Fall 2017

A general purpose artificial intelligence framework for the analysis of complex biological systems

John I. Kalantari
University of Iowa

A General Purpose Artificial Intelligence Framework for the Analysis of Complex Biological Systems

by
John I. Kalantari

A thesis submitted in partial fulfillment
of the requirements for the Doctor of Philosophy
degree in Biomedical Engineering in the
Graduate College of
The University of Iowa

December 2017

Thesis Supervisor: Associate Professor Michael A. Mackey

Graduate College
The University of Iowa
Iowa City, Iowa

CERTIFICATE OF APPROVAL

PH.D. THESIS

This is to certify that the Ph.D. thesis of

John I. Kalantari

has been approved by the Examining Committee
for the thesis requirement for the Doctor of Philosophy
degree in Biomedical Engineering at the December 2017 graduation.

Thesis Committee:

Michael A. Mackey, Thesis Supervisor

Isabel K. Darcy

Michael J. Schnieders

Alberto M. Segre

Sarah C. Vigmostad

...for Mom and Dad.

Acknowledgments

Foremost, I would like to express my gratitude to my advisor and mentor, Dr. Michael Mackey for his continuous support and guidance. Mike's mentorship, insight, and depth of knowledge helped me develop into the researcher I am today. I will always feel fortunate to have been one of his doctoral students. I want to thank my family, especially my parents Mali and Robert, my fiancee Monica, my sister Anna and my brother Nick, who have all been there for me from the beginning of this journey, witnessing my many triumphs and tribulations. I would not have persevered without your unconditional love and support.

Abstract

This thesis encompasses research on Artificial Intelligence in support of automating scientific discovery in the fields of biology and medicine. At the core of this research is the ongoing development of a general-purpose artificial intelligence framework emulating various facets of human-level intelligence necessary for building cross-domain knowledge that may lead to new insights and discoveries. To learn and build models in a data-driven manner, we develop a general-purpose learning framework called Syntactic Nonparametric Analysis of Complex Systems (SYNACX), which uses tools from Bayesian nonparametric inference to learn the statistical and syntactic properties of biological phenomena from sequence data. We show that the models learned by SYNACX offer performance comparable to that of standard neural network architectures. For complex biological systems or processes consisting of several heterogeneous components with spatio-temporal interdependencies across multiple scales, learning frameworks like SYNACX can become unwieldy due to the the resultant combinatorial complexity. Thus we also investigate ways to robustly reduce data dimensionality by introducing a new data abstraction. In particular, we extend traditional string and graph grammars in a new modeling formalism which we call Simplicial Grammar. This formalism integrates the topological properties of the simplicial complex with the expressive power of stochastic grammars in a computation abstraction with which we can decompose complex system behavior, into a finite set of modular grammar rules which parsimoniously describe the spatial/temporal structure and dynamics of patterns inferred from sequence data.

Public Abstract

This thesis encompasses research on Artificial Intelligence in support of automating scientific discovery in the fields of biology and medicine. At the core of this research is the ongoing development of a general-purpose artificial intelligence framework emulating various facets of human-level intelligence necessary for building cross-domain knowledge that may lead to new insights and discoveries. To learn and build models in a data-driven manner, we develop a general-purpose learning framework called Syntactic Nonparametric Analysis of Complex Systems (SYNACX), which uses tools from Bayesian nonparametric inference to learn the statistical and syntactic properties of biological phenomena from sequence data. We show that the models learned by SYNACX offer performance comparable to that of standard neural network architectures. For complex biological systems or processes consisting of several heterogeneous components with spatio-temporal interdependencies across multiple scales, learning frameworks like SYNACX can become unwieldy due to the the resultant combinatorial complexity. Thus we also investigate ways to robustly reduce data dimensionality by introducing a new data abstraction. In particular, we extend traditional string and graph grammars in a new modeling formalism which we call Simplicial Grammar. This formalism integrates the topological properties of the simplicial complex with the expressive power of stochastic grammars in a computation abstraction with which we can decompose complex system behavior, into a finite set of modular grammar rules which parsimoniously describe the spatial/temporal structure and dynamics of patterns inferred from sequence data.

Contents

List of Tables

List of Figures

Motivation

Modern advances in high-throughput and ”–omics” technologies have generated large amounts of data, capturing novel information about various complex biological systems and processes across multiple spatial and temporal scales. Recent breakthroughs in deep-learning and artificial intelligence have been shown to excel in using this data to build models ranging from image classification models for subcellular protein localization [1], to predictive models for regulatory genomics [2]. These deep-learning frameworks and the neural-network architectures on which they are based, present tremendous opportunities for advancement in modeling and analyzing complex phenomena in the fields of biology and medicine [3]. However, in the absence of expert supervision or annotated data, significant challenges continue to complicate both learning and predictive modeling of many biological systems/processes due their complex interdependencies. The domain of complex biological systems and processes include biological phenemona such as gene regulatory networks, cellular processes and infectious diseases, which can be characterized by a network of interactive and dynamic components often including signaling transduction pathways and positive and negative feedback loops across multiple scales of resolution. These complex interdependencies complicate the learning process by introducing uncertainty about the data used and the models constructed. While data uncertainty can be attributed to a lack of examples and imperfections in the measurement process, model uncertainty arises from having insufficient prior knowledge about the pertinent variables and multi-level relations inherent to a given system.

This thesis, explores an alternative approach to Artificial Intelligence called the *Syntactic Non-parametric Analysis of Complex Systems* (SYNACX), with which computational models can be

learned, despite these uncertainties, directly from data and without expert supervision. We postulate that by emulating those facets of human intelligence which enable general-purpose learning, we can:

1) infer subtle and latent patterns from various data sources over time, given limited *a priori* knowledge; and
2) construct computational models that can be validated and integrated with existing knowledge previously learned or found in the literaure.

As a proof of concept, the underlying theory and implementation of this approach is evaluated in online prediction tasks using biomedical data, specifically electrocardiography (ECG) waveforms and live-cell imaging data sequences. The content of this thesis is as follows:

- Chapter 1 briefly presents the key concepts adopted in development of the methodologies described in this thesis.
- Chapter 2 introduces the use of topological structures as the primary data abstraction in a new modeling formalism called Simplicial Grammar.
- In Chapter 3 presents the SYNACX algorithm and the advanced Bayesian probability model designed for unsupervised learning and incremental model construction in high-throughput data applications.
- Chapter 4 evaluates the feasiblity of Simplicial Grammar and SYNACX in online prediction and classification tasks for biomedical datastreams.
- Chapter 5 presents a final discussion of the analyses presented in the thesis and a description of future work.

CHAPTER 1

Background

In general, pattern-recognition methods can be broadly classified as being one of two approaches: statistical or syntactic [4, 5]. Statistical approaches are based on the statistical modeling of data and the application of probability theory and decision theory. Despite being the more popular approaches, statistical methods suffer from a significant shortcoming in their handling of contextual or structural information in patterns. While many statistical approaches struggle to incorporate complex relational information as discriminating features, syntactic pattern recognition methods excel in tasks where such information is abundant [5]. Syntactic methods enable the modeling of complex patterns through recursive decomposition of a problem into progressively simpler subpatterns, the interrelationships in the hierarchy of decompositions being well defined [6]. Rather than regarding patterns as numeric vectors in a finite-dimensional vector space, syntactic approaches describe patterns in terms of very simple sub-elements, called *pattern-primitives*, and a set of *rules* governing the relationships among them. The collection of primitives and rules together form the *grammar* which characterizes a specific class of patterns. Thus, in syntactic pattern recognition, the inference of a set of generative grammars for each class of patterns results in the design of a pattern classifer. The collection of patterns and the processes by which they are generated, provide the basis of a pattern recognition system that can be used for modeling complex systems. Despite an initial popularity, syntactic approaches have been largely ignored due to the computational intractability of inferring the appropriate stochastic grammars.

As will become evident in subsequent chapters, our primary objective is to construct a computational framework and modeling formalism which may enable a computer to emulate a expert's

ability to elucidate critical patterns, derive predictions and integrate knowledge learned from multiple modalities. The proposed approach combines statistical and syntatic approaches via the use of Bayesian nonparametric models for the purposes of inferring the implicit temporal dependencies inherent to a heterogeneous data stream, and explictly modeling them as a probabilistic grammar depicting a collection of oriented simplex chains. In this chapter, we introduce concepts from Grammar-based modeling, Probabilistic modeling and Algebraic topology, which are combined for the purpose of building an Artificial General Intelligence framework.

1.1 Artificial General Intelligence and General-Purpose Learning

Artificial Intelligence (AI) is the interdisclipinary study of the mechanisms underlying thought and intelligent behavior and their simulation in computer software algorithms [7]. The original goal of the field of Artificial Intelligence has been the construction of software frameworks with demonstrable 'human-level general intelligence' [7–9]. This goal has been found to be challenging [10–12]. As a result, AI research as of late has instead focused on the production of frameworks displaying domain-specific intelligence on a narrow range of issues and highly constrained tasks. The focus of these 'narrow AI' frameworks have helped the field explore critical methodologies, and produce results that have attracted wide attention to the field. In addition, the sub-field of Artificial Intelligence known as Machine Learning helps to achieve a set of goals similar to those of AI but with a focus on the development of algorithms that allow computers to automatically learn from data. The direct objective of this sub-field is to develop software programs for specific practical learning tasks in application domains [8]. In recent years, with the rapid pace of technological advancement and decreasing costs of entry for high-throughput computing, interest in the original goals of AI and recognition of the necessity and feasability of achieving 'human-level intelligence' has resulted in growth in the field of Artificial General Intelligence (AGI) [12]. AGI research differs from ordinary AI research by emphasizing the versatility and general-purpose nature of intelligence, and the inclusive development of computational frameworks that can combine five critical facets of human-level intelligence: multi-task learning, transfer learning, one-shot learning, multi-modal learning, and reinforcement learning [12–14].

1.1.1 Multi-task learning

The development of an AGI framework with the ability to multi-task is critical in the pursuit of 'human-level intelligence'. This seemingly effortless ability in humans allows for several hundred tasks to be performed concurrently. One naive approach to achieve this is to learn different tasks independently using multiple models. However such an approach does not exploit the interdependencies between related tasks. A more elegant solution is the development of a framework that can learn multiple tasks simultaneously. This idea of multi-task learning in the machine learning literature focuses on the design of a single learning algorithm that can learn multiple tasks simultaneously, with the goal of each learned task acting as an aide during the learning of other related tasks. In recent years, research has progressed in the development of multi-task learning algorithms, particularly in Natural Language Processing (NLP) [15, 16] and Question-Answering [17] domain tasks. Despite their success in the aforementioned domains, current multi-task learning algorithms are limited to tasks which utilize the same data type (i.e. text). Thus methods with the ability to multi-task with several heterogeneous data sources (e.g., image, text, and speech signal) remains elusive.

1.1.2 Transfer Learning

In the computer science and cognitive psychology literature, transfer learning refers to the human ability to transfer learned experiences from one task to another task such that the new task can be learned more efficiently [18]. Thus, in a sense, multi-task learning can be viewed as a transfer-learning problem, in which, the goal is to transfer knowledge between various tasks in order to support learning in one another [19].

1.1.3 One-Shot Learning

While the successful development of algorithms for multi-task and transfer learning is a technological improvement in itself, these methods must also be scrutinized under learning conditions similar to humans in order to achieve human-level performance. Specifically, they must demonstrate the ability to make accurate inferences given only limited exposure to a concept, category or

situation. This ability to learn from one or very few data examples is referred to as one-shot learning [20, 21]. Despite growing interest in evaluating the one-shot learning ability of deep learning frameworks when applied to image and static data [22], research investigating one-shot learning methods designed specifically for sequence data remains minimal [23], necessitating further research in one-shot *sequence* learning methodologies.

1.1.4 Multimodal Learning

As was previously mentioned, most of the existing research focuses on multi-task transfer learning across similar data sources [24]. However, a true general-purpose learning algorithm must demonstrate the ability to work with multiple modalities of data (e.g. text, image, speech, and video). In recent years, researchers have started looking into this problem [25–28] but a general-purpose algorithm which can take an arbitrary set of modalities as input continues to remain elusive.

1.1.5 Reinforcement Learning

The last facet of human-intelligence to be considered here reflects the ability of humans to learn many tasks without direct supervision by learning through interactions with their environment. This ability to interact with the environment and learn through feedback is called reinforcement learning. In an AGI framework this amounts to learning a policy for how to interpret input data by applying a reward function to new experiences and finding the policy that maximizes the total reward. As one of the oldest fields in machine learning, reinforcement learning has been applied to various domains and recently attained wide popularity due to success in unsupervised game-play tasks [29–31].

1.2 Neural Networks

A Neural Network, or Artificial Neural Network, is a computational model consisting of a set of nodes, connected with weighted interconnections, whose connectivity tries to replicate the structure of neurons in the human brain. With such a weighted connectivity structure, the system is highly adaptive to new information flows. Neural network models offer a nonlinear, adaptive modeling approach, in which, the architecture and the parameters are determined by a dataset. The nodes

of a neural network model are organized into layers: the input layer, one or more hidden layers and the output layer. Each node is characterized by its own set of connection strengths, activation function and threshold value. The effect of a set of input signals $\vec{x}$ on a node is equal to the product $\vec{w} \cdot \vec{x}$, where $\vec{w}$ are weights which can take on any real value. Research on artificial neural networks has undergone various periods of extensive activity, beginning with the pioneering work of McCulloch and Pitts on the perceptron [32], followed by the introduction of Rosenblatt's perceptron convergence theorem [33], and the work of Minsky and Papert showing the limitations of the simple perceptron [34]. This early work has led to the introduction and reinvention of various neural network architectures. Among the most popular neural network architectures is the feedforward neural network, where information flow is unidirectional and the output of each node is calculated by passing the sum of the weighted signals from incoming nodes through an activation function. The activation a_i, or net input of the node is given as the sum of all weighted inputs from incoming nodes j:

$$a_i = \sum_{j=1}^{N} w_{ij} x_j \tag{1.1}$$

By subtracting the sum of weighted inputs by the node threshold ϑ_i, we can determine the output signal y_i. This output signal's profile is described by the activation function $A()$:

$$y_i = A(a_i) = A(\sum_{j=1}^{N} w_{ij} x_j - \vartheta_i) \tag{1.2}$$

In addition to the activation function's original binary form, several continuous and nonlinear activation functions may be used. Among the most common activation functions is the sigmoid function:

$$A(a_i) = \frac{1}{1 + e^{-a_i}} \tag{1.3}$$

Prior to its implementation on a test dataset, the feed-forward neural network endures a training process using a learning algorithm such as backpropagation, where the node weights at each previous layer are recursively modified such that their error is reduced. This algorithm consists of computing the error contribution of hidden-layer nodes by transmitting the error computed at the output-layer nodes back to the hidden-layer nodes through the same weighted connections used to

propagate activation signals from hidden-layer to output-layer nodes. For networks designed with an arbitrary number of layers and nodes, this core algorithm performs gradient descent of the error function E. This function uses a set of training patterns composed of pairs of input vectors $\vec{x}^p$ and desired output vectors $\vec{z}^p$, to evaluate the set of weights that best correspond to the desired response profile. This evaluation requires calculating the mean quadratic error between the desired response profile and the actual neural network output signal:

$$E = \frac{1}{2}\sum_p \sum_i (z_i^p - y_i^p)^2 = \frac{1}{2}\sum_p \sum_i (z_i^p - \sum w_{ij} x_j^p)^2 \tag{1.4}$$

where p denotes a training pattern, and y the actual neural network output signal.

Among the most prominent feed-forward neural networks are the two-layer perceptron and the three-layer radial-basis function neural network [35]. The perceptron works as a binary classifier without any hidden layers, however, once it consists of three or more layers, with at least one hidden layer, the perceptron is referred to as a multilayer perceptron. As the name implies, the radial-basis function neural network uses the radial-basis function as an activation function. This type of function has a radial symmetry such as a Gaussian function, which causes the activation of a node to depend on its distance from a center vector, thus allowing it to respond to a local region of feature space. These static feed-forward networks, however, struggle on more complex classifications, and since they are mere input-output devices they cannot detect or produce temporal sequences [36, 37].

In problems, such as feature detection or time sequence formation, where dynamical behavior is necessary, either a dynamic neuron model [8] may be used, or nodes in already existing neural network are coupled with feedback connections. The underlying principle of dynamic neuron models is the use of a feedback loop between a node's input and output along with a temporal delay Δ. This temporal delay indicates the time between updates of activation levels at consecutive time steps, thus indicating the length of time a node can hold a specific activation level. Along with a weight μ_i, the feedback loop (recurrent connection) brings the current activation back to the input used to compute the next activation level. Thus, for each update step n, we compute the activation followed by the output signal:

1) $a_i(n+1) = \mu_i a_i(n) + \sum_{j=1} w_{ij} x_j(n)$

2) $y_i(n) = A(a_i(n) - \vartheta)$

Nevertheless, to provide the necessary temporal dynamics to an already existing static neural network, recurrent connections may be added from neighboring nodes in either the same layer or from higher layers. In such case the response profile of a node is given by:

$$y_i^t = A(\sum_j^N w_{ij} X_j^t + \sum_j^N r_{il}^{t-1} - \vartheta_i), \tag{1.5}$$

where q_l^{t-1} are the outputs of same-layer nodes at the previous time step, and r_{il}^{t-1} are the connections between them.

Most classic neural network models are typically supervised and offer only discriminative models that do not describe their input [38]. These discriminative models are a general class of machine learning models that are commonly used to model the dependence of an unobserved variable y on an observed variable x. As opposed to discriminative models, generative models are typically more flexible in expressing dependencies in complex learning tasks and are not inherently supervised [8]. Despite not allowing samples to be generated from the joint distribution of two variables, discriminative models yield superior performance for classification and regression tasks where this constraint is generally obsolete. These discriminative, or non-generative, approaches develop black-box models of the data that often map inputs to the correct answer by requiring a large quantity of training data that typically leads to over-fitting. While newer forms of neural network such as the Boltzmann machine do offer stronger generative semantics [39], they typically use stochastic sampling to learn a probability distribution. In contrast to those neural network models that adapt to supervised learning algorithms, some neural network implementations use unsupervised learning during the training process. Unsupervised learning allows for the extraction of statistically significant information, or memorization and reconstruction from the distribution of input patterns. Without any feedback from the user or environment, unsupervised learning consists of detecting common or distinctive features that allow the neural network to classify the input patterns. Some of the statistical operations performed to learn specific properties of the input pattern distribution include principle component analysis, probability density function parameter

estimation, and computation of correlation indices. However, in order for these operations to detect structure, input pattern distributions must present sufficient redundancy. Many neural network machine learning models also include the use of gradient descent methods in their implementations. Gradient descent is an iterative optimization procedure used to minimize an objective function J_θ parameterized by a given machine learning model's parameters $\theta \in \mathbb{R}^d$. This method updates the model parameters in the opposite direction of the objective function gradient $\nabla_J J_\theta$ with respect to the parameters. Gradient descent algorithms utilize a learning rate η that determines the size of steps used to reach a (local) minimum. Given a large dataset, the calculation of the gradient can become costly since every data point must be processed before making a single step. An alternative approach, called stochastic gradient descent [40], updates θ sequentially with each observation i such that $\theta := \theta - \eta \cdot \nabla_\theta J_i(\theta)$. This approach reaches an optimal θ much faster and can be used for online learning, however, it results in a larger variance of the loss function and never fully converges to the local or global minimum. There are many variations of the stochastic gradient descent algorithm [40–42].

