightly longer than the trees form CA3 region, and they exhibit a different branching pattern than apical trees

form CA3. The apical trees from CA1 have fewer branches at lower levels, and more branches at higher levels, and topologically, they form much taller trees.

3.5 Simulation Study

In this section we conduct simulation study to demonstrate the performance of our proposed method. To begin with, we focus on the probabilistic framework under which the arborizations are generated from stochastic processes. In Section 3.5.1, we first propose a stochastic generative model to simulate geometric properties of trees. In Section 3.5.2, based on the simulated geometry, we further develop a probabilistic framework to generate tree topology.

3.5.1 Neuron geometry

An intuitive choice of simulating geometric curve $g(x)$ from Section 3.2.3 is the age-dependent branching process, which represents the number of distinct tree branches at a distance x from the root; see Bellman and Harris [1952] for details. In general, age-dependent branching processes are not Markovian, and it can be further assumed that the distribution function of the length of every branch and the probability to produce offsprings are independent. Those authors further assumed homogeneity of the process, which is questionable for modeling neuron growth. In this paper, under our model settings, lengths of branches change as a function of x, the distance from the root.

Recall that each geometric curve is characterized by a set of discontinuity points (i.e., jump points), and the sign of each jump, either positive or negative. Correspondingly, our proposed simulation scheme includes two steps, generating a sequence of discontinuity points from a Cox process and, at each point, generating a sign of the jump from a generalized linear model. The Cox process is a doubly stochastic Poisson process with a random rate function, denoted here by $\Lambda(x)$. We outline the algorithm as follows.

1. Generate a sufficiently large number of realizations from $\Lambda(x)$, and calculate the maximum λ_{max}.

2. Set $x_0 = 0, x^* = 0$, and $N = 1$.

3. At step i, $i = 1, \ldots,$

   (a) Generate W as an exponential random variable with rate λ_{max} and set $x^* = x^* + W$.

   (b) Simulate a random variable U from the standard uniform distribution.

   (c) Accept point x^* if $U < \lambda_i/\lambda_{max}$, where λ_i is a realization from $\Lambda(x^*)$. If x^* is accepted, set $x_i = x^*$, otherwise go back to (a).

   (d) At point x_i, simulate a Bernoulli random variable Z with probability $p(x_i)$, where $p(x_i) = \Phi(\psi(x_i))$. Here, Φ is the normal distribution function, and ψ is a smooth function. If $Z = 1$, there is a positive jump at x_i, and set $g(x_i) = g(x_{i-1}) + 1$ and $N = N + 1$, otherwise there is a negative jump, and set $g(x_i) = g(x_{i-1}) - 1$ and $N = N - 1$.

   (e) Stop, if $N = 0$

Here, steps (3a)-(3c) essentially generate realization from a Cox process; see Burnecki et al. [2004] for detailed discussion on simulation of Cox processes. At each iteration, N represents the number of distinct branches, and the generation stops when N reaches zero. As a result, our algorithm yields observations from a randomly stopped Cox process; see Silvestrov [2006] for more information on stopped stochastic processes.

Our proposed simulation algorithm relies on two important components, including the distribution of the random rate function $\Lambda(x)$ and the probability function $p(x)$, or equivalently $\psi(x)$. To mimic the set of observed neuron trees, as described in Section 3.2.1, we take a heuristic approach and estimate both components from the real data. Let $Y(x)$ denote the total number

of jump points up to a distance x from the root. Conditioning on $\Lambda(x)$, $Y(x)$ has a Poisson distribution with rate $x\Lambda(x)$. Assuming that $\Lambda(x)$ follows a Gamma distribution, $\Gamma(\phi, \theta(x)/\phi)$, we have

$$Y(x) \sim \text{Negative-binomial}(\text{mean} = x\theta(x), \text{dispersion} = \phi), \tag{3.9}$$

where $\theta(x)$ is a positive smooth function and can be expressed as $\theta(x) = \exp(\alpha(x))$ for some smooth function $\alpha(x)$, which provides a flexible shape for the mean structure. In addition, $\alpha(x)$ can be approximated by a set of orthogonal polynomial basis functions with a vector of coefficients denoted by $\boldsymbol{\beta}$. Estimation of coefficients is quite challenging, which is partially due to the fact that $Y(x)$'s are dependent between different x's. For simplicity, we randomly sample one Y from each neuron tree, which results in a sample of independent observations. Moreover, parameter estimation of model (3.9) requires an alternating sequence of two steps, including an *Iteratively Reweighted Least Squares* (IRLS) procedure to estimate $\boldsymbol{\beta}$ and a *pseudo-likelihood maximization* procedure to estimate parameter ϕ [Carroll and Ruppert, 1988, Ruppert et al., 2003]. The left panel of Figure 3.18 depicts the estimated functions $\theta(x)$ for apical dendrites (solid line) and basal dendrites (dashed line). It can be seen that the point processes generating the apical and basal dendrites are inhomogeneous stochastic processes with the mean rate diminishing as the distance from the root increases.

Similarly, $\psi(x)$ can also be approximated by a set of basis functions. The parameter estimation can be conducted using the IRLS procedure. The estimated probability functions for apical dendrites and basal dendrites are depicted in the right panel of Figure 3.18. Note that both functions dimini sh with the distance from the root.

3.5.2 Neuron topology

In the previous section, we propose a probabilistic scheme to simulate geometric tree curves. Recall that trees with the same geometric curve or from the same geometric equivalence class may

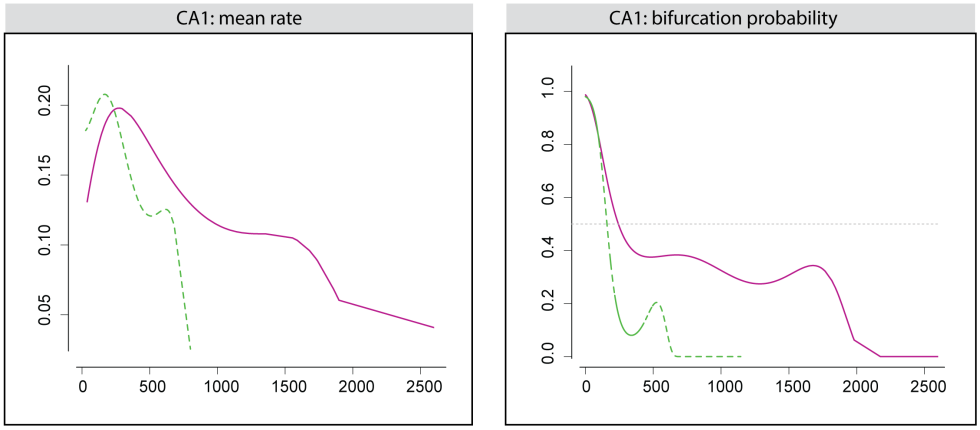

Figure 3.18: Left: Estimated mean rate functions for length of branches in two types of dendrites from CA1 using Negative-binomial model. Right: Estimated probability function of the branch bifurcation for two types of dendrites from CA1. (Solid line: apical dendrites; Dashed line: basal dendrites).

not have the same topology. Next, we focus on the simulation scheme of topological tree curve for a given geometric tree curve.

For a tree with m splits (i.e., internal nodes) and $m+1$ leaf nodes, its corresponding geometric curve has m positive, excluding the origin, and $m+1$ negative jumps. On the other hand, for a binary tree with m internal nodes, the total number of possible topologies was studied in the context of Conditioned Binary Galton-Watson trees [Pitman, 2006]. Without considering topological equivalence, there are $\mathbb{C}_m$ full binary trees with m internal nodes, where $\mathbb{C}_m = \binom{2m}{m}/(m+1)$ is the Catalan number. The number of different equivalence classes, denoted by $\mathbb{T}_m$, is much smaller. For comparison, Table 3.1 contains a list of first 10 Catalan numbers $\mathbb{C}_m$ and corresponding $\mathbb{T}_m$.

Table 3.1: Comparison of $\mathbb{C}_m$ and $\mathbb{T}_m$ for $m = 1, \ldots, 10$. Here, $\mathbb{C}_m$ represents the Catalan number, and $\mathbb{T}_m$ represents the number of topological equivalence classes.

m	1	2	3	4	5	6	7	8	9	10
$\mathbb{C}_m$	1	2	5	14	42	132	429	1430	4862	16796
$\mathbb{T}_m$	1	1	2	3	5	9	16	28	50	89

To simulate a topological tree curve for a given m, one possible approach is to assume that each of $\mathbb{C}_m$ topologies is equally likely. A method to generate such "uniformly distributed" tree topologies is available; see Mäkinen [1999] for a review of tree encodings and context-free grammars applied to simulation of full binary trees. Consequently, a topological tree curve can be produced.

Our proposed simulation scheme of the geometric and topological tree curves raises an important question regarding the existence of a tree objects; that is, is there a tree object given any pair of geometric and topological curves with the same number of internal nodes? Unfortunately, the answer is negative, and a counter-example is shown in Figure 3.19. Here, a geometric tree curve and a topological tree curve are displayed in panels (A) and (C) respectively. Notice that each curve corresponds to a tree with three internal nodes and four leaf nodes. The topological curve suggests that the tree is a complete binary tree with three levels (levels 0, 1 and 2), and all leaf nodes are on level two. An example tree is shown in panel (D). However, the geometric curve has a negative jump at 5.5, which suggests that there is a leaf at level one. Thus, both tree curves are not compatible. In addition, it is worth mentioning that the geometric curve in panel (A) corresponds to an equivalence class with unique topology. A depiction is shown in panel (B).