1.2.1 Neural Networks for Time-Series Predictive Modeling

Since neural networks are able to learn complex patterns and estimate both linear and nonlinear functions, they are widely used for modeling and predicting time-series data. Predictive modeling in neural networks involves two phases: training and prediction. During training, the neural network performs supervised learning by presenting training data at the input layer and dynamically adjusting the model weights in order to achieve a desired output value. The prediction phase consists of presenting a new, unobserved input to the neural network and calculating the output, thereby predicting the outcome of the new input dataset. Traditional feed-forward neural networks often operate on a fixed-size window of a time-series data sequence and therefore cannot capture historical dependencies that fall outside of the data window [36, 43]. By contrast, Recurrent Neural Networks (RNNs), which are built on the same computational unit as the feed-forward neural network contain at least one feedback connection. Such recurrent/feedback connections allow for the network activation from a previous time step to be fed as input to influence predictions at the current time step. This process enables the models to perform temporal processing and sequence

learning. Recurrent neural network architectures can have many different forms. Some of the more simple recurrent neural network models often exploit the powerful non-linear mapping capabilities of the multi-layer perceptron, and have some form of memory, whereas others have a more uniform structure in which all neurons are interconnected. For simple architectures and models using deterministic activation functions, learning can be achieved using gradient descent procedures similar to those that enable the backpropagation algorithm for feed-forward networks. The standard backpropagation algorithm can be extended to perform gradient descent on a completely unfolded recurrent neural network via a technique known as backpropagation through time (BPTT). However, training recurrent neural networks with the backpropagation through time technique is often difficult due to the vanishing gradient and exploding gradient problems. The influence of a given input on the hidden layers, and therefore on the network output, either decays or grows exponentially when cycling around the network's recurrent connections. These problems impose limitations on a recurrent neural network's ability to model long range context dependencies [44]. To address these problems, a novel recurrent neural network architecture called Long Short-Term Memory (LSTM) [45] has been proposed. The long short-term memory architecture forces constant error through the internal state of special memory units called memory blocks. Each memory block contains memory cells with self-connections which attempt to store the temporal state of the network. Access to these memory cells is protected by special logistic units called gates, which control the flow of information. Each memory block contains an input gate to control the flow of input activations into the memory cell, an output gate to control the output flow of cell activations into the rest of the network and a forget gate. The forget gate scales the internal state of the memory cell before subsequently feeding it back to the cell as input through self-connections, thereby adaptively forgetting or resetting the cells memory. A slightly more dramatic variation on the long short-term memory architecture is the Gated Recurrent Unit (GRU) [46]. Among the most significant changes made by this architecture are the combining of the forget and input gates into a single update gate and the merging of the cell state and hidden state. The resulting model is simpler than standard long short-term memory models, and has been growing increasingly popular in sequence prediction and sequence labeling tasks [47–49].

1.3 Grammar-based Modeling

1.3.1 String Grammar

In formal language theory, a language L is defined as a (possibly infinite) set of finite length strings (sequences of elements) over a finite alphabet [50]. An *alphabet* is specified by giving a finite set, Σ, whose elements are called *pattern-primitives* (or symbols). Any set qualifies as a possible alphabet, as long as it is finite. Σ^* denotes the set of all finite strings of primitives from Σ, including the empty string. Elements of L are called 'words', 'phrases', or 'sentences' in formal language theory, linguistics, or bioinformatics, respectively. A string grammar of such a language is a finite device which generates all and only the strings of L in a deterministic manner [51]. Given such definitions, a primary goal of grammar-based modeling is to construct a system of rules from which the language can be derived. Put simply, a grammar is not merely a description of a language; it is an explanatory theory about the structure and patterns of a language–i.e., why a language has the properties it does. To acquire a language, one must devise a hypothesis compatible with the observed data and infer the grammar(s) appropriate for the data currently available [50, 52].

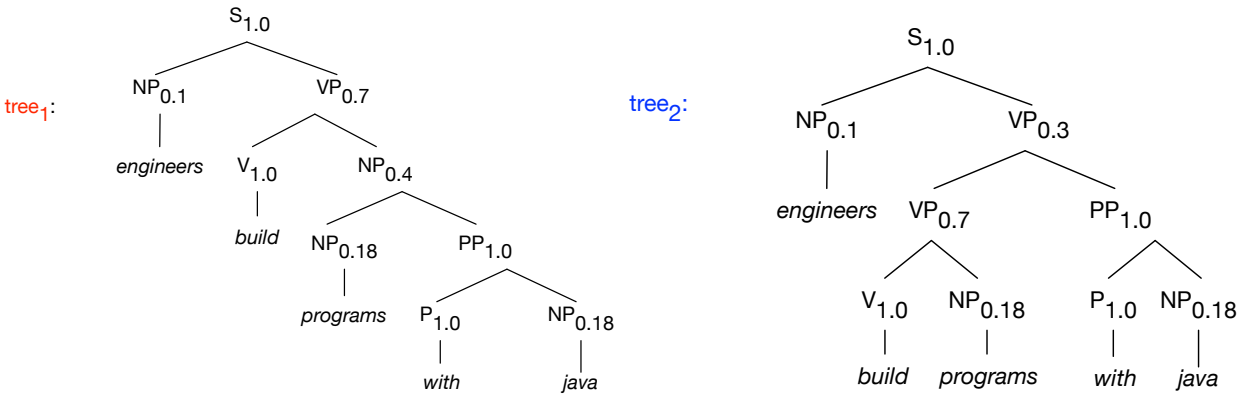
DEFINITION 1.1 (Context-Free Grammar) A *context-free grammar* G is a formal system that generates a language $L(G) \subseteq \Sigma^*$. It uses a set V of non-terminal symbols (one of which serves as the axiom), and a set of production rules that have the form $X \rightarrow \alpha$, where $X \in V$ and $\alpha \in (V \cup \Sigma)^*$.

The production rules of a grammar are used to derive the words/strings of the language, starting from the axiom. In this process, the production rules are used as rewrite rules. In constrast to the non-terminal symbols from V (denoted by upper-case letters), the primitives from Σ are called terminal symbols, because once generated during a derivation, they are never replaced. A derivation of a word $w \in L(G)$ begins with the axiom symbol, and incrementally replaces one of the nonterminal symbols in the emergent string according to one of the productions. Each derivation can be subsequently represented in the form of a tree. The inner nodes of the tree are labeled with the production names and terminal symbols are depicted as leaf nodes. By considering these leaf nodes from left to right, the string $w \in L(G)$ produced by the derivation can be obtained. Given a context-free grammar G and string $w \in \Sigma^*$, the process known as syntax checking or parsing con-

sists of constructing a derivation tree for w in order to determine whether $w \in L(G)$. The derivation tree constructed for a given $w \in \Sigma^*$ during parsing is called a parse tree. A context-free-grammar is considered syntactically ambiguous, if there is more than one parse tree for any given w. In such case, the set of all parse trees for w is denoted by $T_G(w)$. While syntactic correctness, $w \in L(G)$, is taken for granted in some domains such as bioinformatics and natural language prcessing, in others ambiguous grammars are considered a nuisance and actively avoided [50]. In situations where a large number of parse trees for a given $w \in \Sigma^*$ exist, a scoring scheme and objective function can be used to select the most plausible tree. When this scoring scheme is based on a probabilistic model, we obtain stochastic context-free grammars. Example parse trees for the same string using a probabilistic scoring scheme are shown in Figures 1.1a and 1.1b.

DEFINITION 1.2 (Stochastic Context-Free Grammar) A *stochastic context-free grammar* G is a context-free grammar which associates a transition probability p_r with each production rule r, such that for all $A \in V$, with $A \rightarrow \alpha_1|\alpha_2|...|\alpha_k$, named $r_1, r_2, ..., r_k$, $\sum_{i=1}^{k} p_{r_i} = 1$. The probability of a parse tree t, $P_G(t)$, is given as the product of the p_{r_i} for all uses of productions r_i in t. The probability of a word w assigned by grammar G is then defined as $P_G(w) = \sum_{t \in T_G(w)} P_G(t)$.

According to this definition, $P_G(w) = 0$ if $w \notin L(G)$ and under some conditions, a stochastic grammar defines a probability distribution on $L(G)$, i.e. $\sum_{w \in L(G)} P_G(w) = 1$ [53].



(a) Parse Tree 1 (b) Parse Tree 2

FIGURE 1.1 Parse Trees for Identical Strings using Probabilistic Scoring Scheme

1.3.2 Higher-order Grammars

In general, when learning a grammar, one has to decide the intended use of the grammar. Grammars have two applications: to parse or to generate a language. In addition to its use in linguistics, grammar induction is commonly used as an approach to data mining. While string grammars are very useful, they are limited to univariate datasets that can be represented as a sequence. Many datasets in biology and medicine, however, can have significant structural components that are sequential, non-sequential or a combination of both due to their inherent multi-scale relations and temporal dependencies. As a result, higher-order grammars using more complex data abstractions such as trees, graphs and matrixes have become increasingly popular alternatives to traditional string grammars since they can still represent the simpler feature vector-type datasets as well as sequential, multi-dimensional datasets [54–56].

One such higher-order grammar known as graph grammar is used to parse and generate graphs. A graph grammar is a mathematical structure that consists of an initial graph G_0 together with a collection of local production rules. This collection of rules can be thought of as binding rules and conformational changes used to model the construction of a graph or graph sequence [57]. The role of the production is to replace one sub-graph by another and act as local replacement operations on graphs. This process of replacing one sub-graph with another depends on a specification of the desired embedding: edges to/from a predicate sub-graph must be transformed into edges to/from the succeeding sub-graph. (This is in contrast to string grammars, which do not require explicit specification of an embedding: there is only one way to insert a replacement string substring into a host string.)

Let G denote a labeled graph: $G = (V, E, l)$ where V is a set of vertices, $E \subset V \times V$ is a set of edges, and $l : V \to \Sigma$ is a labeling function over some alphabet Σ . We denote by V_G, E_G, l_G the vertex set, edge set and labeling function of the graph G or by the triple (V, E, l). We assume basic definitions from graph theory such as connectivity, isomorphism, and embedding. More specifically, a rule is a pair of graphs $r = (g_L, g_R)$ where $V_{g_L} = V_{g_R}$. The graphs g_L and g_R are called the left hand side and right hand side of r respectively. The size of r is $|V_{g_L}| = |V_{g_R}|$. Rules are said to be *constructive* if $(E_{g_L} \subset E_{g_R})$. A rule r is applicable to a graph G if there exists a

label-preserving embedding $h : g_L \rightarrow G$. An action of a rule r on a graph G is a pair (r, h) such that r is applicable to G via h. The application of (r, h) to G yields a new graph $G' = (V', E', l')$ defined by replacing the image of g_L in G with a copy of g_R. More formally, G' is defined thus:

$$V' = V$$

$$E' = (E - h(x), h(y)|x, y \in g_L \cup h(x), h(y)|x, y \in g_R)$$

$$l'(x) = \begin{cases} l(x) & if x \notin h(v) \\ l_{g_R} \circ h^{-}1(x) & otherwise \end{cases}$$

We write $G \xrightarrow{r,h} G'$ to denote that G' was obtained from G by the application of (r, h).

As an example, consider the development of a graph grammar used to describe the behavior of a system of particles over time. In this setup, imagine each particle can take one of three conformations corresponding to primitives in our alphabet, $\Sigma = \{v_a, v_b, v_c\}$. Let us assume we begin with eight particles, each of which displaying conformation v_a. Thus, at the initial timestep, t_0, the graph G_0 modeling our system depicts the system of particles as a set of labelled vertices with no edges. The example set of production rules for this grammar, denoted Φ, define how this graph representation can evolve. In this particular set, the rules are considered *constructive* and describe the process of taking a pair of vertices within a graph, creating an edge between them and changing their conformation. Furthermore, graph grammar rule sets can also include production rules describing the destruction (r_{d_*}), relabelling (r_{l_*}) and higher-order aggregation (r_{a_*}) of various size sub-graphs. It should be noted that restrictions can be imposed on a graph grammar application such that only one rule can be applied at a time or the validity of a rule set be dependent on the occurence of a specific sub-graph. Figure 1.2 depicts an example assembly sequence for the given graph grammar model.

$$\Phi = \begin{cases} r_{c_1} : v_a v_a \rightarrow v_b - v_b \\ r_{c_2} : v_a v_b \rightarrow v_b - v_c \\ r_{c_3} : v_b v_b \rightarrow v_c - v_c \end{cases}$$

$$r_{d_*} : v_b - v_b \rightarrow v_a v_a$$

$$r_{l_*} : v_b v_b \rightarrow v_a - v_c$$

$$r_{a_*} : v_c - v_c - v_c \rightarrow v_c - v_a - v_c$$

1.4 Probabilistic Modeling

The amount of information that is available in the form of unstructured and semistructured data keeps increasing at an unprecedented rate. As the number of these rich datasets continues to grow, researchers are becoming interested in learning increasingly complex information and interactions directly from the data [58]. Probabilistic modeling in general, and Bayesian approaches in particular, provide a unifying framework for flexible modeling that includes clustering, prediction, estimation, and uncertainty quantification [59–61].

1.4.1 Prerequisites of Probabilistic Modeling

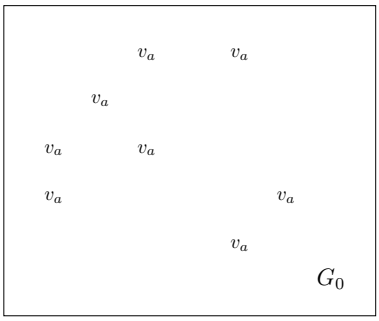
There are many complex phenomena in nature whose outcome cannot be predicted with certainty in advance but for which the set of all possible individual outcomes is known. These are referred to as random phenomena or random experiments, and are the primary focus of probabilistic modeling [62, 63]. The set of individual outcomes or sample points/ elementary events for a random experiment is known as the sample space of the experiment and denoted Ω. Any subset A of the sample space Ω is called an event, and is said to occur if the random experiment and the observed outcome $x \in A$. For the sake of brevity we present only those essential concepts underlying the methodologies discussed in subsequent sections of this dissertation.

DEFINITION 1.3 A δ-algebra F on set Ω, is a set of all subsets on Ω satisfying the following conditions:

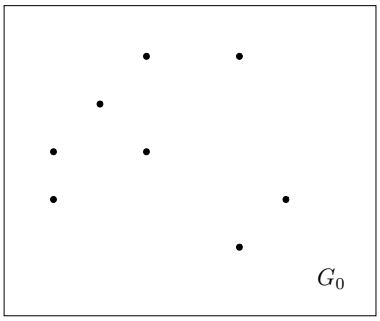
1) $\Omega \in F$;

2) $A \in F \implies A^c \in F$, where $A^c = \Omega - A$;

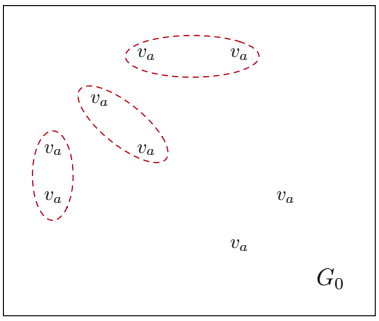
3) $A_1, A_2, ... \in F \implies \bigcup_n^\infty A_n \in F$;



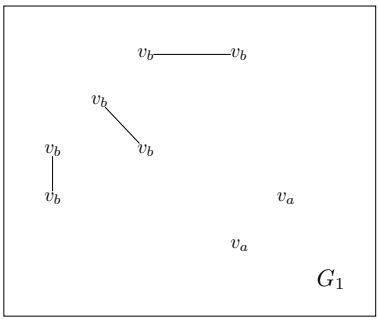
(a) Graph G_0 modeling the initial state of eight particle system



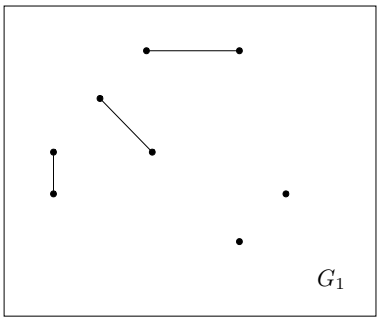
(b) Initial Graph G_0 as depicted graphically by vertices



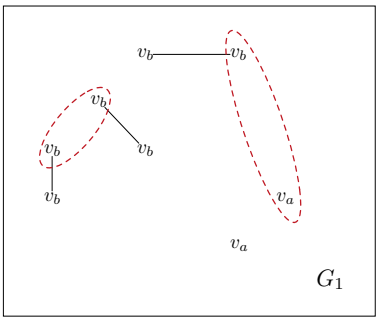
(c) Possible actions defined by rules applicable to G_0 (dotted clusters)



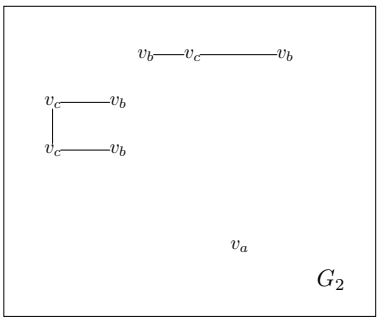
(d) Graph produced at timestep t_1, G_1, following application of rule set Φ to graph at timestep t_0, G_0



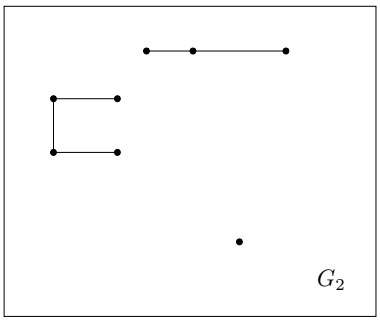
(e) Graph G_1 as depicted graphically by vertices



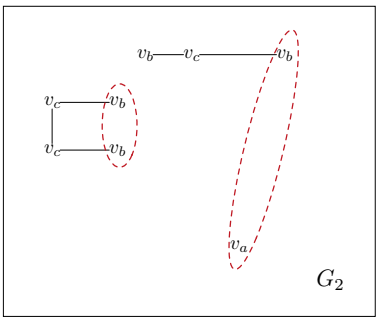
(f) Possible actions defined by rules applicable to G_1 (dotted clusters)



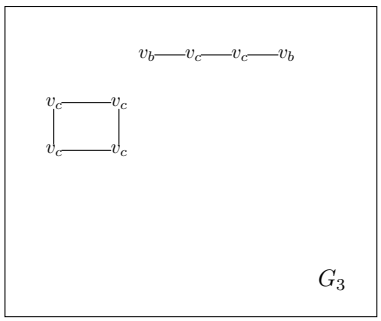
(g) Graph produced at timestep t_2, G_2, following application of rule set Φ to graph at timestep t_1, G_1



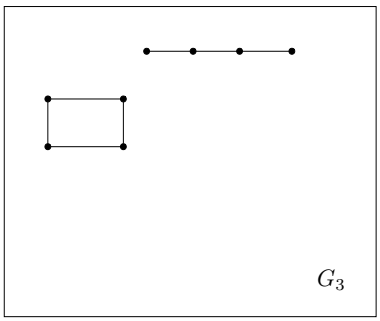
(h) Graph G_2 as depicted graphically by vertices



(i) Possible actions defined by rules applicable to G_2 (dotted clusters)



(j) Graph produced at timestep t_3, G_3, following application of rule set Φ to graph at timestep t_2, G_2



(k) Final Graph G_3 as depicted graphically by vertices

FIGURE 1.2 An example assembly sequence generated by a graph grammar model

From conditions (2) and (3), it follows that the empty set $\phi = \Omega^c \in A$ and thus the intersection $\bigcap_n^\infty A_n = (\bigcup_n^\infty A_n^c)^c \in F$.

DEFINITION 1.4 A *probability measure* P on the pair (Ω, F) is a function $P : F \to [0, \infty]$, satisfying the following properties:

1) $P(A) \geq 0$, for all $A \in F$;

2) $A \cap B = \phi \implies P(A \cup B) = P(A) + P(B)$;

3) $P(\Omega) = 1$

4) $B_1 \supset B_2 \supset ... \supset B_n \supset ...$ and $\bigcap_n^\infty B_n = \phi \implies \lim_{n\to\infty} P(B_n) = 0$.

The triple (Ω, F, P) is called a *probability space*.

DEFINITION 1.5 A *random variable* X on a probability space (Ω, F, P) is a function $X : \Omega \to R$, satisfying $\{\omega \in \Omega | X(\omega) \leq x\} \in F$, for all $x \in R$.

THEOREM 1.6 *Let $X_i, i = 1, 2, ..., n$, be discrete random variables on a probability space (Ω, F, P). Then the linear combination*

$$c_1X_1 + c_2X_2 + ... + c_nX_n$$

is a random variable on (Ω, F, P), for all $c_i \in R, i = 1, 2, ..., n$..

DEFINITION 1.7 A random variable $X : \Omega \to R$ on a probability space (Ω, F, P), is called discrete, if the set $D = \{x | P(X = x) > 0\}$ consists of either a finite set, $x_1, x_2, ..., x_n$, or an infinite countable set, $x_1, x_2, ...$, where the sum equals one

$$\sum_{x_i \in D} P(X = x_i) = 1 \tag{1.6}$$

DEFINITION 1.8 A distribution f_X of a discrete random variable X on probability space (Ω, F, P), is defined by

$$f_X(x_i) = P(X = x_i), \tag{1.7}$$

for all $x_i \in D = \{x | P(X = x) > 0\}$.

THEOREM 1.9 *If f_X is a distribution of a discrete random variable $X : \Omega \to R$, then*

1) $f_X(x) > 0$, for all $x \in R$ and $f_X(x) > 0 \Leftrightarrow x \in \{x_1, x_2, ..., x_n, ...\} \subset R$;

2) $\sum_{x_i \in D} f_X(x_i) = 1$.

DEFINITION 1.10 Let $X : \Omega \to R$ be a discrete random variable on a probability space (Ω, F, P). The *expected value* or *mean* of X, $\mu = E(X)$, is defined by

$$\mu = E(X) = \sum_{x_i \in D} x_i P(X = x_i), \tag{1.8}$$

provided that the sum equals

$$\sum_{x_i \in D} |x_i| P(X = x_i) < \infty, \tag{1.9}$$

where $D = \{x | P(X = x) > 0\}$.

DEFINITION 1.11 Let $X : \Omega \to R$ be a discrete random variable on a probability space (Ω, F, P). The *variance* of X, $\delta^2 = Var(X)$, is defined by

$$\delta^2 = Var(X) = E[(X - \mu)^2] = \sum_{x_i \in D} (x_i - \mu) P(X = x_i), \tag{1.10}$$

where μ is the expected value of X and $D = \{x | P(X = x) > 0\}$.

DEFINITION 1.12 A *stochastic process* with state space S is a sequence $\{X_t\}_{t \in T}$ of random variables $X_t \in S$ defined on the same probability space (Ω, F, P).

Here, the set T is called the parameter set and the index $t \in T$ represents time. Thus the state X_t, $t \in T$, can be thought of as the state of the process at time t. If T is finite or countable, the process is referred to as a discrete-time or discrete parameter process.

1.4.2 Bayesian Probability

In Bayesian statistics, probability is interpreted as a measure of our *belief in an event*, i.e., how confident we are in the occurence of an event [64]. Thus, if we believe with absolute certainty that an event will or will not occur, a belief of 1 or 0 will be assigned to the event, respectively. Otherwise, beliefs between 0 and 1 may be assigned to events in order to allow for weightings of other outcomes. This Bayesian interpretation of probability differs from traditional frequentist statistics where it is generally assumed that probability is the long-term frequency of events [64–66]. To clar-

ify this distinction, consider for example the *probability of car accidents* which under a frequentist interpretation is the *long-term frequency of car accidents*. While the logic behind this interpretation makes sense for the probability of some events, it becomes increasingly difficult to justify its application to events which do not have a long-term frequency of occurences. For example, in the case of political elections in the United States, it is very common to assign probabilities to election outcomes despite the fact that each election is a unique event and occurs only once. In both examples, the Bayesian interpretation of probability is more intuitive and applicable. In the case of the probability of a car accident, if an individual has observed the frequency of car accidents, his/her *belief* will also be equal to the long-term frequency, excluding any outside evidence. By using this definition of probability being a measure of belief, or confidence in the occurence of an event, it also becomes more appropriate to assign probabilities to election outcomes since we are evaluating an individual's confidence in a specific outcome. Under this Bayesian philosophy, inference constitutes updating one's beliefs after considering new evidence. It should be noted that this philosophy of using probability as a measure of belief that can be subsequently updated after seeing new evidence is reflected in human learning and reasoning [10, 64]. As we interact with the world around us; we observe partial truths, but gather evidence to form beliefs. In the 18th century, the Englishman and namesake of the field of Bayesian statistics, Thomas Bayes, became aware of a relation that is now known as *Baye's Theorem* [65]:

$$p(A|B) = \frac{p(B|A)p(A)}{p(B)} \propto p(B|A)p(A) \tag{1.11}$$

This theorem provides an abstract statement about conditional probabilities of events A and B, that when reinterpreted, provides a formula for Bayesian inference:

$$p(\theta|D) = \frac{p(D|\theta)p(\theta)}{p(D)} \tag{1.12}$$

Here, θ is an event and D is data which may give evidence for or against θ. This formula decomposes into four prinicipal terms:

1) The *prior* $P(\theta)$ is the probability of θ before the data is considered.

2) The *posterior* $P(\theta|D)$ is the probability of θ after the data is considered.

3) The *likelihood* $P(D|\theta)$ is the evidence about θ provided by the data D.

4) $P(D)$ is the total probability of the data when all possible events are taken into account.

If the prior and the likelihoood are known for all events, then (1.12) computes the posterior exactly. This is called the deductive logic of probability theory which provides a direct way for comparing events, drawing conclusions and making decisions. In most experiments however, the prior probabilities on the events are not known in which case inference must be performed [64–66]. Bayesian inference differs from traditional statistical inference by its preservation of uncertainty. Under the Bayesian paradigm, the introduction of prior uncertainty about events acknowledges that any guess made can be potentially wrong. Observing data, evidence, or other information, allows us to update our beliefs, and to make our guess *less* wrong. Thus, instead of completely discarding a prior belief after seeing new evidence, the prior is *re-weighted* in order to incorporate new evidence. As the number of instances N of evidence, or data, acquired increases, the less our prior belief matters. In addition, as $N \to \infty$, results obtained via Bayesian inference often converge with those obtained via frequentist methods. Hence for large N, traditional statistical inference is more or less objective. However, for a smaller N, inference becomes more *unstable* (i.e., frequentist estimates display more variance and larger confidence intervals). In such cases, the use of Bayesian methods via the introduction of a prior, return of probabilities (instead of a scalar estimate), allow us to preserve the uncertainty that reflects the instability of statistical inference of a small N dataset.

We can see this mathematically in (1.12), where the posterior distribution for a parameter θ, given a dataset **D** can be written as:

$$p(\theta|\mathbf{D}) \propto \overset{\text{likelihood}}{p(\mathbf{D}|\theta)} \cdot \overset{\text{prior}}{p(\theta)}$$

or, on the log scale,

$$log(p(\theta|\mathbf{D})) = c + L(\theta;\mathbf{D}) + log(p(\theta))$$

Here, we observe that the log-likelihood, $L(\theta;\mathbf{D}) = log(p(\mathbf{D}|\theta))$, scales with the sample size, since it is a function of the data, whereas the prior density does not. Therefore, as the sample size increases, the absolute value of $L(\theta;\mathbf{D})$ gets larger while $log(p(\theta))$ stays fixed (for a fixed value of

θ). The sum $L(\theta; \mathbf{D}) + log(p(\theta))$, as a result, becomes more heavily influenced by $L(\theta; \mathbf{D})$ as the sample size increases. As the sample size increases, the chosen prior has less influence [64]. Hence inference converges regardless of chosen prior, so long as the areas with non-zero probabilities are the same [65, 66].

1.4.3 Unsupervised Learning and Cluster Analysis

Clustering or *Cluster Analysis* is a process of finding similarities between a set of physical or abstract objects according to characteristics found in the data and grouping them into classes of similar objects called *clusters* [67]. In the absence of a class label, clustering is also called *unsupervised learning*, as opposed to supervised learning that includes classification and regression [68, 69]. Thus, clustering entails learning from observations rather than from annotated examples. Clustering is not associated with a particular algorithm but rather several different approaches to data modeling. Among the most popular methods found in the literature are, graph clustering models [70] in which datasets are organized based on the edge structure of observations, distribution models [71] which model the generative distributions of the data via the use of statistics and probabilities, centroid models [72] which represent groups as mean vectors, and connectivity models [73] which focus on the distance connectivity. Since clustering does not require the use of annotated datasets, it has become a powerful tool in numerous applications across many differerent fields [67].

1.4.3.1 Finite Mixture Modeling

An alternative approach to many traditional heuristic clustering methods is model-based clustering, in which, data is considered as coming from a mixture of probability distributions, each of which represents a different cluster [74, 75]. In other words, it is assumed that the data is generated by a mixture of distributions in which each component represents a different cluster. In the family of model-based clustering algorithms, one uses certain models for clusters and tries to optimize the fit between the data and the models. One such modeling approach called *finite mixture modeling* provides a convenient yet formal framework for model-based clustering, in which, a mixture of parametric distributions is used to model data, estimating both the parameters for the various dis-

tributions and the probabilities of cluster membership for a given observation. Following inference of the model and its parameters, the estimated mixing probabilities are treated as the prior probablity of an observation originating from a specific mixing distribution. Bayes rule is then used to allocate observations to clusters in accordance with their posterior probabilities, such that each observation is assigned to the cluster having the highest posterior probability (maximum posterior probability) of being where the observation originated. A benefit of using a model-based approach to clustering is that it enables assessment of the probabilities of events and simulation of draws from an unknown distribution within the framework of standard statistical theory.

1.4.3.2 Limitations of Clustering

In finite mixture models and other popular clustering methodologies, where the number of clusters k existing in the data is known *a priori*, a fundamental question that arises is how to perform model selection, i.e., how to determine the appropriate number of components for a given model. Many of these methods, from K-means clustering to Gaussian Mixture models, require specification of this parameter k prior to analysis. In addition, the results generated during the application of these methods are heavily influenced by the k value selected. Nevertheless, k is rarely known in real-world applications *a priori* and may change over time as more data becomes available [76].

While several algorithms such as X-means [77], Hierarchical clustering [73], and DBCSAN [78] have been developed to overcome this problem via estimation, outright avoidance of the problem, or not requiring direct specification, respectively, most of these techniques rely on heuristics and do not use a probabilistic framework. One alternative approach, however, allows for the dynamic estimation of k as well as its adaptation over time in the presence of new data. This approach, known as *Bayesian Nonparametric mixture modeling*, has been applied to a large number of different problems in machine learning and statistics and addresses many of the aforementioned limitations of clustering and unsupervised learning within a probabilistic framework [79, 80].

1.4.4 Bayesian Nonparametric Modeling

The probabilistic approach to statistical modeling provides an intuitive framework for expressing various aspects of uncertainty in a model [79]. In this approach, prior knowledge about the

model or model parameters is incorporated with the goal of inferring a posterior distribution over unobserved variables. Among the most common approaches, are Bayesian *parametric* formulations in which models define prior and posterior distributions on a single fixed parameter space and Bayesian *nonparametric* models in which the dimension of the parameter space is allowed to grow with data size. This ability to grow the model structure is possible with the use of an infinite-dimensional parameter space and the invocation of only a finite subset of the avaliable parameters on any given finite data set. More formally, Bayesian nonparametric models place a prior over the (infinite dimensional) space of distributions on an arbitrary space. When combined with data, the resulting posteriors give a distribution on structures that can grow with new observations. Bayesian nonparametric priors such as the Dirichlet process [81] and its extensions, e.g. Pitman-Yor Processes [82], allow for the creation of flexible probabilistic models with an unbounded number of parameters. These models are appropriate when the latent dimensionality of the data is unknown or may grow with sample size. Both distributions have found a number of applications in text and language modeling [83–85].

Bayesian Nonparametric models are relevant today because the kinds of latent structures and datasets encountered are becoming increasingly more complex. The flexiblity in model selection and adaptation offered by these models provides a solution to the problem of underfitting and overfitting in machine learning [79]. Specifically, unlike classical model selection where the number of components is fixed, the predictive distribution of Bayesian nonparametric models allows for a subsequent data point (i.e. the data point to be predicted) to exhibit a previously unseen component of the latent parameter space. In addition, the adaptive nature of Bayesian nonparametric methods allows for the modeling of uncertainty over complicated latent structures with hierarchical settings, such that they can grow and change in the predictive distribution. Lastly, by casting model selection and parameter estimation in a single model, Bayesian nonparametric models only need a single posterior inference algorithm to search and learn the best model [86].

1.4.4.1 Dirichlet Process

Imagine a Dirichlet—a random distribution over k elements—but for a distribution on any space (including continuous spaces). This is a Dirichlet process. The Dirichlet process (DP), developed

formally in [61], is a stochastic process whose realizations are probability distributions.

DEFINITION 1.13 (Dirichlet Process) Let (X, Ω) be a measurable space and μ be a measure on this space. A random probability measure P^μ on (X, Ω) is a *Dirichlet process* with parameter μ under the following condition: given a measurable partition of Ω, $\{B_1, B_2, ..., B_N\}$, where each $\mu(B_k) > 0 \, \forall k$, the joint distribution of random probabilities $(P^\mu(B_1), P^\mu(B_2), ..., P^\mu(B_N))$ is distributed according to the standard Dirichlet distribution $Dir(\mu(B_1), \mu(B_2), ..., \mu(B_N))$ [87]. The parameter μ can be interpreted as the product, $\mu = \alpha H$, where α is a real number > 0 referred to as the concentration parameter and H the base probability measure of the Dirichlet process.

It can be shown that $G \sim DP(\alpha, H)$ is a probability measure drawn from a Dirichlet process,

$$(G(B_1), G(B_2)..., G(B_N))|\alpha, H \sim Dir(\alpha H(B_1), \alpha H(B_2), ..., \alpha H(B_N)). \tag{1.13}$$

From this definition it can be shown how α controls the variance of G about its mean H, via

$$E[G(B)] = H(B), V[G(B)] = H(B)(1 - H(B))/(\alpha + 1) \tag{1.14}$$

Since the random probability measure G can be thought of as a probability distribution over Ω, which can be sampled to yield independent and identically distributed samples $\theta_1, ..., \theta_m \sim G$, it can be shown that a draw, G, is discrete with probability 1, and has the following form:

$$G = \sum_{i=1}^{\infty} \pi_i \delta_{\theta_i}, \tag{1.15}$$

where the weights π_i satisfy $\sum_{i=1}^{\infty} \pi_i = 1$, δ_x is a Kronecker delta point mass concentrated at x, and $\theta_i \sim H$ are the locations of the point masses.

This discrete nature of random probability measures drawn from a Dirichlet process is made explicit by [88] in the *stick-breaking construction.* The stick-breaking construction is an iterative procedure for sampling the weights π_i of (1.15). This procedure is based on the analogy of taking a stick of unit length, and repeatedly breaking off Beta-distributed pieces, and assigning them to locations θ_i in Ω. A draw from a Dirichlet process can be sampled by alternately sampling θ_i from

H, and calculating π_i by sampling β_i, as given by:

$$\pi_i = \beta_i \prod_{j=1}^{i-1}(1-\beta_j) \tag{1.16}$$

$$\beta_i \sim Beta(1, \alpha)$$

It is important to note that the sequence of random variables $\beta_1, \beta_2, ...$ constructed by (1.16) satisfy $\sum_{i=1}^{\infty} \pi_i = 1$ from (1.15) with probability one.

Another perspective on the construction of the Dirichlet Process is described by [81,87], where the discrete nature of observations from the Dirichlet Process allow observations from the distributions/measures drawn from a Dirichlet process to be viewed as countably infinite mixtures. Given a model that first draws a probability measure G from a Dirichlet process with parameter $\mu = \alpha H$ and then draws i.i.d observations θ_1, θ_2 from G, one can analytically integrate out G to obtain the following conditional distributions from the observations θ_m:

$$\theta_m|\theta_1, ..., \theta_{m-1} \sim \frac{\alpha}{\alpha + m - 1} H + \frac{1}{\alpha + m - 1} \sum_{i=1}^{m-1} \delta_{\theta_i} \tag{1.17}$$

This second perspective on the Dirichlet process is provided by the *Polya urn scheme*, where draws from the Dirichlet process exhibit a clustering property [89]. In this analogy we assume a non-transparent urn that contains colored balls from which we draw balls randomly. Three assumptions are made:

1) H is a distribution over colors
2) each θ_m represents a distinct color of ball placed in the urn
3) the process begins with an empty urn.