For a given geometric curve, certain topologies among those $\mathbb{C}_m$ possibilities are not feasible. On the other hand, there is at least one topology that is feasible. To demonstrate the existence of such solution, we give a short introduction to tree traversal or sequentialization. A tree traversal of a full binary tree refers to visiting every node of a tree in a systematic way. In the example in panel (C) of Figure 3.3, the tree can be traversed *depth-first*, where nodes are visited in the order $1, 2, 4, 8, 9, 5, 3$ or *breadth-first*, where nodes are visited in the order $1, 2, 3, 4, 5, 8, 9$. If the labelling only identifies internal nodes as a "1", and leaves as a "0", both traversals will produce binary strings, "1110000", and "1101000" respectively. This 0-1 encoding of a tree is also known as Zach's encoding and is used to generate binary trees at random from a context free grammar [Mäkinen, 1999]. Notably, any geometric curve can be mapped one-to-one to such a tree sequentialization.

We simply represent a positive jump as a "1", and a negative jump as a "0". The resulting binary string has exactly m occurrences of "1" and $m+1$ occurrences of "0". By interpreting this string as a trace of either a depth-first or a breadth-first tree traversal, we can generate at least one feasible topology.

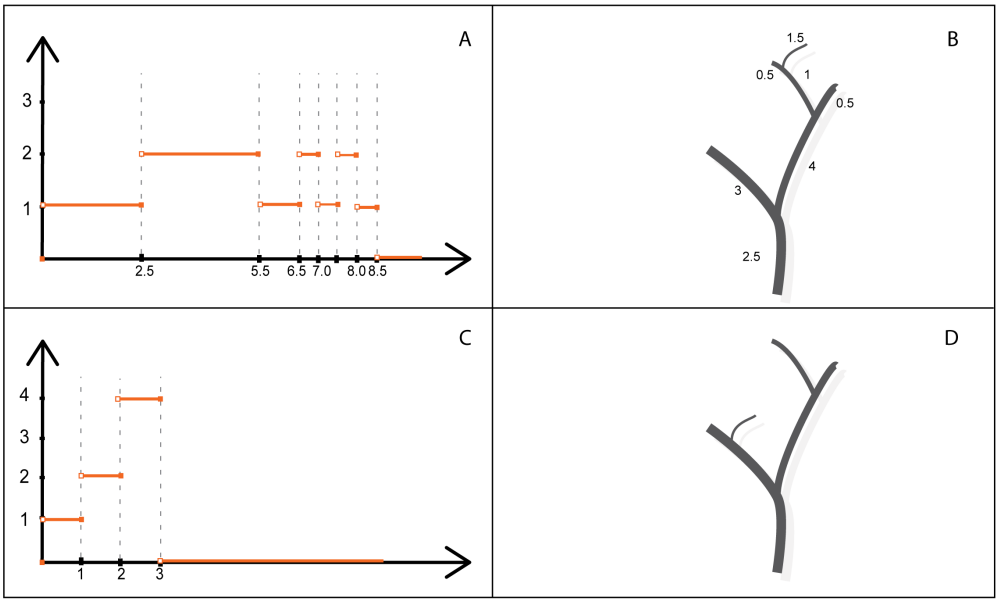

Figure 3.19: An example of incompatible geometric (A) and topological (C) curves with the same number of internal nodes $m = 3$. Tree in panel (B) has the geometric tree curve (A), and tree in panel (D) has the topological tree curve (C).

In practice, we consider the approach based on rejection sampling to generate tree topology. The rejection sampling of tree topologies is based on the uniform sampling of conditioned binary trees [Mäkinen, 1999]. Each of $\mathbb{C}_m$ trees is generated with equal probability, but infeasible topologies are rejected. Consequently, feasible tree topologies are generated with a uniform probability.

3.5.3 Tree Reconstruction

In Sections 3.5.1 and 3.5.2, we study the simulation for geometric and topological curves respectively. Here we focus on the reconstruction of a tree object corresponding to these two curves.

Let $g(x)$ be a geometric curve with discontinuity points $0 = x_0 < x_1 < \cdots < x_{2m+1}$. At each point, the jump size is denoted by $z_j = g(x_j) - g(x_{j-1})$ for $j = 1, \ldots, 2m+1$. Note that z_j is either 1 or -1. Let $\ell(x)$ be a feasible topological curve with m internal nodes. The algorithm below outlines the procedure to construct a tree-structured object.

1. Create the root node of the tree at level 0.
2. Create two child nodes of the root at level 1 if $\ell(2) = 2$. If $\ell(2) = 0$, stop.
3. At level i,
   (a) Randomly draw $\ell(i)/2$ nodes from nodes at level $i-1$.
   (b) Mark the selected nodes as internal nodes and the remaining nodes as leaves.
   (c) For each internal node at level $i-1$, create two child nodes at level i
4. Mark all nodes on the last level as leaves
5. Assign x_1 as the nodal attribute of the root, and mark the root as *visited*
6. Repeat, for $j = 2, \ldots, 2m+1$,
   (a) if $z_j > z_{j-1}$, randomly select an internal node with a visited parent among nodes closest to the leaves, and assign distance x_j to the selected node, mark it as visited.
   (b) if $z_j < z_{j-1}$, randomly select a leaf node with a visited parent, and assign distance x_j to the selected node, mark it as visited.

Our proposed algorithm consists of two stages, creation of a tree topology based on the topological curve (Steps 1-4) and assignment of lengths to edges based on the geometric curve (Steps 5-6). A combination of geometric and topological properties does not describe a unique tree, so nodes are selected at random in both stages. Examples of simulated dendritic trees are shown in Figure 3.20.

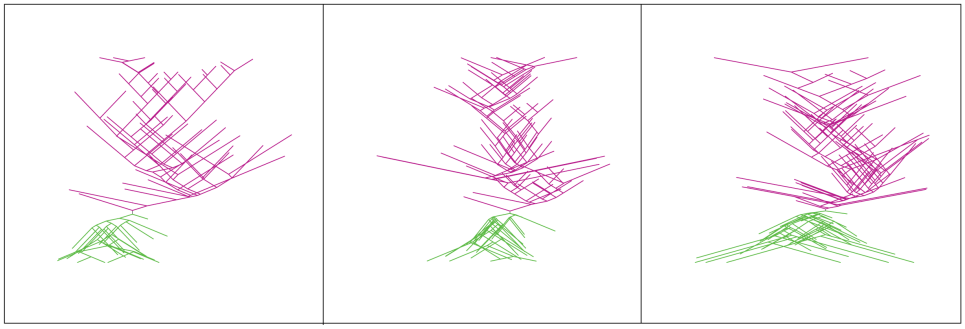

Figure 3.20: A sample of simulated neuron objects. The angles between branches are chosen for improved visibility.

3.5.4 Simulation Results

A simulation study with 100 repetitions is conducted. For each repetition, a sample of $n = 200$ tree objects is generated using the algorithms outlined in the previous sections. The sample topological and geometric Pareto quantiles are calculated based on our proposed methods.

In Figure 3.21, the pointwise 95% confidence intervals for both topological (left panel) and geometric (right panel) Pareto medians are depicted. In each panel, the population Pareto median is shown as thick black line, which is computed using Monte Carlo simulation based on 1000 trees. It can be seen that the population Pareto medians are within the 95% confidence intervals. Similar lessons are also learned for other Pareto quantiles; see Figures 3.22 and 3.23.

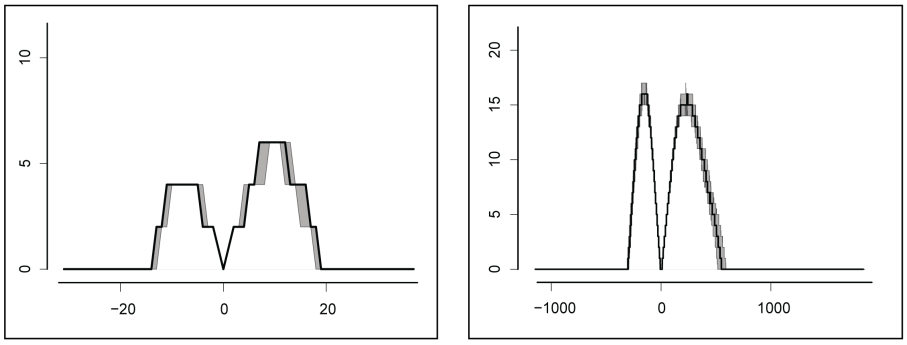

Figure 3.21: Topological (left) and geometric (right) pareto population median (black) and sample 95% confidence intervals (grey). The population medians are within the 95% confidence interval.

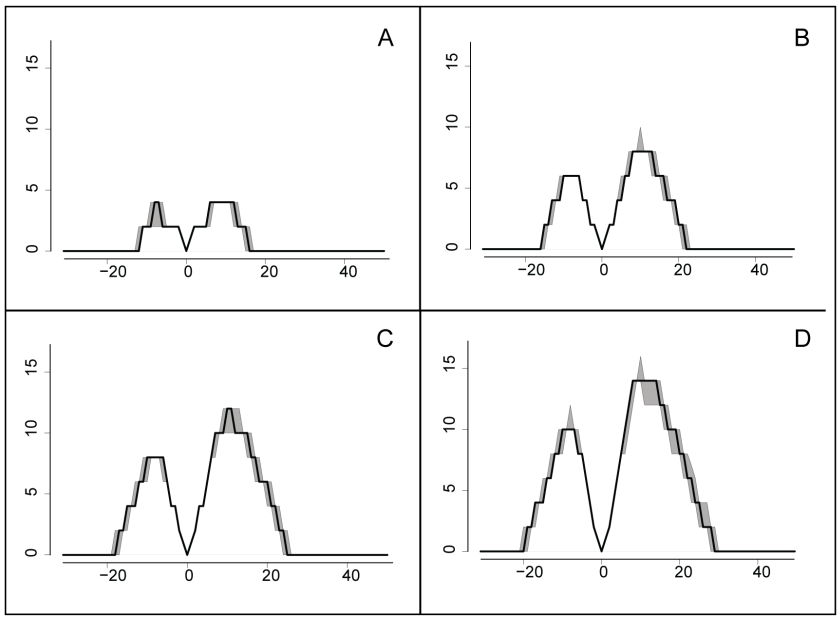

Figure 3.22: (A) 10th, (B) 25th, (C) 75th and (D) 90th topological Pareto population quantiles (black) and sample 95% confidence intervals (grey). The population quantile is within the 95% confidence interval.

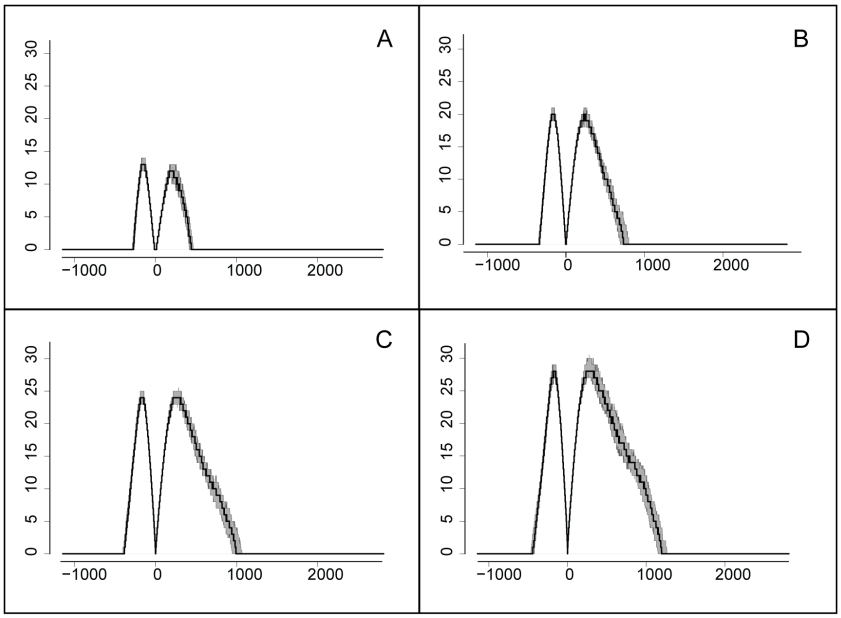

Figure 3.23: (A) 10th, (B) 25th, (C) 75th and (D) 90th geometric Pareto population quantiles (black) and sample 95% confidence intervals (grey). The population quantile is within the 95% confidence interval.

3.6 Proofs

3.6.1 Proof of Theorem 1

Lemma 1. Let s be a full binary tree with a topological curve $\ell_s(x)$. Let $A_i = \{2^i, \ldots, 2^i + \ell_s(i) - 1\}$. Then the union of all A_i's, denoted by $A = \cup_i A_i$, is the level-order index set of a tree in the topological equivalence class of s. This tree is called the *left-shifted modification* of s.

Proof of Lemma 1. First, we can see that $1 \in A_0 \subset A$. For any $i > 1$, consider an element $2k \in A_i$, i.e., $2^i \leq 2k \leq 2^i + \ell_s(i) - 1$. Thus, by the fact that $\ell_s(i) \leq 2\ell_s(i-1)$, we have

$$2^{i-1} \leq k \leq 2^{i-1} + \ell_s(i)/2 - 1/2 \leq 2^{i-1} + \ell_s(i)/2 - 1 \leq 2^{i-1} + \ell_s(i-1) - 1.$$

Similarly, consider an element $2k+1 \in A_i$, we have $2^{i-1} \leq k + 1/2 \leq 2^{i-1} + \ell_s(i)/2 - 1/2$, and

$$2^{i-1} \leq k \leq 2^{i-1} + \ell_s(i)/2 - 1 \leq 2^{i-1} + \ell_s(i-1) - 1.$$

That is, $k \in A_{i-1}$ and A corresponds to a binary tree, say s'.

Next, we will prove that s' is a full binary tree. Consider a node with the level-order index $2k$ at level i, such that $2^i \leq 2k \leq 2^i + \ell_s(i) - 1$. Note that, for a full binary tree s, $\ell_s(i)$ is an even number for $i > 1$, and thus, $2^i + \ell_s(i) - 1$ is an odd number. Consequently, $2k + 1 \leq 2^i + \ell_s(i) - 1$, i.e., $2k + 1 \in A_i$.

Finally, it can be seen that s and s' have the same number of nodes at each level. Thus, they have the same topological tree curve, and $s' \in [s]_{\mathtt{T}}$.

□

Proof of Theorem 1. For any two full binary trees s and t, let $\ell_s(x)$ and $\ell_t(x)$ be the corresponding topological tree curves. Let h_s and h_t denote the maximum levels of s and t, and $h = \max(h_s, h_t)$.

Recall that both $\ell_s(x)$ and $\ell_t(x)$ are piecewise constant functions. By (3.2), we have

$$d([s]_{\mathtt{T}}, [t]_{\mathtt{T}}) = ||\ell_s(x) - \ell_t(x)||_1 \equiv \int_0^\infty |\ell_s(x) - \ell_t(x)|dx = \sum_{i=0}^{h} |\ell_s(i) - \ell_t(i)|. \tag{3.10}$$

In the last summation, $|\ell_s(i) - \ell_t(i)|$ quantifies the difference in the number of nodes at level i.

On the other hand, the integer tree metric d_I [Wang and Marron, 2007] is defined as the symmetric difference between the level-order index sets of two trees. In addition, it can be written as a summation of level-wise differences [Wang, 2003]. That is, for any $s' \in [s]_{\mathtt{T}}$ and $t' \in [t]_{\mathtt{T}}$, $d_I(s', t') = \sum_{i=0}^{h} d_i(s', t')$, where $d_i(s', t')$ is a pseudo-metric and counts the number of different nodes with labels, at level i, in one tree but not the other. It can be seen that $d_i(s', t') \geq |\ell_{s'}(i) - \ell_{t'}(i)| = |\ell_s(i) - \ell_t(i)|$. Combining with (3.10), we have $d_I(s', t') \geq d([s]_{\mathtt{T}}, [t]_{\mathtt{T}})$

Next, we will prove that the equality is attainable for some trees. By Lemma 1, for trees s and t, there exists s' and t' which are the left-shifted modifications of s and t respectively. At level i, both s' and t' have nodes with consecutive indices starting from 2^i. Thus, we have $d_i(s', t') = |\ell_{s'}(i) - \ell_{t'}(i)| = |\ell_s(i) - \ell_t(i)|$, which completes the proof.

□

3.6.2 Proof of Theorem 2

Lemma 2. A function $\ell(x), x \in [0, \infty)$, is a topological tree representation of a full binary tree if the following conditions hold

(a) $\ell(0) = 0$ and $\ell(1) = 1$.

(b) $\ell(x)$ is a piecewise constant function, and there exist a positive integer M and a sequence of positive integers $c_1, \cdots, c_M$ such that $\ell(x) = c_k$ for $x \in (k-1, k]$ and $k = 1, \ldots, M$. In addition, $\ell(x) = 0$ for $x > M$.

(c) For any $k > 1$, c_k is an even number and $c_k \leq 2c_{k-1}$.

Lemma 2 is a direct consequence of the tree reconstruction method described in Section 3.5.3. In addition, condition (b) suggests that $\ell(x)$ has a finite support, and there exists a positive integer M such that $\ell(x) > 0$ for $x \in (0, M]$ and $\ell(x) = 0$ for $x \in (M, \infty)$. Next, we will prove Theorem 2.

Proof of Theorem 2. We will prove that $m_0(x)$ satisfies all conditions listed in Lemma 2. Condition (a) is straightforward. Note that $\ell_i(0) = 0$ and $\ell_i(1) = 1$ for $i = 1, \ldots, n$. Thus, we have $m_0(0) = 0$ and $m_0(1) = 1$.

Next, we will prove that (b) is satisfied. Note that $\ell_i(x) = 1$ for $x \in (0, 1]$. Therefore, we have $m_0(x) = 1$ for $x \in (0, 1]$. Let k be any integer and $k > 1$. For each interval $(k-1, k]$, $\ell_i(x)$ takes constant even numbers for all i. Thus, $m_0(x)$ is a piecewise constant function. In fact, when the median is unique, $m_0(x)$ is clearly an even number. On the other hand, when the median is not unique, we take the minimum to break the tie, which also yields an even number.