The algorithm for this scheme is as follows:

- With probability proportional to α, draw $\theta_m \sim H$ and add a ball of this color into the urn.
- With probability proportional to $m - 1$, draw a random ball from the urn, observe its color, place it back into the urn and add an additonal ball of the same color into the urn.

We can thus characterize the $m+1^{th}$ observation as being drawn with probability proportional to α

from H, and equal to a previously drawn observation θ_i with probability proportional to $\sum_{j=1}^{m} \delta_{\theta_i=\theta_j}$. This expression shows that there is a positive probability of different observations having the same exact value θ_k as one of the previous observations and therefore a positive reinforcement effect.

1.4.4.2 Dirichlet Process Mixture Modeling

The use of Dirichlet Processes in the construction of mixture models with infinite components can be thought of as taking the limit of the finite mixture model for i to infinity. Such a model takes on the following form:

$$G \sim DP(\alpha, H) \tag{1.18}$$

$$\theta_i|G \sim G \tag{1.19}$$

$$x_i|\theta_i \sim f(\theta_i) \tag{1.20}$$

As previously mentioned, the hyperparameters of the Dirichlet Process, $DP(\alpha, H)$, correspond to the base probability measure H and the concentration parameter α, where H is often the conjugate prior to the generative distribution f and α is a scalar value indicating the strength of belief in H. It should also be noted that α affects the number of clusters (components) generated, such that larger values of α result in more clusters, whereas smaller values result in less. G is defined by (1.15) and is sampled from the Dirichlet process. Each θ_i denotes a parameter vector sampled from G containing the parameters of a given cluster. The generative distribution f is parameterized by θ_i and used to generate the observations (datapoints) x_i. From these generative distributions we can define a mixture distribution of the countable infinite mixture $f_x(\cdot) = \sum_{i=1}^{\infty} \pi_i f(\cdot|\delta_{\theta_i})$ with mixing proportions π_i and mixing clusters $f(\cdot|\delta_{\theta_i})$. In addition, these cluster parameters θ_i can be viewed as latent variables on x_i indicating the orignating cluster for x_i and thereby the identity/configuration of said cluster. Thus, for every x_i, a θ_i is drawn from G, such that with every draw, G will change based on previous observations. As was shown in the Poly Urn scheme (1.17), G can be integrated out such that future draws of θ_i only depend on H. However, estimating θ_i using this formula is not always tractable since implementations must often enumerate through an exponentially increasing number of i clusters.

Given the aforementioned definitions and observations, it is desirable to estimate the posterior of the Dirichlet Process given the samples θ_i. Using Bayes rule and the conjugacy between the Dirchlet and Multnomial distributions [90], the posterior Dirichlet process can be described by:

$$G \sim DP(\alpha, H) \; and \; \theta|G \sim G \Leftrightarrow \theta \sim H \; and \; G|\theta \sim DP(\alpha + 1, \frac{\alpha H + \delta_\theta}{\alpha + 1}) \tag{1.21}$$

Thus, to perform clustering on data without specifying a fixed number of clusters in advance, the Dirichlet process can be used as a prior on the mixing probabilities of a mixture model [90, 91].

1.4.4.3 Chinese Restaurant Process Representation

The Dirichlet Process mixture model defined in Section 1.4.4.2 is mathematically sound, nevertheless it has a major drawback: for every new x_i that is observed, a new θ_i must be sampled, taking into account the previous values of θ. In many applications, sampling these parameters can be a difficult and computationally expensive task. Using a somewhat different, yet computationally tractable, metaphor of the Polya urn scheme, the Dirichlet Process can be alternatively represented by the *Chinese Restaurant Process* (*CRP*) [90]. The scheme uses the following analogy: Consider a Chinese restaurant with an unbounded number of tables. An observation, ϕ_i, corresponds to a customer entering the restaurant, and the distinct values θ_k^* correspond to the tables at which customers can sit.

$$\phi_i|\phi_1, ..., \phi_{i-1} = \begin{cases} \theta_k^* & \textit{with probability } \frac{c_{i-1}^{\theta_k^*}}{i-1+\alpha} \\ \textit{new draw from } H & \textit{with probability } \frac{\alpha}{i-1+\alpha} \end{cases}$$

Using the above conditional probability distribution, where α is the concentration parameter of the Dirichlet process and $c_{i-1}^{\theta_k^*}$ is the total number of customers sitting at a table with the distinct value θ_k^*, the Chinese Restaurant Process can be considered an induced distribution over partitions. Assuming an intially empty restaurant, the Chinese Restaurant Process algorithm can be expressed as follows:

- With probability proportional to $c_{i-1}^{\theta_k^*}$, the i^{th} customer sits at the table indexed by θ_k^*, in which case $\phi_i = \theta_k^*$.

- With probability proportional to α, the i^{th} customer sits at a new table, in which case $\phi_i \sim H$.

The result of the Chinese Restaurant Process is a distribution on the space of partitions of the positive integers. Thus, in this alternative Dirichlet Process mixture model representation, the latent variables ϕ_i of cluster *assignments* can be modeled. This way instead of using θ_i to denote both the cluster parameters and the cluster assignments, the latent variable will indicate the cluster *id* which can be subsequently used to assign the cluster parameters. As a result, it no longer becomes necessary to sample a θ for each new observation. Instead the cluster assignment is obtained by sampling ϕ_i from the Chinese Restaurant Process. With this scheme a new θ is sampled only when a new cluster needs to be created.

1.4.4.4 Pitman-Yor Process

The Pitman-Yor process (PYP) is a two-parameter extension of the Dirichlet process, parameterized by a discount parameter $0 \leq d \leq 1$, a concentration parameter $\alpha > -d$, and the base probability measure H [82, 92] . We can observe the behavior of a draw by considering the stick-breaking construction for the Pitman-Yor process:

$$\pi_k = \beta_k \prod_{j=1}^{k-1}(1-\beta_j) \; for \; k = 1, 2, ...\infty \tag{1.22}$$

$$\beta_k \sim Beta(1-d, \alpha + kd), \; for \; k = 1, 2, ...\infty$$

As d increases, the rate of decay of the ordered atom sizes will decrease. When $d = 0$, we recover the stick-breaking construction of the Dirichlet process given in (1.16.) Given G is a Pitman-Yor process with parameters α, d and base measure H, we can denote the mixture model as:

$$G \sim PYP(\alpha, d, H)$$

$$\theta_i | G \sim G$$

$$x_i | \theta_i \sim f(\theta_i)$$

If H is a smooth distribution and $\theta_1, \theta_2, ...$ are i.i.d draws from G, the distribution of θ_i conditioned on $\theta_1, ..., \theta_{i-1}$ and G marginalized out, can be generalized to follow the Polya urn scheme as follows:

$$\theta_i|\theta_1, ..., \theta_{i-1}, d, \alpha, H \sim \sum_{t=1}^{K} \frac{n_t - d}{\alpha + i - 1}\delta_{\theta_t^*} + \frac{\alpha + Kd}{\alpha + i - 1}H, \tag{1.23}$$

where θ_t^* denotes the t^{th} distinct value among $\theta_1, ..., \theta_{i-1}$, K the total number of distinct values, and n_t the total number of recorded observations of the value θ_t^*.

1.4.4.5 Hierarchical Pitman-Yor Process

We can also construct a hierarchy of Pitman-Yor processes, known as the *Hierarchical Pitman-Yor Process* (HPYP), that allows us to jointly cluster multiple related groups of data. Each group is associated with a Pitman-Yor process-distributed random measure. These group-specific Pitman-Yor processes are coupled via a shared, PYP-distributed *base measure* G_0. For J groups, each containing n_j data points, the $HPYP$ is defined as follows:

$$G_0 \sim PYP(\alpha_*, d_*, H) \tag{1.24}$$

$$G_j|G_0 \sim PYP(\alpha_0, d_0, G_0), j = 1, ..., J$$

$$\theta_{ji}|G_j \sim G_j, i = 1, ..., n_j$$

$$x_{ji}|\theta_{ji} \sim f(\theta_{ji})$$

This hierarchical construction generalizes to a multiple-level hierarchy (Figure 1.3).

1.4.4.6 Chinese Restaurant Franchise Representation of the Hierarchical Pitman-Yor Process

Similar to the Dirichlet Process, the Pitman-Yor Process and Hierarchical Pitman-Yor Process can be better understood via their *Chinese Restaurant Process* and *Chinese Restaurant Franchise* (*CRF*) interpretations, respectively [80, 93]. In the Pitman-Yor process variation of the Chinese Restaurant process, the first customer θ_1 sits at the first available table θ_1^* while each of the subse-

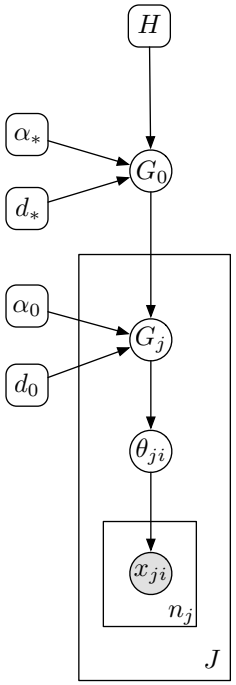


FIGURE 1.3 Plate notation for Hierarchical Pitman-Yor Process Model

quent customers θ_i either sit at the k^{th} occupied table θ_k^* with probability proportional to $(c_k - d)$ where c_k is the number of customers already sitting at that table, or sit at a new unoccupied table θ_{K+1}^* with probability proportional to $(dK + \alpha)$ where K is the current number of occupied tables. In the Chinese Restaurant Franchise representation of the Hierarchical Pitman-Yor process, the random measure labeled by the random variable G_j, corresponds to the j^{th} restaurant in the franchise, draws $\theta_{ji} \sim G_j$ correspond to customers, tables in restaurant j correspond to draws $\theta_{jt}^* \sim G_0$, and dishes correspond to draws $\theta_k^{**} \sim H$. In this analogy, customer seating is done independently in the restaurants and coupling among restaurants is achieved via the use of a single franchise-wide menu H. The first customer to sit at a table in a restaurant chooses a dish from the menu and all subsequent customers who sit at that table are served that dish. Dishes are chosen with probability proportional to the number of tables in the entire franchise which have already served that dish. Thus if the i^{th} customer in the jth restaurant, θ_{ji}, sits at the t^{th} table, t_{ji}, which serves dish k_{jt}, let $\theta_{ji} = \theta_{jt_{ji}}^* = \theta_{k_{jt_{ji}}}^{**}$

Let n_{jtk} be the number of customers in restaurant j seated at table t and eating dish k, m_{jk} be the number of tables in restaurant j serving dish k, and K be the number of dishes served throughout the franchise. The conditional distributions given by the Chinese Restaurant Franchise

for the Hierarchical Pitman-Yor process are described by a Polya urn scheme as follows:

$$\theta_{ji}|\theta_{j1},...,\theta_{j,i-1},\alpha,d,G_0 \sim \sum_{t=1}^{m_{j.}} \frac{n_{jt.}-d}{\alpha+n_{j..}}\delta_{\theta^*_{jt}} + \frac{\alpha+m_{j.}d}{\alpha+n_{j..}}G_0, \tag{1.25}$$

where dots are used to denote marginal counts. A draw from this mixture can be obtained by drawing from the terms on the right side with probabilities given by the corresponding mixing proportions. If a term in the first summation is chosen then the customer sits at an already occupied table: $n_{jt.}$ is incremented, θ_{ji} is set equal to θ^*_{jt} and $t_{ji} = t$ for the chosen t. If the second term is chosen then the customer sits at a new table: $m_j.$ is incremented by one, $n_{jm_{j.}}$ is set equal to 1, $\theta^*_{jm_{j.}} \sim G_0$ is drawn, and the variables are set s.t $\theta_{ji} = \theta^*_{jm_{j.}}$ and $t_{ji} = m_{j.}$. In Chapter 3, the predictive/posterior distribution for each distinct value (or pattern-primitive) given a sequence of observations is derived in terms of the Chinese Restaurant Franchise interpretation, using customer counts c and table counts t.

1.5 Algebraic Topology

The combinatorial nature of many complex patterns makes it is necessary to utilize a modeling formalism that does not adopt a strict order or parametric form, yet can be used for analyzing massive amounts of sampled data [94–96]. Acquiring knowledge about a sampled space from point data remains a key problem in many areas of science and engineering [94, 95, 97]. The sampled space could be a hidden manifold sitting in some high dimension, or could be a compact subset of some Euclidean space. The principle of algebraic topology is to attach algebraic invariants to topological spaces in order to classify them up to homeomorphism.

DEFINITION 1.14 (Topological Space) A *topological space* is a set X and a set τ of subsets of X satisfying the axioms:

AXIOM 1. *$\emptyset$ and X are in τ,*

AXIOM 2. *If the sets $U_1, U_2, ..., U_n$ are in τ, then so is $\cap_{i=1}^{n} U_i$,*

AXIOM 3. *If $U_i, i \in I$ are in τ, then so is $\cup_{i \in I} U_i$.*

A map f between topological spaces is said to be continuous if the inverse image of every open set is an open set. A homeomorphism is a continuous bijection whose inverse is also continuous. Two topological spaces $(X, \tau_X), (Y, \tau_Y)$ are said to be homeomorphic if there exists a homeomorphism $f : X \to Y$. From the viewpoint of topology, homeomorphic spaces are essentially identical. Properties of a topological space which are preserved up to homeomorphisms are said to be topological invariants. One can consequently study the latent properties of a discrete algebraic structure instead of studying a continuous domain directly, which would be hard to handle algorithmically. Topological information such as the rank of the homology groups, or their persistent behavior can divulge important features of these hidden spaces [94].

1.5.1 Simpicial Complexes

In algebraic topology, one studies simplicial complexes. A simplicial complex is a combinatorial structure that consists of a collection of simplices, which are elementary building blocks like vertices, edges, triangles, tetrahedral and higher-dimensional equivalents glued together along common faces. Simplicial complexes are a generalization of graphs in the sense that they allow higher order adjacency, i.e. more than two vertices being connected by a simplex. In this section we introduce and familiarize readers with the concepts and terminologies used in subsequent chapters.

DEFINITION 1.15 (Abstract Simplicial Complex) An *abstract simplicial complex* K is a pair $K = (V, S)$ where V is a finite set whose elements are called the vertices of K and S is a set of non-empty subsets of V that is required to satisfy the following two conditions:

1) $v \in V \Rightarrow \{v\} \in K$

2) $\sigma \in K, \tau \subseteq \sigma \Rightarrow \tau \in K$

Each element $\sigma \in K$ is called a simplex or a face of K and, if $\sigma \in K$ has precisely $d+1$ elements $d \geq -1$, σ is called a d-simplex and the dimension of σ is d. The dimension of the simplicial complex K is the largest dimension d of any simplex, $dim(K) = max\{dim(\sigma)|\sigma \in K\}$. The set of all the d-simplices of K is denoted by $K^{(d)}$.

DEFINITION 1.16 (Simplex) A *simplex* σ is a subset of the vertices from the vertex set, $\sigma = \{v_0, ..., v_d\} \subseteq V$, and we say that $v_0, ..., v_d$ are the vertices of σ.

DEFINITION 1.17 (Face) A *face* of a simplex $\sigma = \{v_0, ..., v_d\}$ is a simplex whose vertices form a subset of $\{v_0, ..., v_d\}$.

DEFINITION 1.18 (Ordered Abstract Simplicial Complex) An *ordered (oriented) abstract simplicial complex* K is an abstract simplicial complex in which the set V of vertices is ordered, such that $[v_{i_0}, ..., v_{i_d}]$ may stand for a simplex iff $v_{i_c} < v_{i_e}$ whenever $c < e$.

An example of an ordered simplicial complex is the ordered q-simplex itself. The ordered q-simplex is simply a q-dimensional simplex with ordered vertices. It is an ordered simplicial complex when considered together with its faces. If we denote the ordered q-simplex $[v_0, ..., v_q]$, then each k-face has the form $[v_{i_0}, ..., v_{i_k}]$, where $v_0 \leq v_{i_0} \leq v_{i_1} \leq ... \leq v_{i_k} \leq v_q$.

DEFINITION 1.19 (Proper Face) A *proper face* is a face that is different from σ.

DEFINITION 1.20 (Boundary of σ) *Boundary of* σ is the set of proper faces of maximal dimension $dim(\sigma) - 1$, s.t

$$\partial(\sigma) = \partial\{v_0, ..., v_d\} = \{\{v_0, ..., \hat{v_i}, ..., v_d\} : 0 \leq i \leq d\}$$

where the symbol $\hat{v_i}$ means that v_i is removed from the set.

We can define the boundary operator ∂_d for a simplex σ as follows:

$$\partial_d \sigma = \sum_{i=0}^{d} (-1)^i [v_0, ..., \hat{v_i}, ..., v_d],$$

where the signs (-), account for orientation.

DEFINITION 1.21 (Facets of σ) *Facets of* σ are faces of the boundary of a simplex.

DEFINITION 1.22 (Coface of σ) *Coface of* σ is a simplex $\tau \in K$ admitting σ as a face.

DEFINITION 1.23 (Dimension of the Simplicial Complex) *Dimension of the simplicial complex* is the maximum dimension of its simplices.

For each non-negative integer q, let $\Delta_q(K)$ be the additive group consisting of all formal sums of the form $n_1(\mathbf{v}_0^1, \mathbf{v}_1^1, \mathbf{v}_2^1, ..., \mathbf{v}_q^1) + n_2(\mathbf{v}_0^2, \mathbf{v}_1^2, \mathbf{v}_2^2, ..., \mathbf{v}_q^2) + ... + n_j(\mathbf{v}_0^j, \mathbf{v}_1^j, \mathbf{v}_2^j, ..., \mathbf{v}_q^j)$, where $n_1, n_2, ..., n_j$ are integers and $\mathbf{v}_0^*, \mathbf{v}_1^*, \mathbf{v}_2^*, ..., \mathbf{v}_q^*$ are (not necessarily distinct) vertices of K that span a simplex of K for $* = 1, 2, .., s$. Thus, $\Delta_q(K)$ can be regarded as the free abelian group gener-

ated by the set of all $(q+1)$-tuples of the form $(\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_q)$, where $\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_q$ span a simplex of K.