Let $k_0 > 0$ be a positive integer such that $m_0(k_0) > 0$ and $m_0(k_0 + 1)$=0. Clearly, $m_0(x)$ is zero for all $x \in (k_0, k_0 + 1]$. By the property of median, it can be seen that there are at least $\lceil n/2 \rceil$ tree curves $\ell_i(x)$ such that $\ell_i(x) = 0$ for all $x > k_0$. This implies that $m_0(x) = 0$ for all $x > k_0$, and condition (b) is satisfied.

Finally, we will prove (c) by showing that $m_0(k) \leq 2m_0(k-1)$ for any integer $k > 1$. Note that $m_0(k-1)$ is the pointwise median at $k-1$. Thus, there are at least $\lceil n/2 \rceil$ elements less than or equal to $m_0(k-1)$ at $k-1$. Without loss of generality, we denote these elements as $\ell_{\pi_j}(x)$, $j = 1, \ldots, \lceil n/2 \rceil$, and $\ell_{\pi_j}(k-1) \leq m_0(k-1)$. Each element $\ell_{\pi_j}(x)$ is a valid topological curve satisfying condition (c), therefore $\ell_{\pi_j}(k) \leq 2\ell_{\pi_j}(k-1)$. Thus there are at least $\lceil n/2 \rceil$ elements $\ell_{\pi_j}(x)$ such that, at k, $\ell_{\pi_j}(k) \leq 2m_0(k-1)$, which implies that $m_0(k) \leq 2m_0(k-1)$. This completes the proof. □

3.6.3 Proof of Theorem 3

Lemma 3. A function $g(x), x \in [0, \infty)$, is a geometric tree representation of a full binary tree if the following conditions hold

(a) $g(0) = 0$.

(b) g(x) is a piecewise constant function, and there exist a positive integer m, a sequence of real numbers $0 = x_0 < x_1 < \cdots < x_{2m+1}$ and a sequence of positive integers $c_1, \ldots, c_{2m+1}$ such that $g(x) = c_k$ for $x \in (x_{k-1}, x_k]$. In addition, $g(x) = 0$ for $x > x_{2m+1}$.

(c) $c_1 = 1$ and, for any $k = 2, \ldots, 2m + 1$, $|c_k - c_{k-1}| = 1$.

Similar to Lemma 2, Lemma 3 is a direct consequence of the tree reconstruction method described in Section 3.5.3. In addition, condition (b) suggests that $g(x)$ has a finite support. Next, we will prove Theorem 3.

Proof of Theorem 3. We will prove that $m_1(x)$ satisfies all conditions listed in Lemma 3. First, note that $g_i(0) = 0$ for $i = 1, \ldots, n$. Thus, we have $m_1(0) = 0$.

For each i, $g_i(x)$ has a finite, odd number of jump points on $(0, \infty)$, denoted by $A_i = \{x_1^{(i)}, x_2^{(i)}, \ldots\}$, and is a constant function on each interval $(x_{k-1}^{(i)}, x_k^{(i)}]$, for $k = 1, 2, \ldots, |A_i|$ where $x_0^{(i)} = 0$. Let A denote the collection of all distinct jump points of all geometric curves and $A = \cup_i A_i$. In addition, denote elements in A as $x'_1 < \cdots < x'_K$ where $K = |A|$ is the total number of distinct jump points. Clearly, $m_1(x)$ is a constant on $(x'_{k-1}, x'_k]$ for $k = 1, \ldots, K$, and also takes integer values. Next, all geometric tree curves have finite support. Consequently, the pointwise median $m_1(x)$ also has finite support. Let x'_k and x'_{k+1} be two adjacent jump points with $m_1(x'_k) > 0$ and $m_1(x'_{k+1}) = 0$. It can be seen that $m_1(x)$ is zero for all $x \in (x'_k, x'_{k+1}]$. By the property of median, it can be shown that there are at least $\lceil n/2 \rceil$ tree curves $g_i(x)$ such that

$g_i(x) = 0$ for all $x > x'_k$. This implies that $m_1(x) = 0$ for all $x > x'_k$. Before proving that the number of jump points is an odd number, we will first prove condition (c).

We will prove (c) by first showing that $m_1(x'_k) - m_1(x'_{k-1}) \in \{-1, 0, 1\}$ for any two adjacent points x'_{k-1}, x'_k, where $k = 2, \ldots, K$. Note that $m_1(x'_{k-1})$ is the pointwise median at x'_{k-1}, thus there exist $\lceil n/2 \rceil$ elements less than or equal to $m_1(x'_{k-1})$ at x'_{k-1}, and $n - \lceil n/2 \rceil + 1$ elements greater or equal than $m_1(x'_{k-1})$. Without loss of generality, we denote those elements as $g_{\pi_j}(x)$ and $g_{\pi_j}(x'_{k-1}) \leq m_1(x'_{k-1})$ for $j = 1, \ldots, \lceil n/2 \rceil$ and $g_{\pi_j}(x'_{k-1}) \geq m_1(x'_{k-1})$ for $j = \lceil n/2 \rceil, \ldots, n$. Each element $g_{\pi_j}(x)$ is a valid geometric curve satisfying condition (c), so $|g_{\pi_j}(x'_k) - g_{\pi_j}(x'_{k-1})| = 1$ if $x'_k \in A_{\pi_j}$ and 0 otherwise. Thus, at x'_k, there are at least $\lceil n/2 \rceil$ elements $g_{\pi_j}(x)$ such that $g_{\pi_j}(x'_k) \leq m_1(x'_{k-1})+1$, and at least $n-\lceil n/2 \rceil+1$ elements $g_{\pi_j}(x)$ such that $g_{\pi_j}(x'_k) \geq m_1(x'_{k-1})-1$, which implies that $|m_1(x'_k) - m_1(x'_{k-1})| \leq 1$.

Finally, we select points x'_k such that $m_1(x'_k) \neq m_1(x'_{k-1})$ and denote them as $x_1, \ldots, x_M$. Note that $|m_1(x_k) - m_1(x_{k-1})| = 1$, and $\{x_k\}_{k=1}^M$ are jump points of $m_1(x)$. It remains to be shown that $m_1(x_1) = 1$ and M is an odd number such that $M = 2m + 1$ for some m.

Recall that $x_1^{(i)}$ is the first jump point for $g_i(x)$, and $g_i(x_1^{(i)}) = 1$, for $i = 1, \ldots, n$. Let y_1 be such a point that $y_1 = \max\{x_1^{(i)} : \text{at least } \lceil n/2 \rceil \text{ of } g(x_1^{(i)}) = 1\}$. Clearly, the pointwise median at y_1 is 1. In addition, y_1 is a jump point, and $m_1(x) = 1$ for $x \in (0, y_1]$. Thus, we have $x_1 = y_1$. Similarly, $m_1(x_M) = 1$. By noting that the jump sizes are either 1 or -1, we can see that M is an odd number, which completes the proof. □

3.6.4 Proof of Theorem 4

Proof of Theorem 4. Let $q_0(x)$ be the pointwise 100τ-th quantile for the set of topological curves $\ell_1(x), \ldots, \ell_n(x)$. We will prove that $q_0(x)$ satisfies all conditions listed in Lemma 2. Condition (a) is straightforward. In fact, note that $\ell_i(0) = 0$ and $\ell_i(1) = 1$ for $i = 1, \ldots, n$. Thus, we have $q_0(0) = 0$ and $q_0(1) = 1$.

Next, we will prove that (b) is satisfied. Note that $\ell_i(x) = 1$ for $x \in (0, 1]$. Therefore, we have $q_0(x) = 1$ for $x \in (0, 1]$. Let k be any integer, and $k > 1$. For each interval $(k-1, k]$, $\ell_i(x)$ takes constant even numbers for all i. Thus, $q_0(x)$ is a piecewise constant function. In fact, when the quantile is unique, $q_0(x)$ is clearly an even number. On the other hand, when the quantile is not unique, we take the minimum to break the tie, which also yields an even number.

Let $k_0 > 0$ be a positive integer such that $q_0(k_0) > 0$ and $q_0(k_0 + 1)$=0. Clearly, $q_0(x)$ is zero for all $x \in (k_0, k_0 + 1]$. By the property of a quantile, it can be seen that there are at least $\lceil n\tau \rceil$ tree curves $\ell_i(x)$ such that $\ell_i(x) = 0$ for all $x > k_0$. This implies that $q_0(x) = 0$ for all $x > k_0$, and condition (b) is satisfied.