DEFINITION 1.24 (Subcomplex of a Simplicial Complex) A *subcomplex* of the simplicial complex K is a simplicial complex L, such that every face of L belongs to K, $L \subset K$. A subcomplex that consists of all subsets of a single face of K is referred to as a simplex of K.

DEFINITION 1.25 (n-Skeleton of the Simplicial Complex K) Given a simplicial complex K and a non-negative integer n, the *n-skeleton* of K, denoted $K^{(n)}$ is the set of simplices in K of dimension no greater than n. The n-skeleton of K is itself a simplicial complex and therefore a subcomplex of K.

DEFINITION 1.26 (q^{th} Chain Group of the Simplical Complex K) The *q^{th} chain group of the simplicial complex K* is the quotient group $\Delta_q(K)/\Delta_q^0(K)$, where $\Delta_q^0(K)$ is the subgroup of $\Delta_q(K)$ generated by elements of the form $(\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_q)$, where $\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_q$ are not all distinct, and by elements of the form $(\mathbf{v}_{r(0)}, \mathbf{v}_{r(1)}, \mathbf{v}_{r(2)}, ..., \mathbf{v}_{r(q)})$ where r is some permutation of $\{0, 1, 2, ..., q\}$.

DEFINITION 1.27 (q-Simplex Chain) *q-simplex chain* (or *q-simplicial chain*) is an element of the chain group $C_q(K)$, composed of a final formal sum of q-dimensional simplices from $K^{(q)}$, written as $c = \sum a_i \sigma_i$.

Under $\mathbb{Z}_2$ coefficients ($a_i = 0 \; or \; 1$), a $q-chain$ is a collection of $q-simplices$. Furthermore, we can define a binary operation $+$, over the set of $q-chains$ for a simplicial complex as follows:

$$c_0 + c_1 = \sum a_i \sigma_i + \sum b_i \sigma_i = \sum (a_i + b_i mod 2)\sigma_i$$

Under *mod2* addition, the orientation of simplices does not matter. Hence the boundary of a $q-simplex$ can be written as the addition of its $(q-1)$-faces, i.e., $\partial[v_0 v_1] = [v_0] + [v_1]$. Thus, for a given $q-chain$, $c = \sum a_i \sigma_i$, the boundary is the sum of the boundaries of its simplices, $\partial_q c = \sum a_i \partial_q \sigma_i$. Since the boundary operator commutes with addition $\partial_q(c_0 + c_1) = \partial_q c_0 + \partial_q c_1$, the map between chain groups, $\partial_q : C_q \to C_{q-1}$, is a homomorphism.

Lemma 1.

*Let K be a simplicial complex and **k** be a commutative ring. The chain complex of K with co-*

efficients in $\mathbf{k}$*, denoted* $C_\cdot(K;\mathbf{k})$ *has* $C_q(K;\mathbf{k})$ *as its* q^{th} *module and boundary operators defined by:*

$$C_q(K;\mathbf{k}) \ni [v_0, v_1, ..., v_q] \overset{\partial_q}{\longmapsto} \sum_{i=0}^{q} (-1)^i [v_0, ..., \hat{v}_i, ..., v_q] \in C_{q-1}(K;\mathbf{k})$$

The chain complex can be expressed as the sequence of chain groups connected by boundary homomorphisms.

$$... \xrightarrow{\partial_{q+2}} C_{q+1} \xrightarrow{\partial_{q+1}} C_q \xrightarrow{\partial_q} C_{q-1} \xrightarrow{\partial_{q-2}} ...$$

1.5.2 Homology

The fundamental group is a remarkably powerful tool for capturing the global structure and characterizing holes in topological spaces [98]. Unfortunately, it is difficult to determine whether two groups are isomorphic or not. Homology theory is designed to address this problem by building a sequence of abelian groups $H_q(X)$, called homology groups, from a topological space . In addition to providing a language for characterizing holes of topological spaces, homology groups can be extended to higher dimensions and computed efficiently using routines adopted from linear algebra [99].

1.5.2.1 Simplicial Cycles and Boundaries

To define homology groups we must focus on two types of simplicial chains, cycles and boundaries.

DEFINITION 1.28 (q-cycle) A q-cycle c is a q-chain whose boundary is zero, denoted $\partial_q c = 0$.

DEFINITION 1.29 (*kernel of* ∂_q) The *kernel of* ∂_q, denoted $Z_q = ker\partial_q$, is the set of all q-chains whose boundaries go to zero under the q^{th} boundary homomorphism, where Z_q represents the subgroup formed by the set of all q-cycles denoted $Z_q \subseteq C_q$. Since the chain groups are abelian, their cycle subgroups are abelian as well.

The kernel of ∂_q can be expressed as follows:

$$ker\partial_q = c \in C_q | \partial_q(c) = e_{C_{q-1}},$$

where $e_{C_{q-1}}$ denotes the identity element in C_{q-1}.

DEFINITION 1.30 (*q-boundary*) A *q-boundary* c is a q-chain that is the boundary of a $(q+1)$-chain, denoted $c = \partial_{q+1} d$, where $d \in C_{q+1}$.

DEFINITION 1.31 (*image of* ∂_{q+1}) The *image of* ∂_{q+1}, denoted $B_q = img\partial_{q+1}$, is the set of all q-boundaries under the $q+1^{th}$ boundary homorphism forming the group denoted $B_q \subseteq C_q$, where B_q is a subgroup of C_q. Since the chain groups are abelian, so are their boundary subgroups.

The image of ∂_{q+1} can be expressed as follows:

$$img\partial_{q+1} = \{c_q \in C_q | \exists_{c_{q+1} \in C_{q+1}} \partial_{q+1}(c_{q+1}) = c_q\}$$

The idea of homology derives from a chain complex of simplicial chain groups together with some map ∂ between chain groups $C_q \to C_{q-1}$. The fundamental lemma of homology intuitively states the map ∂ has the the property that $\partial\partial c = 0$ for all chains c.

Lemma 2. $\partial_q \partial_{q+1} d = 0$ *for every dimension-index* q *and* $q+1$*-simplicial chain* d

Proof:

$$\partial_{q+1}(d) = \sum_{i=0} (-1)^i d|[v_0, ..., \hat{v_i}, ..., v_{q+1}] \tag{1.26}$$

$$\partial_q \partial_{q+1}(d) = \sum_{j<i} (-1)^i (-1)^j d|[v_0, ..., \hat{v_j}, ..., \hat{v_i}, ..., v_{q+1}] + \sum_{j>i} (-1)^i (-1)^{j-1} d|[v_0, ..., \hat{v_i}, \hat{v_j}, ..., v_{p+1}]$$

Since the boundaries form subgroups of the cycle groups, the q^{th} homology group $H_q(X)$ for a chain complex can be defined as the quotient group denoted as the q^{th} cycle group modulo of the q^{th} boundary group:

$$H_q = Z_q / B_q = ker\partial_q / Im\partial_{q+1} \tag{1.27}$$

Each element is in the form a coset, obtained by adding each q-boundary to a given q-cycle, $c + B_q$ where $c \in Z_q$. Any two q-cycles,c_1, c_2, are said to be homologous $c_1 \sim c_2$ if they are elements of the same coset. The homology group is the collection of all such cosets.

A generating set H of a group G is a set of elements such that any element in G can be represented as a combination of elements in H and their inverse. If the group G is free, then there exists a basis H such that any element in G can be uniquely represented as a combination of elements in H and their inverse. The basis of a free abelian group G is also the smallest generating set for G, where the $rank$ of G, denoted $rank(G)$ represents the cardinality of any basis of G. Let card(G) denote the cardinality, or order, of the group G. Given a simplicial complex K, where the set of q-simplices form a basis for the group C_q, it can be shown that rank$(G) = n_q$, where n_q is the number of q-simpices in K. Thus, under modulo2 addition, card$(C_q) = 2^{n_q}$. The number of cycles in a coset is the cardinality of B_q, hence the number of cosets in the homology group is computed as follows:

$$\text{card}(H_q) = \text{card}(Z_q)/\text{card}(B_q). \tag{1.28}$$

Equivalently the rank of H_q, also known as the q^{th} Betti number and denoted β_q , is computed as the difference:

$$\beta_q = \text{rank}H_q = \text{rank}Z_q - \text{rank}B_q \tag{1.29}$$

1.5.3 Gröbner Bases

The method of Gröbner bases has become one of the most important techniques in providing an exact solution to nonlinear problems in multivariate polynomial ideal theory, computational algebra, elimination theory, and in solving systems of algberaic equations [99–101]. Our brief treatment of algebraic geometry follows [98, 102]. Let $R = k[x_1...x_r]$ denote the polynomial ring in r indeterminates over a field k and x^a as the shorthand for the monomial $x_1^{a_1}...x_n^{a_n}$. Unless otherwise stated, we will call a module a R-module and vector space a k-vector space. Let R^N be a finitely generated free module with canonical basis $e_1, ..., e_N$. A monomial in R^N is an element of the form $m = x^a e_i$ for some i, and a term is an element of the form $cx^a e_i$ for $c \in k$. Each element $f \in R^N$ is a linear combination of monomials $x^u e_i$. The monomial submodule generated by elements $f_1, ..., f_t$ in R^N will be denoted by $< f_1, ..., f_t >$. A monomial order on R^N is a total

order $>$ on the monomials of R^N. Given a monomial order $>$ on R^N, and $f \in R^N$, we will define the leading monomial $LM(f)$ and leading coefficient $LC(f)$ as the greatest monomial of f and its leading coefficient, respectively. If M is a finitely generated submodule of R^N, then $LM(M)$ is the submodule of R^N generated by the leading monomials of the elements of M. A finite set of generators $\{f_1, ..., f_t\}$ for a module $M \subset R^N$ is a Gröbner basis of M if

$$LM(< f_1, ..., f_t >) =< LM(f_1), ..., LM(f_t) >$$

It follows that a Gröbner basis for an ideal I in R is a set of generators for I with an additional property that allows for an efficient solution to the ideal membership problem. With the use of Gröbner bases, many intractable problems involving ideals in polynomials rings can be reduced to easy computations expressed in terms of monomial ideals. Computing Gröbner bases and finding primary decomposition of polynomial ideals are two closely related topics that are fundamental in computational algebraic geometry [99]. A Gröbner basis of a module $M \subset R^N$ can be computed from any finite set of generators using the Buchberger algorithm [103].

1.6 Sequence Learning

The aim in sequence learning is to build a model with which we can predict the next element in a data sequence, given an already observed element of the same sequence. Time-series data is a particular manifestation of sequence data that further extends the challenges already associated with sequence learning [104]. If a random variable X is indexed to time, usually denoted by t, the observations $\{X_T, t \in \mathbb{T}\}$ are called a time series, where $\mathbb{T}$ is a time index set (for example, $\mathbb{T} = \mathbb{Z}$, the integer set). In many statistical applications, it can be assumed that data is independent and identically distributed (iid) [62, 105], i.e., given a sequence of data $D = \{x_1, x_2, x_3, ..., x_N\}$ sampled from the random variable X, the likelihood can be written as the product of the individual samples

$$p(D|M) = \prod_{n=1}^{N} p(x_n). \tag{1.30}$$

However, due to the temporally correlated nature of observations in time-series data the iid assumption becomes unrealistic [37]. Instead, causal dependence among observations is often assumed, such that

$$p(D|M) = p(x_N, x_{N-1}, .., x_2, x_1) \tag{1.31}$$

$$= p(x_N|x_{N-1}, .., x_2, x_1)p(x_{N-1}|x_{N-2}, .., x_2, x_1)...p(x_2|x_1)p(x_1) \tag{1.32}$$

$$= p(x_1) \prod_{n=2}^{N} p(x_n|x_{1:n-1}) \tag{1.33}$$

In order to make inference tractable, stochastic models called Markov models utilizing weaker assumptions about the sequence data can be constructed for the purpose of predictive modeling [62, 105]. One such stochastic model is known as the Markov process.

DEFINITION 1.32 A stochastic process $\{X_t\}_{t\in T}$ with state space S on a probability space (Ω, F, P) is a *Markov process*, if for any set of $n+1$ values $t_1 < t_2 < t_3, ... < t_n < t_{n+1},$, $t_i \in T$, $i = 1, 2, 3, ..., n+1$, and any set of states $\{x_1, x_2, ..., x_{n+1}\} \subset S$, the conditional probability equals

$$\begin{aligned} &P(X(t_{n+1}) = x_{n+1}|X(t_1) = x_1, X(t_2) = x_2, X(t_3) = x_3, ..., X(t_n) = x_n) \\ =&P(X(t_{n+1}) = x_{n+1}|X(t_n) = x_n) \end{aligned} \tag{1.34}$$

In general, Markov processes can be classified into one of four types based on a system's state space and time parameter index: discrete-time Markov processes, discrete-time Markov chains, continuous-time Markov processes, and continuous-time Markov chains. Given the discrete time index set $\mathbb{T}$, the discrete-time Markov process can be used to define an approximation of the data sequence:

$$p(x_n|x_{n-1}, ..., x_2, x_1) \approx p(x_n|x_{n-1}). \tag{1.35}$$

This approximation is called a first-order discrete-time Markov process because, given the previous observation x_{n-1}, the observation x_n is conditionally independent of all previous observations. As a result, (1.32) can be written as the product of conditional probabilities on the previous observation

$$P(D|M) = p(x_1) \prod_{n=2}^{N} p(x_n|x_{n-1}). \tag{1.36}$$

To capture longer-range temporal dependencies, higher order Markov models can be constructed, such that, an observation x_n is independent of all other previous observations, given the previous M observations:

$$P(D|M) = p(x_n|x_{n-1}, ..., x_2, x_1) = p(x_n|x_{n-1}, ..., x_{n-M}) \tag{1.37}$$

In cases where the observed variable X is discretized, such that the sequence of observations $D = \{x_1, x_2, x_3, ..., x_N\}$ can be described using a discrete data sequence with each variable x_i taking on one of K states $\{s_1, s_2, ..., s_K\}$, a discrete-time Markov chain with finite space $S = \{s_1, s_2, ..., s_K\}$ can be used instead[1].

DEFINITION 1.33 A Markov process $\{X_t\}_{t\in T}$ with state space S on a probability space (Ω, F, P) is called a *Markov chain*, if the state space S is finite or countable.

Consequently, the likelihood of this discretized data sequence can be written:

$$p(D|M) = p(x_1 = s_i) \prod_{n=2}^{N} p(x_n = s_j|x_{n-1} = s_i). \tag{1.38}$$

Each conditional probability $p(x_n = s_j|x_{n-1} = s_i)$ is referred to as a transition probability and describes the probability of being in state s_j at time $n + 1$, given that the state at time n is s_i:

$$\pi_{i,j} = p(x_n = s_j|x_{n-1} = s_i). \tag{1.39}$$

In many cases it is assumed that the transition probabilities are homogeneous (i.e do not change over time), thus

$$p(x_n = s_j|x_{n-1} = s_i) = p(x_{n+T} = s_j|x_{n-1+T} = s_i), \tag{1.40}$$

where $T \geq 1$. The probabilities $\pi_{i,j}$ are usually presented in the form of a matrix π called the transition matrix:

$$\pi = (\pi_{i,j})_{i=1,j=1}^{K,K} \tag{1.41}$$

[1]When the observed variable is discretized via quantization, each state s_i corresponds to one of K computed quantization levels.

$$\pi = \begin{pmatrix} \pi_{1,1} & \pi_{1,2} & \dots & \pi_{1,K} \\ \pi_{2,1} & \pi_{2,2} & \dots & \pi_{2,K} \\ \vdots & \vdots & \ddots & \vdots \\ \pi_{K,1} & \pi_{K,2} & \dots & \pi_{K,K} \end{pmatrix}$$

Here, each row i of the transition matrix π represents the conditional probability distribution of X_n given that state $X_{n-1} = s_i$. In addition, the transition matrix assumes that $\pi_{i,j} \geq 0$ and $\sum_{j=1}^{K} \pi_{i,j} = 1$, for all $i, j = 1, ..., K$.

In addition to Markov Processes, the Hidden Markov Model (HMM) is another popular Markov model that is commonly used to tractably model time-series and sequence data [106]. The standard HMM model utilizes a state space model in which each observation x_i has a corresponding discrete latent (unobserved) variable called the hidden state z_i that it is associated with. These hidden states $z_1, z_2, z_3, ..., z_n$ emit the sequence of observations $\{x_1, x_2, x_3, ..., x_n\}$ based on the conditional distributions of the form $p(x|z)$. Similar to a Markov chain modeling a discrete data sequence, an HMM is defined by the number of states K, a $K \cdot K$ transition probability matrix π, and a parameterized distribution[2] $F(\cdot)$ describing the conditional probabilities $p(x_i|z_i)$. The generative model describing an HMM is as follows:

$$z_i \sim \pi_{z_{i-1}} \tag{1.42}$$

$$x_i \sim F(\theta_{z_i}), \tag{1.43}$$

where θ_{z_i} is a unique set of parameters associated with z_i. In general, an HMM can be used to describe a data sequence via a mixture model in which the indices of mixture components have a temporal dependency that can modeled as a first-order Markov chain. Unfortunately in standard Hidden Markov models, in which an observation x_i is considered conditionally independent of any other observation x_j, given the hidden state z_i, the possibility that both the observations and the hidden states possess temporal dependencies is ignored [106, 107]. Although many extensions to

[2] Although each hidden state z can have its own corresponding parameterized distribution $F_z(\cdot)$, in many cases it is more common for all hidden state emission distributions to share a common form, e.g. Gaussian distribution. Nevertheless, a unique set of distribution parameters θ_z is associated with each hidden state z.

the standard HMM method exist and address the aforementioned issue in addition to others, these methods still require considerable domain expertise in order to restrict possible model architectures [106]. In addition to this problem of model selection-inferring the optimal model size (number of hidden states)-HMM training typically requires very large training samples [107].

Learning from sequence data is critical when trying to model and predict the dynamics of biological phenomena [2, 108]. Since the function and behavior of many complex biological systems and processes may evolve over time, it is also necessary for predictive models to be continuously updated as data becomes available. This process of continuous model adaptation based on a constant input datastream and limited memory resources is referred to as *incremental learning*. A datastream can be defined as an unbounded sequence of data elements that is indexed on the basis of when each data element is observed. As pointed out by [109], a datastream can be characterized by the following fundamental properties:

1) data elements arrive in a continuous manner, sometimes at different rates;
2) datastreams are potentially unbounded in size;
3) following processing, each data element cannot be retrieved unless it is explictly stored in memory.

Thus, when the data is in the form of a datastream $\vec{x} = [x_1, x_2, ..., x_{t-1}]$, and not available *a priori* to training, incremental learning approaches infer a model M_t after every time step t based on the current data element and the previously learned model (x_t, M_{t-1}). This incremental processing of a datastream, one element at a time, is commonly referred to as *online* learning. Although several sequence learning approaches have been proposed over the years, only few have successfully demonstrated the ability to learn incrementally from datastreams without expert supervision [110–113].