Finally, we will prove (c) by showing that $q_0(k) \leq 2q_0(k-1)$ for integer $k > 1$. Note that $q_0(k-1)$ is the pointwise quantile at $k-1$. Thus, there are $\lceil n\tau \rceil$ elements less than or equal to $q_0(k-1)$ at $k-1$. Without loss of generality, we denote these elements as $\ell_{\pi_j}(x)$ and $\ell_{\pi_j}(k-1) \leq q_0(k-1)$ for $j = 1, \ldots, \lceil n\tau \rceil$. Each element $\ell_{\pi_j}(x)$ is a valid topological curve satisfying the condition (c), therefore $\ell_{\pi_j}(k) \leq 2\ell_{\pi_j}(k-1)$. Thus there are at least $\lceil n\tau \rceil$ elements $\ell_{\pi_j}(x)$ such that at k, $\ell_{\pi_j}(k) \leq 2q_0(k-1)$, which implies that $q_0(k) \leq 2q_0(k-1)$. This completes the proof. □

3.6.5 Proof of Theorem 5

Proof of Theorem 5. Let $q_1(x)$ be the pointwise 100τ-th quantile for the set of geometric curves $g_1(x), \ldots, g_n(x)$ We will prove that $q_1(x)$ satisfies all conditions listed in Lemma 3. First, note that $g_i(0) = 0$ for $i = 1, \ldots, n$, thus we have $q_1(0) = 0$.

For each i, $g_i(x)$ has a finite, odd number of jump points, denoted by $A_i = \{x_1^{(i)}, x_2^{(i)}, \ldots\}$, and is a constant function on each interval $(x_{k-1}^{(i)}, x_k^{(i)}]$, for $k = 1, 2, \ldots, |A_i|$ where $x_0^{(i)} = x_0$. Let A denote the collection of all distinct jump points of all geometric curves and $A = \cup_i A_i$. In addition, denote elements in A as $x'_1 < \cdots < x'_K$ where $K = |A|$. Clearly, $q_1(x)$ is a constant function on $(x'_{k-1}, x'_k]$ for $k = 1, \ldots, K$, and also takes integer values. Next, all geometric tree curves have

finite support. Consequently, the 100τ-th quantile $q_1(x)$ also has finite support. Let x'_k and x'_{k+1} be two adjacent jump points with $q_1(x'_k) > 0$ and $q_1(x'_{k+1}) = 0$. It can be seen that $q_1(x)$ is zero for all $x \in (x'_k, x'_{k+1}]$. By the property of the quantile, it can be shown that there are at least $\lceil n\tau \rceil$ tree curves $g_i(x)$ such that $g_i(x) = 0$ for all $x > x'_k$. This implies that $q_1(x) = 0$ for all $x > x'_k$. Before proving that the number of jump points is an odd number, we will first prove condition (c).

We will prove (c) by first showing that $q_1(x'_k) - q_1(x'_{k-1}) \in \{-1, 0, 1\}$ for any two points x'_{k-1}, x'_k, where $k = 2, \ldots, K$. Note that $q_1(x'_{k-1})$ is the τ-th pointwise quantile at x'_{k-1}, thus there are $\lceil n\tau \rceil$ elements less than or equal to $q_1(x'_{k-1})$ at x'_{k-1}, and $n - \lceil n\tau \rceil + 1$ elements greater or equal than $q_1(x'_{k-1})$. Without loss of generality, we denote those elements as $g_{\pi_j}(x)$ and $g_{\pi_j}(x'_{k-1}) \leq q_1(x'_{k-1})$ for $j = 1, \ldots, \lceil n\tau \rceil$ and $g_{\pi_j}(x'_{k-1}) \geq q_1(x'_{k-1})$ for $j = \lceil n\tau \rceil, \ldots, n$. Each element $g_{\pi_j}(x)$ is a valid geometric curve satisfying condition (c), so $|g_{\pi_j}(x'_k) - g_{\pi_j}(x'_{k-1})| = 1$ if $x'_k \in A_{\pi_j}$ and 0 otherwise. Thus, there are at least $\lceil n\tau \rceil$ elements $g_{\pi_j}(x)$ such that $g_{\pi_j}(x'_k) \leq q_1(x'_{k-1}) + 1$, and at least $n - \lceil n\tau \rceil + 1$ elements $g_{\pi_j}(x)$ such that $g_{\pi_j}(x'_k) \geq q_1(x'_{k-1}) - 1$, which implies that $|q_1(x'_k) - q_1(x'_{k-1})| \leq 1$.

Finally, we select points x'_k such that $q_1(x'_k) \neq q_1(x'_{k-1})$ and denote them as $x_1, \ldots, x_M$. Note that $|q_1(x_k) - q_1(x_{k-1})| = 1$ and $\{x_k\}_{k=1}^M$ are jump points of $q_1(x)$. It remains to be shown that $q_1(x_1) = 1$ and M is an odd number such that $M = 2m + 1$ for some m.

Recall that $x_1^{(i)}$ is the first jump point for $g_i(x)$, and $g_i(x_1^{(i)}) = 1$, for $i = 1, \ldots, n$. Let y_1 be such a point that $y_1 = \max\{x_1^{(i)} : \text{at least } \lceil n\tau \rceil \text{ of } g(x_1^{(i)}) = 1\}$. Clearly, the τ-th quantile at y_1 is 1. In addition, y_1 is a jump point, and $m_1(x) = 1$ for $x \in (0, y_1]$. Thus we have $x_1 = y_1$. Similarly, $q_1(x_M) = 1$. By noting that the jump sizes are either 1 or -1, we can see that M is an odd number, which completes the proof. $\square$

CHAPTER 4

Genetic Algorithm for Tree-Structured Data

4.1 Introduction

The idea of genetic algorithm is built on the theoretical work of Holland [1973], who used the principles of natural selection to solve combinatorial optimization problems. The algorithm mimics the genetic setup in an iterative fashion, with chromosomes, represented as strings of symbols, and genes, which are parts of chromosomes. Starting from a random set of chromosomes, and rating them according to the stated optimization goal, the algorithm follows the rules of natural selection to find the best solution, or its close approximation. At each iteration, the algorithm operates on a set of chromosomes, also referred to as individuals in the current generation. The algorithm selects chromosomes from this set to create a new generation of chromosomes. The individuals with the highest rating, a.k.a. the fittest, have the largest probability to be selected to participate in the reproductive operations. The rules of selection and its adaptation to tree optimization are described in Section 4.4. There are two reproductive operations, a cross-over and a mutation. Cross-over is a technique that creates a new chromosome from two parent chromosomes by taking pieces of information from each parent. A mutation is a random change in a gene. Reproductive operations for tree objects are described in Section 4.5. A new generation of chromosomes can replace the old one entirely, or the best elements from the old one are retained. This is known as a replacement strategy. The process is repeated a preconfigured number of times, or till some specific optimization goals are reaches.

The symbols used in the chromosomes are application specific. In some examples, like model selection with a large number of predictors, binary encoding, in which chromosomes are represented as strings of 0's and 1's, works well; see Givens and Hoeting [2012]. More complex encodings

are often necessary, e.g., quadruple (as in the original DNA encoding), octal or hexadecimal, and the corresponding reproductive operations have to be redefined accordingly. The encoding for tree objects is explained in Section 4.3.

In general, each application of the algorithm needs to address the following: encoding (Section 4.3), selection (Section 4.4), definition of cross-over and mutation (Section 4.5), and replacement (Section 4.6). See Sivanandam and Deepa [2007] for additional details.

4.2 Genetic Algorithm for Tree Objects

Our goal is to solve a constrained multi-objective optimization problem (3.6) in the space of trees. The genetic algorithm is applied to geometric curves which can be naturally encoded as chromosomes; see Section 4.3 for details.

Every chromosome is scored according to the objective functions, G_n^τ and T_n^τ representing the geometric and topological scores respectively. For each objective function, the minimizing solution is known and can be found in linear time as a direct consequence of Theorems 4 and 5. In general, these two solutions do not correspond to a single tree. In fact, they often correspond to trees with different number of nodes. The goal of the algorithm is, in simplified terms, starting from the best geometry (a G_n^τ-minimizer), to apply random changes, so the resulting geometry is compatible with the best topology (a T_n^τ-minimizer). By operating on a set of chromosomes in each generation, the genetic algorithm executes multiple searches concurrently. By iteratively executing the selection process, the algorithm finds all (or many of) Pareto solutions. The constraint, namely each solution has to represent a valid tree, is enforced through the design of the reproductive operations, rather than through penalty scoring.

4.3 Encoding

Recall that, for a geometric tree curve, say $g_i(x)$, $A_i = \{x_1^{(i)}, x_2^{(i),\cdots}\}$ denotes its set of jump points over $(0, \infty)$. Let A be the union of all jump points in the sample, i.e., $A = \cup_i A_i$. Let

$x_1 < x_2 < \cdots < x_K$ be the ordered elements of A. Thus, for a curve $g_i(x)$, a *gene* in the position k represents the value of $g_i(x)$ for $x \in (x_k, x_{k+1}]$. For instance, in Figure 4.1, simple trees depicted in Figure 3.7 can be encoded as strings of digits. Note that there are 11 jump points for this sample.

1	1	1	2	2	2	2	2	2	2	1	
1	1	2	2	1	1	1	1	0	0	0	
1	2	2	2	2	1	2	1	1	0	0	

Figure 4.1: Chromosome-like encoding for trees in the toy example from Figure 3.7.

In our real application, with such simple encoding, a neural tree can be represented as a chromosome consisting of over 10,000 genes, and cannot be efficiently processed by the genetic algorithm. Thus, a compressed encoding is preferable. Here, a simple compression mechanism is employed, in which the consecutive identical symbols are compressed to a pair consisting of the symbol and its count. An example is given in Figure 4.2. The straightforward correspondence between a gene position and a jump point is lost, but the computations gain in efficiency.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
40	106	51	79	121	24	85	90	118	9	82	2	21	29	3	132	243	41	11	69	263	24	25	28	28	55	18	26	47	72	170	25	131	131	35	110	820	410	427	490

Figure 4.2: An example of a compressed chromosome for an apical tree. There are 40 genes in this chromosome. The longest chromosome in the sample has 150 genes.

4.4 Selection

The purpose of selection has been explained in Section 4.1. The selection procedure consists of two steps: scoring and fitness evaluation, repeated in each iteration of the genetic algorithm.

The score evaluates the objective functions for a given geometry encoded as a chromosome. In the case of the multi-objective optimization, the score is multi-dimensional, and it consists

of a geometric score G_n^τ and a topological score T_n^τ, as defined in (3.8). For a given geometry, finding the best topology and its corresponding topological score are relatively expensive. To mitigate the computational burden, we use a topological proxy score instead, which is defined as the difference in the number of nodes between the tree with a given geometry and the tree with the best topology (which is known as a consequence of Theorem 4). A a result, we obtain a sequence of geometric curves, i.e., chromosomes. Each of those geometric curves will lead to a Pareto solution. For each chromosome, the best topological curve can be found as described in Section 4.7. Details of proxy scoring are explained in Sivanandam and Deepa [2007].

The fitness evaluation is the other important step in selection, and it is not synonymous with score. At each iteration, the separate fitness evaluation allows the application of a different "selection pressure" to choose the candidate chromosomes for reproduction from the chromosomes available in the current generation. The candidate chromosomes at each iteration can get a reproductive probability based on the score or the rank. In general, the chromosomes with better scores or ranks will be assigned with higher probabilities. The relative magnitude of probabilities between chromosomes defines high or low selection pressure. It is common to use the number of dominating chromosomes in the current generation as a rank, i.e., all elements in the current Pareto set get a rank 0. The probability to be selected for reproduction is inversely proportional to the rank. Here, we chose the fitness evaluation based on ranking.

4.5 Reproduction and Feasibility

The literature on constrained optimization discusses two common approaches to feasibility. One is to include non-feasible solutions, with some penalty built into the scoring function, and the other is to maintain feasibility through reproductive operations. In our case the non-feasible solutions have to be excluded, or they would quickly dominate the population. To accomplish

this, mutation and cross-over operations, introduced in Section 4.1, have to be designed carefully to produce only valid trees (chromosomes).

A mutation is defined as a change of one symbol in tree encoding by 1 or -1, with some small probability e.g., 0.01, provided that the change does not invalidate the tree. The goal of a mutation is to make a substantial change in the tree, i.e., modifying a branch, adding a branch or deleting a branch. The mutations are depicted and explained in Figure 4.3. Panel (A) shows a mutation that moves a single node, thus changing the length of a branch. This mutation affects only the geometric score G_n^τ. Panel (B) depicts a mutation that creates a new branch by adding one internal node and one leaf. Panel (C) shows a mutation that delets of a branch by removing one internal node and one leaf. Mutations (B) and (C) affect both a topological score and a geometric score.

Cross-overs are applied to two selected individuals (parents). A single position is randomly drawn, which divides each parent into two parts. Next, two individuals (children) are created by merging the opposite pieces of the parents. If merging would create an invalid tree, the closest valid position is found. The result of a cross-over is harder to predict than that of a mutation, but it will likely create (child) chromosomes with different number of nodes than in the parents. Cross-overs can lead to larger changes than incremental single-gene modifications induced by mutations.

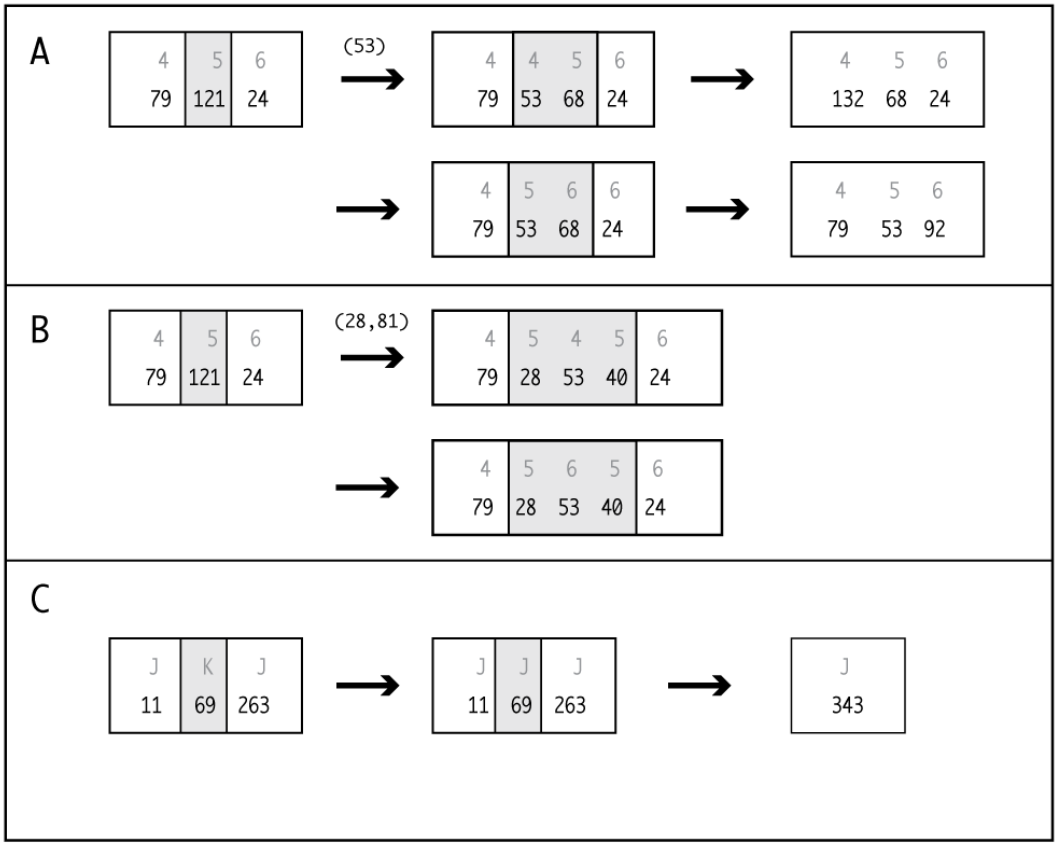

Figure 4.3: Three types of mutations. The center element in the left-most box (marked in grey) is selected for a mutation. The numbers above the arrow are randomly selected, in the range determined by the symbol count. Panel (A): a mutation modifies a branch by moving a node; panel (B): a mutation adds a branch; panel (C): a mutation deletes a branch. Mutations in panels (A) and (B) have two subtypes and each is selected with an equal probability. The probabilities of applying (A), (B) or (C) can be equal or configurable.

4.6 Replacement

The replacement strategies define how long individuals can live, through one generation or forever. There are two commonly used strategies, generational update and steady state. The generational update replaces either the entire generation, referred to as (λ, μ) strategy, or allows elites to survive, referred to as $(\lambda + \mu)$ strategy. The steady state strategy inserts new individuals into one "steady" generation, by replacing, most commonly, the weakest individuals. In our settings, the generational update with elites provided the fastest convergence.

4.7 The best topology for a geometry

It is not difficult to generate a random topology for a given geometry. To generate the best topology, we choose to generate a large number of random topologies, and select the best one, i.e., the one minimizing the topological distance T_n^τ as defined in (3.8).

To generate a random topology, we use the Zach's encoding for the geometric curve, encoding a jump up as a 1, and a jump down as a 0. Recall that, from this encoding, we can obtain information such as internal nodes or leaf nodes. We process the symbols from the encoding of the geometric curve from left to right by placing them, with equal probability, in the nodes of a binary tree. A topological binary tree can be stored in an array using level-order indexing as indices in the array. The root is stored at the position 1, and children of any node with index k are stored in the positions $2k$ and $2k + 1$. Some consideration is needed to generate full binary trees.

CHAPTER 5

Future Work

Complex data objects, specifically tree-structured data objects, are frequently encountered in nature. Examples include plant roots, blood vessels and neural cells. With advances in medical and biological imaging technologies, such data are becoming ubiquitous, and increasingly are elements of statistical analysis. What is often of interest to scientists is the description and the quantification of differences between samples of complex objects. The differences may lie in factors such as species, age, gender, treatment or disease. In many practical problems, the purpose of the study is to find abnormalities in the sample, or estimate how often abnormalities are encountered. The tree-structured data are extremely non-Euclidean, and many classical statistical methods cannot be used for such purpose. Some statistical methods, e.g., kernel based methods, have been adapted to operate on non-Euclidean objects based on a pre-selected distance metric. Thus, a suitable choice of distance between tree-structured objects is essential to statistical inference in the domain of object-oriented data. For instance, equipped with the Hamming distance (i.e., integer tree metric), a nonparametric smoothing technique and principal component analysis of trees have been developed; see Wang and Marron [2007], Aydın et al. [2009], Chang et al. [2011] and Wang et al. [2012]. Similarly, a weighted Hamming distance has been applied to hypothesis testing in the space of protein classification trees [Balding et al., 2009, Busch et al., 2009].