CHAPTER 2

Simplicial Grammar

It has long been recognized that abstractions and approximations are critical to reasoning and problem solving in humans as well as in AI frameworks where it is used to overcome computational intractability by decreasing the combinatorial costs associated with searching large spaces [114–116]. In addition, abstractions and approximations are also useful for acquiring knowledge and generating explanations [117, 118]. Within Artificial Intelligence, the ability to explain reasoning processes and results can have a substantial impact on user confidence. Specifically, by providing evidence of how a result was derived serves to increase user acceptance and guide learning [119, 120]. In this chapter, we introduce the use of topological structures as the primary data abstraction in a new form of probabilistic generative models called simplicial grammar. These grammars derive their name from the simplicial complexes used to characterize a collection of pattern primitives. Simplicial grammars are a nonparametric generalization of graph grammars, designed to articulate the multi-level interdependencies inherent to a complex system via the use of a visually intuitive representation. A simplicial grammar is an efficient and flexible data structure for representing multi-dimensional, complex pattern grammars in terms of a well-defined mathematical formalism that is highly expressive and an ideal extension to traditional probabilistic generative models.

2.0.1 Description of Modeling Formalism

We combine simplicial complexes and stochastic grammars in a new modeling formalism that can be computationally manipulated for the purpose of extracting, analyzing and understanding key patterns and features of multi-level complex systems. A *Simplicial Grammar* is a Bayesian

nonparametric modeling formalism for discrete sequences, which extends traditional graph grammar in order to deal with combinatorial structures. The basic idea is that the state of many complex systems can be naturally represented (at a suitable level of abstraction) as a simplicial complex, and (local) transformations of the state can be expressed as production rule applications.

Let K denote an abstract simplicial complex: $K = (V, S)$ where V is a finite set whose elements are called the vertices of K and S is a set of non-empty subsets of V that is required to satisfy the following two conditions:

1) $v \in V \implies \{v\} \in K$

2) $\sigma \in K, \tau \subset \sigma \implies \tau \in K$

Recall, each element $\sigma \in K$ is called a simplex of K and is a subset of the vertices from the vertex set $\sigma = \{v_0, ..., v_d\} \subset V$. If $\sigma \in K$ has precisely $d + 1$ elements, $d \geq -1$, σ is called a d-dimensional simplex. The face of such a simplex is also a simplex but whose vertices form a subset of $\{v_0, ...v_d\}$. The dimension of the simplicial complex K is the largest dimension d of any simplex, $dim(K) = max\{dim(\sigma)|\sigma \in K\}$. The set of all d-simplices of K is denoted by $K^{(d)}$. Thus, the vertex set can also be regarded as the set of 0-simplices, $V = K^{(0)}$. A simplicial grammar allows a user to finitely describe a (possibly infinite) collection of simplicial complexes, i.e., those complexes which can be constructed from an initial simplicial complex through repeated applications of simplicial production rules. A *simplicial production rule* is, a rule of the kind

$$rule_i^{(d)} : (K_C; K_R),$$

where K_C and K_R are simplicial complexes called the *context* and *replacement* complexes, respectively. Such a rule is *applicable* to a *host* simplicial complex K_H whenever there is an occurence (match) of the K_C in K_H. The *application* of this rule on K_H yields a new complex $K_H^{'}$ by removing the occurence of K_C from K_H and replacing it with an isomorphic[1] copy of K_R. We write $K_H \xrightarrow{rule_i^{(d)}} K_H^{'}$ to denote that $K_H^{'}$ was obtained from K_H by the application of $rule_i^{(d)}$.

Thus, the main component of a simplicial grammar is its finite set of simplicial productions. The form, the notion of match and the mechanisms stating how a production can be applied to a

[1]Two abstract simplicial complexes K_i, K_j, are isomorphic if there is a bijection $b : V_{K_i} \to V_{K_j}$, such that $\sigma \in K_i$ iff $b(\sigma) \in K_j$, where V_{K_*} denotes the vertex set of a complex K_*

simplicial complex, depend on the combinatorial description of the geometric notion of a simplicial complex. Any d-dimensional simplicial complex specified by the replacement-complex K_R is constructed by a unique sequence of operations (*insertion, deletion, replacement, etc*) performed on the context-complex K_C of K_H. It should be noted that the superscript $^{(d)}$ of a simplicial production rule, called the *operating dimension*, indicates the maximal dimension at which operations are performed. For example, in Figure 2.1, operations are performed at the level of the 0-simplices from $K_H^{(0)}$ denoted by the set of singletons $= \{\{a\}, \{b\}, \{c\}, \{d\}, \{e\}, \{f\}, ...\}$. In addition, since rules of maximal dimension $d > 0$ consist of operations using simplices with boundary faces, applying such rules on a host complex also involves the recursive application of lower-level simplical grammar production rules

$$rule^{(d-j)} : (K_{C_*}, K_{R_*}), \text{ for } j = 0, 1, 2, ..d \text{ and } K_{C_*} \subset K_C,$$

whose respective context-complexes K_{C_*} intersect with a common face (simplex) or sequence of faces (simplex chain) of K_C. This inherent mapping between simplicial dimensions has the added benefit of allowing for the decomposition of a model into sub-models of desired granularity. An example illustrating the recursive application of lower-level rules involved in a complex simplicial rule assembly is given in Figure 2.2.

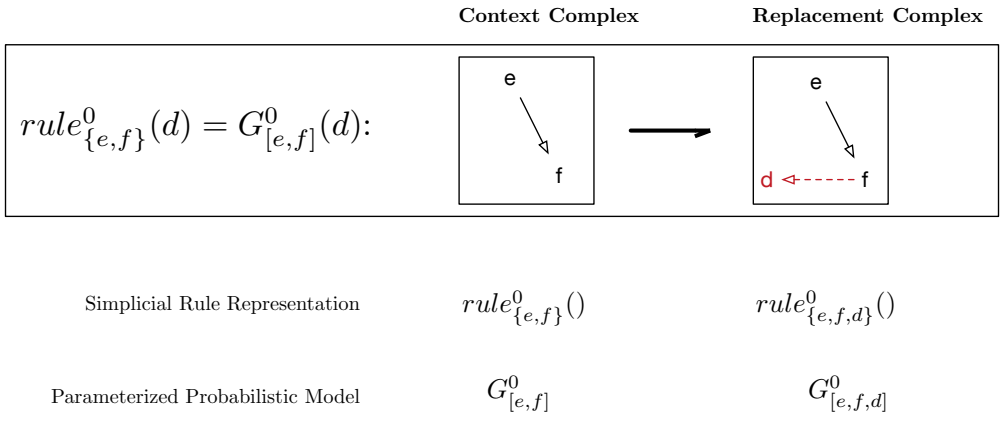


FIGURE 2.1 0-dimensional Simplicial Production Rule

When modeling spatio-temporal structures, the application of simplicial grammar rules explictly models both the sequential order of operations and the temporal dependencies inherent to the sequence/time-series data along with their associated probabilties. In such cases, each simplicial

grammar rule of operating dimension $d \geq 0$, when instantiated, implicitly models the probability of a d-simplex chain $\mathbf{q}^d_{1:T} = (\sigma^d_1, \sigma^d_2, ..., \sigma^d_T) \in C_d$, for $d \geq 0$, as the product of conditional probabilities :

$$P(\mathbf{q}^d_{1:T}) = \prod_{i=1}^{T} P(\sigma^d_i | \sigma^d_{i-1}) \tag{2.1}$$

The key assumption underlying a simplicial grammar and each of its simplicial rewrite-rules, is the existence of a set of random variables drawn from some unknown probability distribution. This unknown probability distribution is itself drawn from some prior distribution. To allow uncertainty in distributional assumptions and to avoid critical dependence on parametric assumptions, each simplicial rewrite-rule is based on a Hierarchical Pitman-Yor process prior. Thus, the conditional probabilities in (2.1) can be parameterized using random probability measures $G_{\mathbf{q}}$:

$$P(\mathbf{q}^d_{1:T}) = \prod_{i=1}^{T} P(\sigma^d_i | \sigma^d_{i-1}) = \prod_{i=1}^{T} G_{\mathbf{q}^d_{1:i-1}}(\sigma^d_i) \tag{2.2}$$

Each random probability measure, $G^d_{\mathbf{q}}(\sigma)$, models the probability of observing a d-simplex $\sigma \in K^{(d)}$, conditioned on a chain of previously observed simplices $\mathbf{q} \in C_d$, given a Hierarchical Pitman-Yor Process prior:

$$G_{\emptyset} \sim PYP(\alpha_{\emptyset}, d_{\emptyset}, H)$$
$$G_{\mathbf{q}} | G_{\pi(\mathbf{q})} \sim PYP(\alpha_{\mathbf{q}}, d_{\mathbf{q}}, G_{\pi(\mathbf{q})}) \;\; \forall \, \mathbf{q} \in C_d \backslash \{\emptyset\}$$

where $\emptyset$ denotes the empty chain and $\pi(\mathbf{q})$ represents a variable-length truncation of chain $\mathbf{q}$. Computing the posterior distribution of a simplicial grammar provides a partition of the data into subgrammars, without requiring that the number of subgrammars be pre-specified in advance.

2.0.2 Making Complex Data Analysis More Tractable and Intuitive

The purpose of developing a modeling formalism based on simplicial complexes and stochastic grammars is to enable users to learn and synthesize knowledge from massive, multi-dimensional, dynamic, heterogeneous, noisy datasets in a form that is tractable yet intuitive. In the simplicial grammar modeling formalism users can incorporate methods from the fields of geometry and topol-

ogy for the purpose of creating an approximation of the topological space from which a finite set of datapoints (noisy observations) may have been sampled. This approximation provides a platform by which structural features and topological invariants inherent to an empirical dataset can be extracted, analyzed, searched and stored in-memory for purposes of *dimensionality reduction* and *quantization*. Recall, a simplicial grammar is a collection of stochastic production rules, each describing a sequence of operations required for the construction of a replacement-complex from an initial context-complex. Since the instantiation and application of a rule to a host complex takes on the form of an abstract simplicial complex, topological invariants such as homology groups can be computed from these complexes via methods developed by the field of Computational Topology. These invariants can reveal topological attributes not inferred using conventional network-theory methods, and thus provide an alternative method for discriminating features within large datasets across multiple scales [95, 97, 121].

Recognition of the importance of simplicial complexes and their combinatorial and topological properties can be dated back to the seminal works of Euler and Riemann as well as the relatively recent contribution of homology classes by Poincaré [122]. The topological property known as homology offers a general procedure by which a sequence of abelian groups or modules can be associated to a given topological space or manifold. Determination of the different dimensional homology groups provides information about the topological invariant characteristics of a system which may be subsequently used for recognition, classification and prediction purposes. The homology group, denoted $H_d(X)$, pertaining to a given topological space X and dimension d, provides a global description of the d-simplicial chains. Given some simplicial complex K, a d-dimensional simplical-chain, or d-chain, in K is a finite formal sum of d-simplices, formally expressed as $q = \sum_{i=1}^{k} \alpha_i \sigma_i$, where σ_i are the d-simplices and α_i are the coefficients from the field $\mathbb{Z}_2$. It follows from **Lemma 3**, that under the binary addition operator, a set of d-chains form a group called the d-th chain group.

Lemma 3. *Let K be a simplicial complex and C_d the set of d-chains in K. The set C_d with the operator $+$ form a group, denoted $(C_d, +)$.*

Proof: The identity is the chain $0 = \sum_{i=1}^{k} 0\sigma_i$, and the inverse of a chain, $-q = q$ since $q + q = 0$ under $\mathbb{Z}_2$ additions. The set of oriented d-simplices in K, $\{e_1, e_2, ..., e_{n_d}\}$, define a basis for C_d

At different dimensions, these chain groups are related by a boundary operator, ∂_d, that, given a d-simplex, returns the $(d-1)$-chain of its boundary $(d-1)$-simplices. Thus, if $\sigma = [v_0, ..., v_d]$ denotes a d-simplex, its boundary is $\partial_d \sigma = \sum_{i=0}^{d} (-1)^i [v_0, ..., \hat{v_i}, ..., v_d]$. Furthermore, because the boundary operator commutes with addition, $\partial_d(q_1 + q_2) = \partial_d q_1 + \partial_d q_2$, if extended to chain groups, the map $\partial_d : C_d \rightarrow C_{d-1}$ becomes a homomorphism. A sequence of chain groups connected by these boundary homorphisms is called a chain complex $\hat{C}$. To define the homology groups of dimension d, we must focus on the two simplicial-chain subtypes, d-cycles and d-boundaries. A d-dimensional simplicial cycle or d-cycle, z, is a d-chain whose boundary is zero, $\partial_d z = 0$. The set of all d-cycles form a group denoted $Z_d \subseteq C_d$, where Z_d is a subroup of C_d. Since Z_d is the set of all d-chains that go to zero under the dth boundary homomorphism, Z_d is the kernel of ∂_d denoted $Z_d = ker \partial_d$. Furthemore, a d-dimensional simplicial-boundary or d-boundary is a d-chain that is the boundary of a $(d+1)$-chain, $z = \partial_{d+1} q$ for $q \in C_{d+1}$. The set of all d-boundaries form a group denoted $B_d \subseteq C_d$, where B_d is a subgroup of C_d. The group of d-boundaries is the image of the $(d+1)$-st boundary homomorphism, $B_d = img \partial_{d+1}$. From the fundamental lemma of homology **(Lemma 2)**, which intuitively states that the boundary of a boundary is null, it follows that B_d is a subgroup of Z_d. From this we can define an equivalence relation over Z_d. Two d-cycles z_1 and z_2 are considered homologous if the d-cycle $z_1 - z_2$ is a d-boundary. The equivalence class of a d-cycle z_1 is the homology class $[z_1]$. Addition of homology classes is well-defined; for any d-cycles z_1 and z_2, we have $[z_1 + z_2] = [z_1] + [z_2]$. Thus, the set of homology classes of d-cycles forms a well-defined group under addition, called the dth homology group, H_d. By taking the quotient of the cycle groups with the boundary groups, we can define the homology groups for each dimension d, $H_d = \frac{Z_d}{B_d}$. Thus, each homology group is the collection of d-cycles that are not boundaries of $(d+1)$-simplices. The rank of the d-th homology group H_d is called the d-th Betti number β_d and informally describes the number of unconnected d-dimensional surfaces.

In addition to allowing users to identify distinct features of a topological space such as a torus, sphere, or annulus, these homology groups can be used for dimensionality reduction, as well as for quick visual inspection and recognition of discrepancies indicative of distinct classes of patterns that are only evident after scrutinizing the connectivity and shape of a given dataset.

2.0.2.1 Dimensionality Reduction

The use of geometry and topology for the purpose of analyzing high-dimensonal data has garnered significant attention in recent years due to continued advancement in technology and the resultant abundance of generated data [95]. The primary application of methods from these fields has been in reducing data size while minimizing information loss [123]. This process, also known dimensionality reduction, focuses on reducing the dimensionality of a given dataset and extracting a low-dimensional embedding while preserving as much structural information as possible. In practice, the structural property being preserved is often expressed as the pairwise distances or topology between objects. The application of dimensionality reduction techniques enables visualization of high-dimensional data and solves a fundamental problem in many data analysis tasks–estimating the appropriate number of dimensions required for representing a given dataset without considerable loss of information. Dimensionality reduction methods can be divided into two groups, linear and non-linear methods. Popular linear methods include Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), Multi-Dimensional Scaling (MDS), Factor Analysis (FA), and Linear Regression among many others, which transform original variables into new variables using a linear combination of the original variables. However, the assumption of linear relationships among variables is not strictly valid. Often data lies on a non-linear manifold/surface, in which case non-linear methods such as Locally Linear Embedding (LLE), and Isomap are often used instead. Although these non-linear techniques often make fewer hypotheses about the underlying manifold/surface and/or model, they require the use of optimization procedures, evaluation criteria and the definition of application-specific objective criteria [124].

Recent progress in the field of Computational Topology [95, 121, 125] has provided additional tools for dimensionality reduction by using homology to measure topological features of a surface or manifold from discrete data sets. One approach in particular, known as persistent homology, has seen recent widespread use in scientific and engineering applications [97]. This approach allows for the topological simplification of combinatorial data by measuring the life-time of intrinsic topological features via a filtration process that incrementally builds simpicial complexes of increasing complexity and structural stability through a sequence of nested subcomplexes. As a result, each

preceding subcomplex is contained within the simplicial complex generated in subsequent filtration stages. Consider the following sequence describing the filtration of a simplicial complex K

$$K_0 \subset K_1 \subset ... \subset K_n = K$$

This increasing sequence can be understood as a process in which new simplices are incrementally added following changes to some free parameter. This filtration parameter can be used to capture additonal quantitative information about the topological invariants at multiple geometric scales such as the birth and death of individual connected components (dimension 0), tunnels (dimension 1) and voids (dimension 2).

2.0.2.2 Homology Group Computation

In order to compute homology modules from simplicial grammar, we have implemented an extension to the Buchberger and Schreyer algorithms [103, 126] based on similar work in the area of Topological Data Analysis [127]. This extension recasts homology analysis as a problem within computational algebraic geometry, and allows us to utilize powerful algorithms from this area for computing Gröbner bases for ideals and the related szygy modules in simplicial grammars. Specifically, for a given dimension q, we compute:

1) the boundary module $im\partial_{q+1}$
2) the cycle module $ker\partial_q$
3) the quotient H_q

The boundary module is obtained by computing a Gröbner basis via the Buchberger algorithm, given in **Algorithm 1** , and then performing the submodule membership problem via the division algorithm for multivariate polynomials. The cycle module is subsequently obtained via Schreyer's algorithm by computing the syzygy submodule. Finally, the quotient is computed by determining whether the generators of the syzygy submodule are in the boundary submodule (submodule membership problem).