The distance metric introduced and applied in these papers is based on topological properties of trees and a node labeling scheme. The node labeling in the context of trees provides a systematic approach to uniquely identify a node in a tree. Such designation can be straightforward for phylogenetic and classification trees, but in many cases relies on the notion of correspondence between branches [Aydın et al., 2009]. A different choice of correspondence, e.g., thickness of branches or number of descendants, could potentially lead to different research conclusions. In

our future work, motivated by the study of dendritic structures of neurons from two regions of the hippocampus, CA1 and CA3, we will extend the framework developed in Chapter 3, and devise analytical tools, invariant to node assignment, to test statistical hypotheses related to complex objects.

Hypothesis testing in the space of trees requires a probability measure on that space. For instance, Banks and Constantine [1998] introduced a probability measure on the space of finite graphs, with two parameters analogous to the centrality and scale of the exponential family distribution. A similar idea was followed by Wang et al. [2012], who introduced a probability measure on the space of trees and successfully simulated trees to demonstrate their proposed method.

In literature, two other tree distributions have been widely used, including a uniform tree distribution [Arnold and Sleep, 1980, Flajolet and Odlyzko, 1982], and a distribution of binary search trees [Devroye, 1986, 1987]. Both distributions describe a population of binary trees with a fixed number of nodes. For trees with m internal nodes, the uniform distribution assigns each full binary tree with an equal probability. It is known that the number of such trees is described by the mth Catalan number $\mathbb{C}_m$ [Harris, 1952]. Asymptotic properties associated with heights of uniformly distributed trees have been studied by Flajolet et al. [1993], Flajolet and Odlyzko [1982]. A distribution of binary search trees is often considered in the context of sorting and searching applications in computer science. The asymptotic results [Devroye, 1986, 1987] can be used to estimate an average stack size for algorithmic executions, or to detect anomalies in the data. The binary search trees are not full binary trees, but we consider a one-to-one mapping to the space of full binary trees. We refer to this new distribution as a leaf-uniform based on its construction method.

In our future work, we plan to consider both one-sample and two-sample hypothesis testing problems in the space of full binary trees. For one-sample testing, we examine two reference

distributions, including a uniform and a leaf-uniform distribution. As indicated earlier, these distributions are characterized by a fixed number of nodes, and yet most biological tree-structured data come in many different sizes. We plan to investigate a novel mixture distribution, whose components are from the uniform or leaf-uniform distributions. This new distribution provides great opportunities for further methodological development, e.g., parameter estimation. In addition, we will derive a test statistic and obtain its distribution based on simulations. An example of a relevant test statistic is the tree height, which is a univariate statistic. There are many tests available to compare the mean, the variance and the distributions of heights, against those coming from a reference distribution, e.g., t-test and Kolmogorov-Smirnov test, Multiple-Response Permutation Procedure (MRPP). In some cases, for large trees, certain asymptotic results from probabilistic literature may be applicable. The simulation based one-sample testing will rely on the availability and effectiveness of sampling methods for trees. The uniform and leaf-uniform distributions can be efficiently sampled, as a result of work by Arnold and Sleep [1980], Mairson [1994], Mäkinen [1999] and many others. Similarly, for two-sample hypothesis testing, we will also consider test statistics based on tree heights, using tests mentioned in the context of one-sample testing.

BIBLIOGRAPHY

Akaike, H. (1973). Information theory and an extension of the maximum likelihood principle. In *Proceeding of the 2nd International Symposium on Information Theory*, pages 267–281.

Antoniou, A. and Lu, W. (2007). *Practical Optimization.* Springer.

Archer, E. W., Park, I. M., and Pillow, J. W. (2013). Bayesian entropy estimation for binary spike train data using parametric prior knowledge. In *Advances in Neural Information Processing Systems*, pages 1700–1708.

Arnold, D. B. and Sleep, M. R. (1980). Uniform random generation of balanced parenthesis strings. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 2(1):122–128.

ASA (2014). Statistical research and training under the brain initiative.

Ascoli, G., Donohue, D., and Halavi, M. (2007). NeuroMorpho.Org: a central resource for neuronal morphologies. *Journal of Neuroscience*, 27:9247–9251.

Ascoli, G. A. and Krichmar, J. L. (2000). L-neuron: a modeling tool for the efficient generation and parsimonious description of dendritic morphology. *Neurocomputing*, 32:1003–1011.

Aydın, B., Pataki, G., Wang, H., Bullitt, E., and Marron, J. (2009). A principal component analysis for trees. *The Annals of Applied Statistics*, 3(4):1597–1615.

Balding, D., Ferrari, P. A., Fraiman, R., and Sued, M. (2009). Limit theorems for sequences of random trees. *Test*, 18(2):302–315.

Banks, D. and Constantine, G. (1998). Metric models for random graphs. *Journal of Classification*, 15(2):199–223.

Bellman, R. and Harris, T. (1952). On age-dependent binary branching processes. *Annals of Mathematics*, pages 280–295.

Berger, T., Song, D., Marmarelis, V., LaCoss, J., Wills, J., Gerhardt, G., Granacki, J., Hampson, R., and Deadwyler, S. (2005). *Neural Engineering*, chapter Reverse Engineering the Brain: A Hippocampal Cognitive Prosthesis for Repair and Enhancement of Memory Function. In He [2005].

Billera, L. J., Holmes, S. P., and Vogtmann, K. (2001). Geometry of the space of phylogenetic trees. *Advances in Applied Mathematics*, 27(4):733–767.

Brown, E., Barbieri, R., Ventura, V., Kass, R., and Frank, L. (2002). The time-rescaling theorem and its application to neural spike train data analysis. *Neural computation*, 14(2):325–346.

Burnecki, K., Härdle, W., and Weron, R. (2004). Simulation of risk processes. *Encyclopedia of actuarial science*.

Busch, J. R., Ferrari, P. A., Flesia, A. G., Fraiman, R., Grynberg, S. P., and Leonardi, F. (2009). Testing statistical hypothesis on random trees and applications to the protein classification problem. *The Annals of Applied Statistics*, pages 542–563.

Cannady, J. (1998). Artificial neural networks for misuse detection. In *National information systems security conference*, pages 368–81.

Carroll, R. J. and Ruppert, D. (1988). *Transformation and weighting in regression*, volume 30. CRC Press.

Chang, H., Iyer, H., Bullitt, E., and Wang, H. (2011). Statistical modeling of branching probabilities for tree-structured data objects.

Chen, J. and Chen, Z. (2008). Extended bayesian information criteria for model selection with large model spaces. *Biometrika*, 95:759–771.

Chu, C., Kim, S. K., Lin, Y.-A., Yu, Y., Bradski, G., Ng, A. Y., and Olukotun, K. (2007). Map-reduce for machine learning on multicore. *Advances in neural information processing systems*, 19:281.

Cline, H. and Haas, K. (2008). The regulation of dendritic arbor development and plasticity by glutamatergic synaptic input: a review of the synaptotrophic hypothesis. *The Journal of physiology*, 586(6):1509–1517.

Coello, C. A. C., Lamont, G. B., and Van Veldhuisen, D. A. (2007). *Evolutionary algorithms for solving multi-objective problems*. Springer.

de Boor, C. (2001). *A Practical Guide to Splines.* Springer.

Dean, J. and Ghemawat, S. (2008). Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113.

Debar, H., Becker, M., and Siboni, D. (1992). A neural network component for an intrusion detection system. In *Research in Security and Privacy, 1992. Proceedings., 1992 IEEE Computer Society Symposium on*, pages 240–250.

Devroye, L. (1986). A note on the height of binary search trees. *Journal of the ACM (JACM)*, 33(3):489–498.

Devroye, L. (1987). Branching processes in the analysis of the heights of trees. *Acta Informatica*, 24(3):277–298.

Donoho, D. L. and Johnstone, I. M. (1995). Adapting to unknown smoothness via wavelet shrinkage. *Journal of the american statistical association*, 90(432):1200–1224.

Efron, B., Hastie, T., Johnstone, I., and Tibshirani, R. (2004). Least Angle Regression. *The Annals of Statistics*, 32:407–499.

Fan, J., Han, F., and Liu, H. (2014). Challenges of big data analysis. *National science review*, 1(2):293–314.

Fan, J. and Li, R. (2001). Variable Selection via Nonconcave Penalized Likelihood and its Oracle Properties. *Journal of the American Statistical Association*, 96(456):1348–1360.

Fiala, J. C. and Harris, K. M. (1999). Dendrite structure. *Dendrites*, pages 1–34.

Flajolet, P., Gao, Z., Odlyzko, A., and Richmond, B. (1993). The distribution of heights of binary trees and other simple trees. *Combinatorics, Probability and Computing*, 2(02):145–156.

Flajolet, P. and Odlyzko, A. (1982). The average height of binary trees and other simple trees. *Journal of Computer and System Sciences*, 25(2):171–213.

Flake, R. H. (1963). Volterra series representation of nonlinear systems. *American Institute of Electrical Engineers, Part II: Applications and Industry, Transactions of the*, 81(6):330–335.

Fréchet, M. (1948). Les éléments alétories de nature quelconque dans un espace distancié. *Ann. Inst. H. Poincare*, 10:215–310.

Fréchet, M., Appert, A., and Appert, A. (1936). L'arithmétique de l'infini. *Bull. Amer. Math. Soc. 42 (1936), 796 DOI: http://dx. doi. org/10.1090/S0002-9904-1936-06441-4 PII*, pages 0002–9904.