Let, C_q^* denote the set of q-simplex chains representative of the context complexes of simplicial production rules in a simplicial grammar **SG**. Assuming the field is $\mathbb{Z}_2$, the chains of C_q^*

Algorithm 1 Buchberger algorithm pseudocode

procedure $Buchberger(f_1, ..., f_m)$
    $G = \{f_1, ..., f_m\}$,
    $P = \{Spoly(f_i, f_j) \neq 0 | 0 \leq i < j \leq s\}$
    for each $p \in P$ **do**
        $P = P - \{p\}$
        if $h = Reduce(p, G) \neq 0$ **then**
            $G = G \cup \{h\}$,
            $P = P \cup \{Spoly(g, h) \neq 0 | g \in G\}$
        end if
    end forreturn G
end procedure

procedure $Spoly(f_1, f_2)$
    $c = LC(f_1)/LC(f_2)$,
    $s_{ij} = \frac{lcm(LM(f_i), LM(f_j))}{LM(f_j)}$
    return $s_{21} f_1 - c s_{12} f_2$
end procedure

procedure $Reduce(f, \{g_1, ..., g_t\})$
    while there exists a g_i such that $LM(g_i)$ divides $LM(f)$ **do**
        $c = \frac{LC(f)LM(f)}{LC(g_i)LM(g_i)}$,
        $f = f - c g_i$
    end while
    return f
end procedure

are in one-to-one correspondence with the set of subsets of q-simplices. A q-simplex chain corresponds to a n_q-dimensional vector, whose nonzero entries correspond to the included q-simplices. Here n_q is the number of q-simplices in **SG**. We can compute the boundary of a q-chain by multiplying the chain vector with a boundary matrix, whose column vectors are boundaries of q-simplices in the **SG**. Since the boundary maps ∂_q are homomorphism between free abelian groups $\partial_q : C_q(\mathbf{SG}) \rightarrow C_{q-1}(\mathbf{SG})$, they can be expressed as integer-valued matrices M_q. Row reduction of M_q allows for the identification of a basis $v_1, ..., v_n$ for $Z_q(\mathbf{SG})$. As a consequence of $M_q M_{q+1} = 0$, the columns of M_{q+1} are in the kernel of M_q, and can thus be expressed as linear combinations of the v_i. These linear combinations are determined via row reduction of the augmented matrix $(v_1 ... v_n | M_{q+1})$. Thus, the computation of the Betti number can be rewritten as:

$$\beta_q = (n_q - rank(M_q)) - rank(M_{q+1}) \tag{2.3}$$

Given the simplicial complex K in Figure 2.3 containing (5) 0-simplices, (8) 1-simplices and (2) 2-simplices along with the boundary matrices in Figure 2.4, it can be shown that this complex contains one nontrivial homology class, represented by three different non-bounding cycles, $(ab + bc + cd + da)$,$(ab + bc + ca)$, $(ab + bc + ce + ea)$, whose corresponding vectors are $(1, 0, 1, 0, 1, 1, 0)^T$, $(1, 1, 0, 0, 1, 0, 0)^T$ and $(1, 0, 0, 1, 1, 0, 1)^T$ respectively. Simarily, for a set of q-simplicial production rules with identical replacement-complexes (Figure 2.5), we would seek to compute the appropriate homology groups and their representative non-bounding cycles. These invariants are to serve as a tool for identifying the latent higher-dimesional pattern primitives, whose boundary-cycles can be described by pre-existing simplicial production rules. These higher-dimensional pattern primitives will be used to guide the incremental inference of new grammars based on data obtained from new scales of observation. The basic aim is exploit the computation of homology groups and other topological invariants in studying the relationship/mapping between local and global structures for simplicial grammars. It is important to note that the selection and extraction of the finite set (or alphabet) of distinct pattern-primitives that underly complex patterns and the construction of simplicial grammar is a nontrivial challenge. We address this fundamental problem by building on the theory and methods of vector quantization and k-means clustering for

the purpose of identifying the initial set of pattern-primitives and subsequent decomposition of any complex data sequence into a topological representation.

2.0.2.3 Data Quantization

Although measurements of many real-world phenomenon are continuous, it is often desirable to represent data as discrete variables. Quantization (or discretization) is the process by which a continuous variable is converted into a discrete variable. In such cases, the discretized variable has a finite number of possible values, a number that is considerably smaller than the number of possible values found in the empirical dataset. In addition to improving the representation, interpretation, and accessibility of data, continuous feature quantization also offers to increase the speed of induction algorithms [128]. This benefit becomes more apparent in subsequent sections of Chapter 3 detailing the proposed method of grammatical induction for simplicial grammar

Given an arbitrary k-dimensional dataset of N observed datapoints for which we do not know the probability distribution function, we seek to find L Voronoi regions representative of 0-simplices (pattern primitives) in the k-dimensional Euclidean space, where $N >> L$, by minimizing the total squared error. An iterative splitting implementation of the LBG algorithm proposed by Linde, Buzo, and Gray in [129], is used for the purpose of generating a discrete probability distribution defined over a set of pattern-primitives (finite alphabet), that could subsequently be used as the base measure of a Simplicial Grammar's Hierarchical Pitman-Yor Process prior. The implementation of this algorithm which we will call **LBGSplit**, starts with a base measure of size 1, where the only simplex corresponds to the centroid of the training set. The base measure is iteratively enlarged until the number of simplices reaches size L. The sketch of the **LBGSplit** algorithm is given in **Algorithm 2**.

The result of this algorithm is a set of Voronoi simplices $\{Y_i^*\}_{i=1}^{l}$ describing the centroids of k-dimensional partitions (Voronoi regions),$\{V_i^{(l)}\}$, of $\mathbb{R}^k$, i.e., $\cup_{i=1}^{l} V_i = \mathbb{R}^k$. Put simply, the Voronoi region V_i of the Voronoi simplex Y_i, is the partition such that any observed point in V_i is closer to Voronoi simplex Y_i than any other Y_j.

$$V_i = \{X = (x_1, ..., x_k) \in \mathbb{R}^k | d(Y_i, X) < d(Y_j, X) \forall j \neq i\}$$

Algorithm 2 LBGSplit Pseudocode

Initial number of Voronoi simplices $l = 1$;
Maximum iterations = M;
Threshold for change in distortion = ϵ;

procedure $LBGSplit(l)$
  while $l \leq L$ **do**
    $\{Y_i^*\}_{i=1}^{l}$ ▷ Initialize Voronoi simplices
    $V_i^* = X : d(X, Y_j^*) < d(X, Y_i^*) \exists j \neq i$ ▷ Calculate Voronoi regions for training set
    $D_l \leftarrow MinimizeDistortion()$
    $D_{2l} \leftarrow LBGSplit(l \leftarrow 2 * l)$ ▷ Split each Voronoi region into 2 new subregions
    if $|D_{2l} - D_l| < \epsilon$ **then**
      return D_l
    end if
  end while
end procedure

procedure $MinimizeDistortion()$
  $m = 0$;
  $D^0 = \sum_{i=1}^{l} \sum_{X \in V_i^0} (X - Y_i^0)^T (X - Y_i^0)$ ▷ Compute initial distortion
  for ($m <$ M) **do**
    $Y_i^{(m+1)} = \frac{1}{|V_i^{(m)}|} \sum_{X \in V_i^{(m)}} X$ ▷ Compute centroid of Voronoi simplex
    $D^{(m+1)} = \sum_{i=1}^{l} \sum_{X \in V_i^{(m+1)}} (X - Y_i^{(m+1)})^T (X - Y_i^{(m+1)})$ ▷ Compute new distortion
    if $|D^{m+1} - D^m| < \epsilon$ **then**
      return D^m
    end if
  end for
end procedure

As previously noted, the primary goal of the algorithm is to reduce error (distortion) by means of minimizing the sum of squared distances between each observed datapoint and its closest Voronoi simplex. At each iteration of the algorithm, the set of Voronoi simplices is refined to minimize distortion. As a result, the change in distortion is used as a stopping criterion.

To further exploit the inherent structure and network of interdependencies among the primitive elements that form complex patterns, we use the finite set of Voronoi 0-simplices, Σ, obtained via the aforementioned **LBGSplit** algorithm to perform quantization on the continuous-valued input variables gathered empirically. Recall that a simplicial complex K is a collection of simplices and vertices V. Given, $V = K^{(0)} = \Sigma \subset C_0(K)$, for each vertex $v \in V$, we associate a distinct 0-simplex from the alphabet set Σ, where $K^{(0)}$ denotes the set of all 0-simplices and $C_0(K)$ the 0-th chain group of the simplicial complex. Since the set of Voronoi simplices used for quantization correspond to a finite alphabet of pattern-primitives, each quantized sequence of discrete-valued variables in our empirical data set can thus be regarded as a chain of pattern-primtives, or a 0-simplex-chain c of length $|c|$.

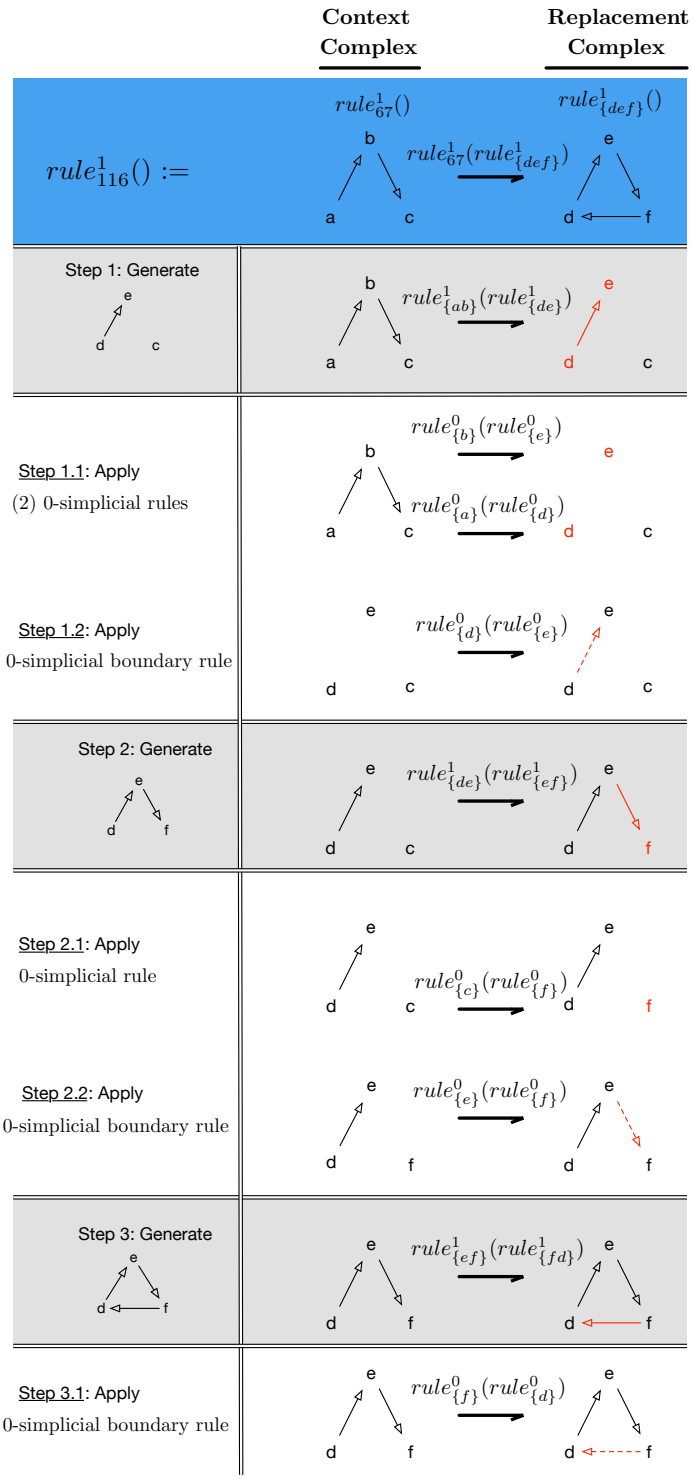


FIGURE 2.2 Complex Simplicial Rule Assembly

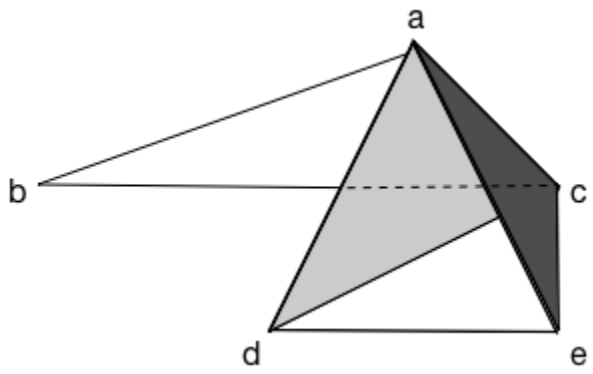


FIGURE 2.3 Simplicial Complex K

$$M_1 = \begin{array}{c} \\ a \\ b \\ c \\ d \\ e \end{array} \begin{array}{c} \begin{array}{ccccccc} ab & ac & ad & ae & bc & cd & ce \end{array} \\ \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{pmatrix} \end{array} \qquad M_2 = \begin{array}{c} \\ ab \\ ac \\ ad \\ ae \\ bc \\ cd \\ ce \end{array} \begin{array}{c} \begin{array}{cc} acd & ace \end{array} \\ \begin{pmatrix} 0 & 0 \\ 1 & 1 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} \end{array}$$

FIGURE 2.4 Boundary matrices of simplicial complex K

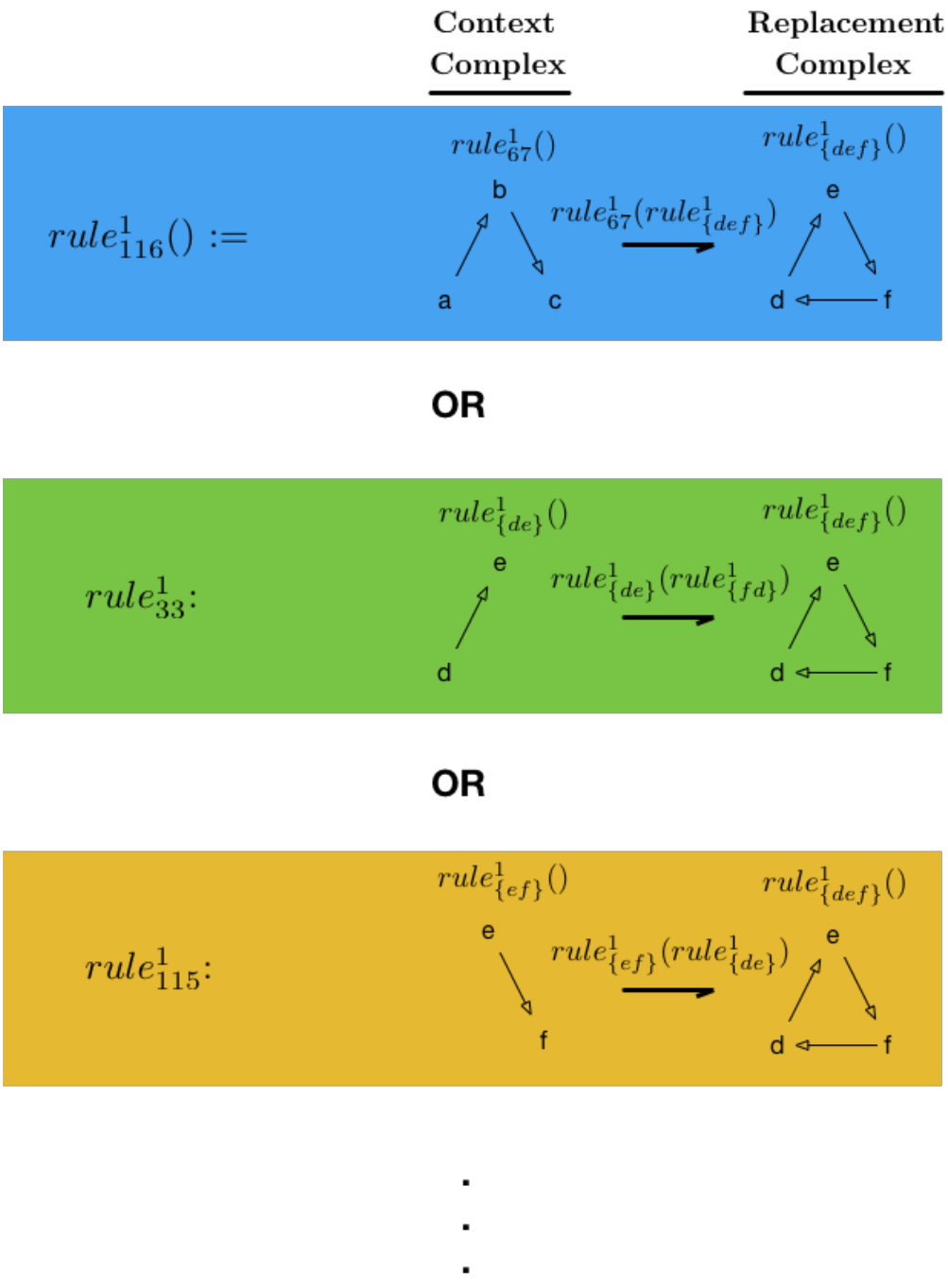


FIGURE 2.5 Multiple simplicial production rules with identical replacement-complex

CHAPTER 3

Syntactic Nonparametric Analysis of Complex Systems

In the preceding chapter we proposed modeling the state of a complex biological system/process at a given moment in time as a simplicial complex K, via a set of vertices V, and a collection S of finite non-empty subsets of V that satisfies the axioms:

Axiom 4. *For each $v \in V$, the singleton $\{v\} \in S$*

Axiom 5. *If $\sigma \in S$ and $\tau \subset \sigma$ is non-empty, then $\tau \in S$,*

where an element $\sigma \in S$ consisting of $n+1$ elements corresponds to a n-simplex of K. This modeling process would begin with the selection of a set of 0-simplices that constitute the basis of the 0^{th}-chain group and correspond to the vertex set V. These 0-simplices would represent the lowest-level granularity at which a complex biological system/process can be modeled. In an unsupervised learning setting, the basis of 0-simplices correspond to a finite set of discrete values used to represent the underlying data space of a complex biological system/process inferred via quantization. This finite, non-empty set of data-derived elements can be referred to as the alphabet Σ of pattern-primitives with which we can construct more complex patterns. For continuous-valued data, these 0-simplices can be obtained via the aforementioned quantization procedure, where continuous-values are discretized according to a finite-size quanization map.