Friedman, J., Hastie, T., Höfling, H., and Tibshirani, R. (2007). Pathwise coordinate optimization. *The Annals of Applied Statistics*, 1(2):302–332.

Friedman, J., Hastie, T., and Tibshirani, R. (2010). Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software*, 33(1):1.

Fuchs, A. (2013). *Nonlinear Dynamics in Complex Systems*. Springer.

Givens, G. H. and Hoeting, J. A. (2012). *Computational statistics*, volume 708. John Wiley & Sons.

Goldberg, D. (1989). *Genetic Algorithms in optimization, search and machine learning*. Addison Wesley Publishing Company, New York.

Harris, T. (1952). First passage and recurrence distributions. *Trans. Amer. Math. Soc*, 73(3):471–486.

Harrison, M. T., Amarasingham, A., and Kass, R. E. (2013). Statistical identification of synchronous spiking. *Spike Timing: Mechanisms and Function*, page 77.

Haslinger, R., Pipa, G., and Brown, E. (2010). Discrete time rescaling theorem: determining goodness of fit for discrete time statistical models of neural spiking. *Neural computation*, 22(10):2477–2506.

He, B. (2005). *Neural Engineering*. Springer.

Holland, J. H. (1973). Genetic algorithms and the optimal allocation of trials. *SIAM Journal on Computing*, 2(2):88–105.

Huang, J., Ma, S., Xie, H., and Zhang, C. (2009). A group bridge approach for variable selection. *Biometrika*, 2:339–355.

Karmakar, S. B. (1979). Solution of nonlinear differential equations by using volterra series. *Indian J. pure appi. Math*, 10(4):421–425.

Koenker, R. and Hallock, K. (2001). Quantile Regression. *Journal of Economic Perspectives*, 15:143–156.

Komarek, P. (2004). Logistic regression for data mining and high-dimensional classification. *Robotics Institute*.

Korenberg, M. J. and Hunter, I. (1986). The identification of nonlinear biological systems: LNL cascade models. *Biological cybernetics*, 55(2-3):125–134.

Korenberg, M. J. and Hunter, I. W. (1996). The identification of nonlinear biological systems: Volterra kernel approaches. *Annals of biomedical engineering*, 24(2):250–268.

Liu, R. Y. (1990). On a notion of data depth based on random simplices. *The Annals of Statistics*, 18(1):405–414.

Maass, W. (1997). Networks of spiking neurons: the third generation of neural network models. *Neural networks*, 10(9):1659–1671.

Mairson, H. G. (1994). Generating words in a context-free language uniformly at random. *Information Processing Letters*, 49(2):95–99.

Mäkinen, E. (1999). Generating random binary treesa survey. *Information Sciences*, 115(1):123–136.

Marmarelis, V. Z. (1993). Identification of nonlinear biological systems using Laguerre expansions of kernels. *Annals of biomedical engineering*, 21(6):573–589.

Marron, J. S. and Alonso, A. M. (2014). Overview of object oriented data analysis. *Biometrical Journal*.

McCullagh, P. and Nelder, J. (1989). An outline of generalized linear models. In *Generalized linear models*, pages 21–47. Springer.

Mel, B. W. (1999). Why have dendrites? a computational perspective.

Mizuseki, K., Sirota, A., Pastalkova, E., and Buzsáki, G. (2009a). Theta oscillations provide temporal windows for local circuit computation in the entorhinal-hippocampal loop. *Neuron*, 64(2):267–280.

Mizuseki, K., Sirota, A., Pastalkova, E., and G., B. (2009b). Multi-unit recordings from the rat hippocampus made during open field foraging.

Nye, T. M. et al. (2011). Principal components analysis in the space of phylogenetic trees. *The Annals of Statistics*, 39(5):2716–2739.

Percival, D. et al. (2012). Theoretical properties of the overlapping groups lasso. *Electronic Journal of Statistics*, 6:269–288.

Phillips, C. and Warnow, T. (1996). The asymmetric median tree: a new model for building consensus trees. *Discrete Applied Mathematics*, 71(1):311–335.

Pitman, J. (2006). *Combinatorial stochastic processes*, volume 1875. Springer-Verlag.

Plesser, H. E. and Gerstner, W. (2000). Noise in integrate-and-fire neurons: From stochastic input to escape rates. *Neural computation*, 12(2):367–384.

Pyapali, G., Sik, A., Penttonen, M., Buzsaki, G., and Turner, D. (1998). Dendritic properties of hippocampal ca1 pyramidal neurons in the rat: Intracellular staining in vivo and in vitro. *Journal of Comparative Neurology*, 391:335–352.

Pyapali, G. and Turner, D. (1994). Denervation-induced dendritic alterations in ca1 pyramidal cells following kainic acid hippocampal lesions in rats. *Brain Research*, 652:279–290.

Pyapali, G. and Turner, D. (1996). Increased dendritic extent in hippocampal ca1 neurons from aged f344 rats. *Neurobiology of Aging*, 17:601–611.

Ramón y Cajal, S. (1995). Histology of the nervous system of man and vertebrates. *Oxford Univ. Press, New York*.

Ruppert, D., Wand, M. P., and Carroll, R. J. (2003). *Semiparametric regression*. Cambridge university press.

Schetzen, M. (1981). Nonlinear system modeling based on the Wiener theory. *Proceedings of the IEEE*, 69(12):1557–1573.

Schumaker, L. (1980). *Spline Functions: Basic Theory*. Wiley.

Schwarz, G. (1978). Estimating the dimensions of a model. *The Annals of Statistics*, 6:461–464.

Serfling, R. (2002). Quantile functions for multivariate analysis: approaches and applications. *Statistica Neerlandica*, 56(2):214–232.

Shen, D., Shen, H., Bhamidi, S., Muñoz Maldonado, Y., Kim, Y., and Marron, J. (2014). Functional data analysis of tree data objects. *Journal of Computational and Graphical Statistics*, 23(2):418–438.

Silvestrov, D. (2006). Limit theorems for randomly stopped stochastic processes. *Journal of Mathematical Sciences*, 138(1):5467–5471.

Sivanandam, S. and Deepa, S. (2007). *Introduction to genetic algorithms*. Springer.

Skwerer, S., Bullitt, E., Huckemann, S., Miller, E., Oguz, I., Owen, M., Patrangenaru, V., Provan, S., and Marron, J. (2014). Tree-oriented analysis of brain artery structure. *Journal of Mathematical Imaging and Vision*, 50(1-2):126–143.

Song, D., Chan, R., Marmarelis, V., Hampson, R., Deadwyler, S., and Berger, T. (2007). Nonlinear dynamic modeling of spike train transformations for hippocampal-cortical prostheses. *Biomedical Engineering, IEEE Transactions on*, 54(6):1053–1066.

Song, D., Chan, R. H., Marmarelis, V. Z., Hampson, R. E., Deadwyler, S. A., and Berger, T. W. (2009). Nonlinear modeling of neural population dynamics for hippocampal prostheses. *Neural Networks*, 22(9):1340–1351.

Song, D., Wang, H., Tu, C. Y., Marmarelis, V. Z., Hampson, R. E., Deadwyler, S. A., and Berger, T. W. (2013). Identification of sparse neural functional connectivity using penalized likelihood estimation and basis functions. *Journal of computational neuroscience*, pages 1–23.

Stein, R. B. (1965). A theoretical analysis of neuronal variability. *Biophysical Journal*, 5(2):173–194.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58:267–288.

Tu, C. Y., Song, D., Breidt, F. J., Berger, T. W., and Wang, H. (2012). Functional model selection for sparse binary time series with multiple inputs. *Economic time series: Modeling and seasonality. Boca Raton: Chapman and Hall/CRC*.

Volterra, V. (1930). Theory of functionals and of integral and integro. *Differential Equations*, pages 147–149.

Walter, S. (2011). *Defining Quantiles for Functional Data*. PhD thesis, The University of Melbourne.

Wang, H. (2003). *Functional data analysis of populations of tree-structured objects*. PhD thesis, University of North Carolina at Chapel Hill.

Wang, H. and Kai, B. (2014). Functional sparsity: Global versus local. *Statistica Sinica, in press*.

Wang, H. and Marron, J. (2007). Object oriented data analysis: Sets of trees. *The Annals Of Statistics*, 35:1849–1873.

Wang, Y., Marron, J., Aydin, B., Ladha, A., Bullitt, E., and Wang, H. (2012). A nonparametric regression model with Tree-structured response. *Journal of the American Statistical Association*, 107:1272–1285.

Wiener, N. (1958). Nonlinear problems in random theory. *Nonlinear Problems in Random Theory, by Norbert Wiener, pp. 142. ISBN 0-262-73012-X. Cambridge, Massachusetts, USA: The MIT Press, August 1966.(Paper)*, 1.

Wray, J. and Green, G. G. (1994). Calculation of the Volterra kernels of non-linear dynamic systems using an artificial neural network. *Biological Cybernetics*, 71(3):187–195.

Yuan, M. and Lin, Y. (2006). Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society, Series B*, 68:49–67.

Zou, H. and Li, R. (2008). One-step sparse estimates in nonconcave penalized likelihood models. *Annals of statistics*, 36(4):1509.