In the SYNACX framework, the spatio-temporal dynamics and emergent behavior of complex biological systems/processes are to be learned from sequence data and represented as simplicial

grammar–that is, probabilistic generative models for sequence data inferred and explictly modeled as simplex chains in an abstract compositional language. Recall, a simplicial grammar is a collection of simplicial production rules, $SG = \{rule_i^{(d)}\}_{i<|SG|}$, defined over a vertex set $V = K^{(0)}$ (alphabet Σ), with operations occuring at dimension $(d) \geq 0$ involving simplices from $K^{(d)}$. Each rule $rule_i^{(d)} : (K_C, K_R)$ in a simplicial grammar describes a unique pair of simplicial complexes for its context-complex and replacement-complex. A *sub-grammar* is a subset of the rules that make up a given simplicial grammar. When modeling spatio-temporal structures learned directly from data, the collection of simplicial grammar production rules can explicitly model both the sequential order of operations and the temporal dependencies that underly a data sequence. Given a complex biological system/process data sequence, $\vec{x}$, it is desirable to automatically infer a collection of simplicial grammars that can generalize the syntactic and statistical properties of the data sequence using a hierarchy of latent variables that can be explicitly modeled as simplicial chains. By assuming that the syntactic structure of the data sequence can be modeled at multiple levels of granularity, from the lowest-level (fine-grain resolution) using a 0-simplex chain of primitives, $\mathbf{q}^0 \in C_0$, to increasingly higher levels using an emergent d-simplex chain of d-dimensional simplices, $\mathbf{q}^d \in C_d$, a complex biological system/process model can be defined in which the distribution over lower-level granular chains is regularized using higher-level granular chains. In addition, by factoring the probability of the data sequence under a distribution, $P(\vec{x})$, discrete subsequences can be directly modeled using the set of conditional distributions,

$$P(\vec{x}) = P(x_0)P(x_1|x_0)P(x_2|x_0, x_1)\ldots, P(x_N|x_0, \ldots, x_{N-1}).$$

The combination of simplicial chains with associated conditional probability distributions in the simplicial grammar modeling formalism provides a versatile approach to encoding the large number of highly interconnected dynamic units of a complex biological system/process into a simplicial complex which can be considered a combinatorial version of a topological space. Consequently, the invariants of each simplicial grammar can be studied from a probabilistic, topological, combinatorial and algebraic perspective, each one providing completely different measures that can be used to discriminate between classes of phenomena across multiple scales.

3.0.1 Overview

We propose constructing the underyling probabilistic model of the Simplicial Grammar formalism in an incremental and hierarchical manner. By incremental, we mean that batches of non-annotated sequence data are learned, one datum at a time. In the SYNACX framework we define a Bayesian Nonparametric predictive model and approximate inference procedure for data sequences of unbounded complexity with which we can build expressive models of complex biological systems/processes using minimal implicit assumptions or expert supervision. In designing and justifying such a model and inference procedure for biomedical datastreams, we extend definitions and inference methods for building hierarchical Pitman-Yor Processes–hierarchical models of sequential stochastic processes that generate discrete observations. Let the basis of the 0th-chain group, $\{e_1^0, e_2^0, e_3^0, ...\}$, correspond to an alphabet V. Given a k-dimensional input data sequence $\mathbf{x}_{1:i} = [x_0, x_1, x_2, ...x_i]$, the learning process begins by incrementally quantizing each each datapoint via **Algorithm 2**, such that we obtain a simplex chain of 0-simplices $\mathbf{q}_{1:i} = [e_{x_0}^0 + e_{x_1}^0 + e_{x_2}^0 + ... + e_{x_i}^0]$. The SYNACX framework works to infer and model the spatial and temporal dependencies inherent to this input simplex chain by incrementally learning a predictive probability distribution and constructing a collective grammar, $\mathcal{G}_{\mathbf{q}_{1:i}}$. To allow uncertainty in distributional assumptions and to avoid critical dependence on parametric assumptions, the probabilistic model of each simplicial grammar is represented by a set of random variables drawn from some unknown probability distribution. This unknown probability distribution is itself drawn from some prior distribution. Thus, each simplicial grammar can be parameterized using random probability measures $G_{\mathbf{q}}$ based on an underlying Hierarchical Pitman-Yor process prior. To make grammar induction computationally tractable for time-series data, we utilize a marginalized hierarchy of Hierarchical Pitman-Yor processes inspired by the language models in [83, 130] and the Sequence Memoizer [131–133]. In this marginalized hierarchy, the discount hyperparameters are stochastically optimized while the concentration parameter is set to zero.

Information about an already observed input sub-sequence $\vec{q}_{1:i} = \{e_{x_0}^0 + e_{x_1}^0 + ... + e_{x_i}^0\}$ and its probability distribution (3.1) is maintained in a collection of sub-grammars that constitute the collective grammar $\mathcal{G}$, where each sub-grammar $G_{\vec{q}_{1:i}} \in \mathcal{G}$ defines the conditional distribution over

V.

$$P(\vec{q}_{1:i}) = \prod_{j=1}^{i} P(e^0_{x_j}|e^0_{x_{j-1}}) = \prod_{j=1}^{i} G_{\vec{q}_{1:j-1}}(j) \tag{3.1}$$

Each simplicial grammar production rule, $G_{\vec{q}_{1:i}}(e^0_k)$, models the probability of observing a pattern-primitive $e^0_k \in V$, conditioned on the observed simplicial chain $\vec{q}_{1:i}$, given a Hierarchical Pitman-Yor Process prior:

$$G_\emptyset \sim PYP(d_\emptyset, H)$$

$$G_{\vec{q}_{1:i}}|G_{\pi(\vec{q}_{1:i})} \sim PYP(d_{\vec{q}_{1:i}}, G_{\pi(\vec{q}_{1:i})}) \;\; \forall \vec{q}_{1:i} \in C_0/\{\emptyset\}$$

where $\emptyset$ denotes the null set and $\pi(\vec{q}_{1:i}) = \{e^0_{x_1} + ... + e^0_{x_i}\}$ represents a variable-length truncation of the observed simplex chain $\vec{q}_{1:i} = \{e^0_{x_0} + e^0_{x_1} + ... + e^0_{x_i}\}$. In this setup, the joint probability of the observed simplicial chain $\mathbf{q}_{1:i}$ and the collective grammar $\mathcal{G}$ is given as:

$$P(\mathbf{q}_{1:i}, \mathcal{G}) = P(\mathcal{G}) \prod_{i=0}^{|\mathbf{q}|-1} G_{\mathbf{q}_{1:i}}(q_{i+1}) \tag{3.2}$$

where the rightmost term is the probability of each primitive conditioned on the 0-chain observed thus far, and $P(\mathcal{G})$ is the hierarchical prior describing the unbounded set of latent variables for the generative collective grammar based on all recent observations. To make SYNACX tractable for biomedical datastreams, a collective grammar of bounded space complexity is maintained via update operations, such that only a finite number of sub-grammars and production rules can be stored at a given moment in time. These maintenance operations given in **Algorithm 3** (lines 15-22) update the collective grammar and its hierarchical probabilistic model in an adaptive manner that is conducive to learning complex biological system/process data. In the current SYNACX implementation, simplicial production rules which share the same context-complex are grouped together into a grammar uniquely identified by the collection of simplices that constitute their respective context-complex. For example, after learning from a sequence of observations which includes the 0-simplex chain $\mathbf{q} = [\sigma^0_B, \sigma^0_A, \sigma^0_D]$, we obtain a collective grammar denoted in (3.5) consisting of various size simplicial grammar. The largest grammar in this collective grammar, $G_{\sigma^0_B+\sigma^0_A+\sigma^0_D}$, is denoted in (3.4) and depicts those simplicial production rules inferred from the simplex chain which

share the same context-complex $\{\sigma_B^0 + \sigma_A^0 + \sigma_D^0\}$. The collection of grammars denoted in (3.3) are also a part of the collective grammar. A subset of these grammars $(G_\emptyset, G_{\sigma_A^0}, G_{\sigma_B^0}, G_{\sigma_D^0}, G_{\sigma_A^0+\sigma_D^0})$ are referred to as the sub-grammars of $G_{\sigma_B^0+\sigma_A^0+\sigma_D^0}$ because their respective context-complexes all intersect the context-complex $\{\sigma_B^0 + \sigma_A^0 + \sigma_D^0\}$.

$$G_{[\emptyset]} := \begin{cases} \{\emptyset\} \to \{\sigma_A^0\} \\ \{\emptyset\} \to \{\sigma_B^0\} \\ \{\emptyset\} \to \{\sigma_C^0\} \\ \{\emptyset\} \to \{\sigma_D^0\} \\ \{\emptyset\} \to \{\sigma_E^0\} \\ \{\emptyset\} \to \{\sigma_X^0\} \\ \{\emptyset\} \to \{\sigma_Y^0\} \\ \{\emptyset\} \to \{\sigma_Z^0\} \end{cases} \quad G_{[\sigma_A^0]} := \begin{cases} \{\sigma_A^0\} \to \{\sigma_B^0\} \\ \{\sigma_A^0\} \to \{\sigma_C^0\} \\ \{\sigma_A^0\} \to \{\sigma_Y^0\} \end{cases} \quad G_{[\sigma_B^0]} := \begin{cases} \{\sigma_B^0\} \to \{\sigma_A^0\} \\ \{\sigma_B^0\} \to \{\sigma_E^0\} \\ \{\sigma_B^0\} \to \{\sigma_Y^0\} \end{cases} \quad G_{[\sigma_D^0]} := \begin{cases} \{\sigma_D^0\} \to \{\sigma_A^0\} \\ \{\sigma_D^0\} \to \{\sigma_B^0\} \end{cases}$$

$$G_{[\sigma_B^0+\sigma_A^0]} := \begin{cases} \{\sigma_B^0 + \sigma_A^0\} \to \{\sigma_Y^0\} \end{cases} \quad G_{[\sigma_C^0+\sigma_A^0]} := \begin{cases} \{\sigma_C^0 + \sigma_A^0\} \to \{\sigma_Y^0\} \\ \{\sigma_C^0 + \sigma_A^0\} \to \{\sigma_B^0\} \end{cases} \quad G_{[\sigma_A^0+\sigma_D^0]} := \begin{cases} \{\sigma_A^0 + \sigma_D^0\} \to \{\sigma_A^0\} \\ \{\sigma_A^0 + \sigma_D^0\} \to \{\sigma_B^0\} \end{cases} \tag{3.3}$$

$$G_{[\sigma_B^0+\sigma_A^0+\sigma_D^0]} := \begin{cases} \{\sigma_B^0 + \sigma_A^0 + \sigma_D^0\} \to \{\sigma_B^0\} \\ \{\sigma_B^0 + \sigma_A^0 + \sigma_D^0\} \to \{\sigma_A^0\} \end{cases} \tag{3.4}$$

$$\mathcal{G} := \begin{cases} G_\emptyset^{L_0} & \text{Root grammar} \\ G_{[\sigma_A^0]}^{L_1} \quad G_{[\sigma_B^0]}^{L_1} \quad G_{[\sigma_C^0]}^{L_1} \quad G_{[\sigma_D^0]}^{L_1} & \text{Level-1 grammar} \\ G_{[\sigma_B^0+\sigma_A^0]}^{L_2} \quad G_{[\sigma_C^0+\sigma_A^0]}^{L_2} \quad G_{[\sigma_A^0+\sigma_D^0]}^{L_2} & \text{Level-2 grammar} \\ G_{[\sigma_B^0+\sigma_A^0+\sigma_D^0]}^{L_3} & \text{Level-3 grammar} \end{cases} \tag{3.5}$$

The SYNACX algorithm takes a Bayesian approach to learning this collective grammar by treating the distributions over subsequent input observations as latent variables on which a hierarchical nonparametric prior is placed. Prediction of a subsequent input observation is performed by averaging over the posterior distribution conditioned on the simplex chains observed thus far. To

achieve this, inference is performed in the Chinese Restaurant Franchise representation [83, 130], where the posterior distribution is incrementally updated after each observation. In this representation, the state of each Pitman-Yor process associated with a grammar $G_{\mathbf{q}} \in \mathcal{G}$ is represented in terms of two sets of counts

$$\mathcal{C}_{G_{\mathbf{q}}} \equiv \{c_{G_{\mathbf{q}}\sigma k}\}_{\sigma\in\Sigma, k\in\{1,\ldots,t_{G_{\mathbf{q}}\sigma}\}} \tag{3.6}$$

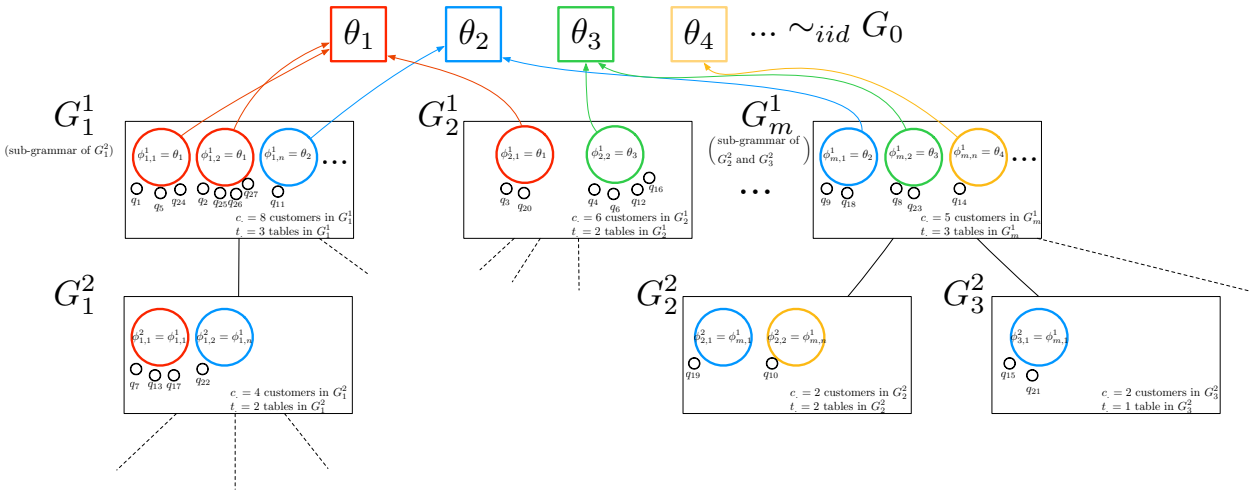
$$\mathcal{T}_{G_{\mathbf{q}}} \equiv \{t_{G_{\mathbf{q}}\sigma}\}_{\sigma\in\Sigma} \tag{3.7}$$

The count $c_{G_{\mathbf{q}}\sigma k}$ corresponds to the number of draws of type σ from the Pitman-yor process associated with a grammar $G_{\mathbf{q}}$ assigned to the k-th draw from its base measure $G_{\pi(\mathbf{q})}$, whereas $t_{G_{\mathbf{q}}\sigma}$ corresponds to the number of draws from this base measure. The sets of counts for all grammars $G_{\mathbf{q}} \in \mathcal{G}_{\mathbf{q}_{1:i}}$, is referred to as the state of the collective grammar $State_{\mathcal{G}_{\mathbf{q}_{1:i}}} = \{\mathcal{C}_{G_{\mathbf{q}}}, \mathcal{T}_{G_{\mathbf{q}}}\}_{G_{\mathbf{q}}\in\mathcal{G}_{\mathbf{q}_{1:i}}}$ and serves as a sufficient statistic for its joint distribution.

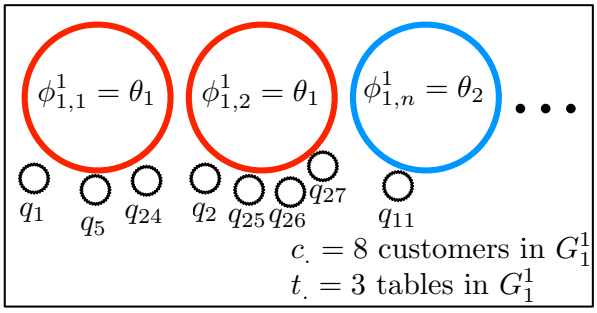
To compute the predictive probability of a primitive associated with a production rule in grammar $G_{\mathbf{q}}$, we follow the Poly urn model as follows:

$$\mathrm{P}(\sigma|G_{\mathbf{q}}, State_{\mathcal{G}_{\mathbf{q}_{1:i}}}) = \frac{c_{G_{\mathbf{q}}\sigma} - d_{G_{\mathbf{q}}} t_{G_{\mathbf{q}}\sigma}}{c_{G_{\mathbf{q}}\cdot}} + \frac{d_{G_{\mathbf{q}}} t_{G_{\mathbf{q}}\cdot}}{c_{G_{\mathbf{q}}\cdot}} \mathrm{P}(\sigma|G_{\pi(\mathbf{q})}, State_{\mathcal{G}_{\mathbf{q}_{1:i}}}) \tag{3.8}$$

when $c_{G_{\mathbf{q}}\cdot} \neq 0$ and $\mathrm{P}(\sigma|G_{\mathbf{q}}, State_{\mathcal{G}_{\mathbf{q}_{1:i}}}) = \mathrm{P}(\sigma|G_{\pi(\mathbf{q})}, State_{\mathcal{G}_{\mathbf{q}_{1:i}}})$ otherwise. In order to compute the predictive distribution $\mathrm{P}(\sigma|G_{\mathbf{q}}, \mathbf{q}_{1:i})$ of a primitive σ in some grammar $G_{\mathbf{q}}$ after having observed input $\mathbf{q}_{1:i}$, (3.8) has to be averaged with respect to the posterior distribution over states $\mathrm{P}(State_{\mathcal{G}_{\mathbf{q}_{1:i}}}|\mathbf{q}_{1:i})$. In the SYNACX framework we implement an approximate inference scheme with which this posterior distribution $P(State_{\mathbf{q}_{1:i}})$ can be estimated by using samples drawn from it. These samples come in the form of various permutations of the counts $\{\mathcal{C}_{G_{\mathbf{q}}}, \mathcal{T}_{G_{\mathbf{q}}}\}_{G_{\mathbf{q}}\in\mathcal{G}_{\mathbf{q}_{1:i}}}$. This scheme allows us to sequentially update samples from $P(State_{\mathbf{q}_{1:i-1}}|\mathbf{q}_{1:i-1})$ such that they become samples from $P(State_{\mathbf{q}_{1:i}})$. Intuitively, this update scheme can be understood as a Particle Filter or Sequential Monte Carlo method [134], in which a set of samples/particles $\{Sample_{\mathbf{q}}^{j}\}_{j=1}^{MaxParticles}$ is used to approximate the current estimate of the posterior distribution over the model space, conditional on observed data. This approximation is represented as a weighted sample, with each particle



(a) Chinese Restaurant Franchise representation of a collective grammar $\mathcal{G}_\mathbf{q}$



$c_{\theta_1} = 7$ customers in G_1^1 served dish $\sigma = \theta_1$ $\quad$ $c_{\theta_2} = 1$ customer in G_1^1 served dish $\sigma = \theta_2$

$t_{\theta_1} = 2$ tables in G_1^1 serving dish $\sigma = \theta_1$ $\quad$ $t_{\theta_2} = 1$ table in G_1^1 serving dish $\sigma = \theta_2$

(b) Chinese Restaurant representation of simplicial sub-grammar from $\mathcal{G}_\mathbf{q}$

FIGURE 3.1 Chinese Restaurant representation of Simplicial Grammar

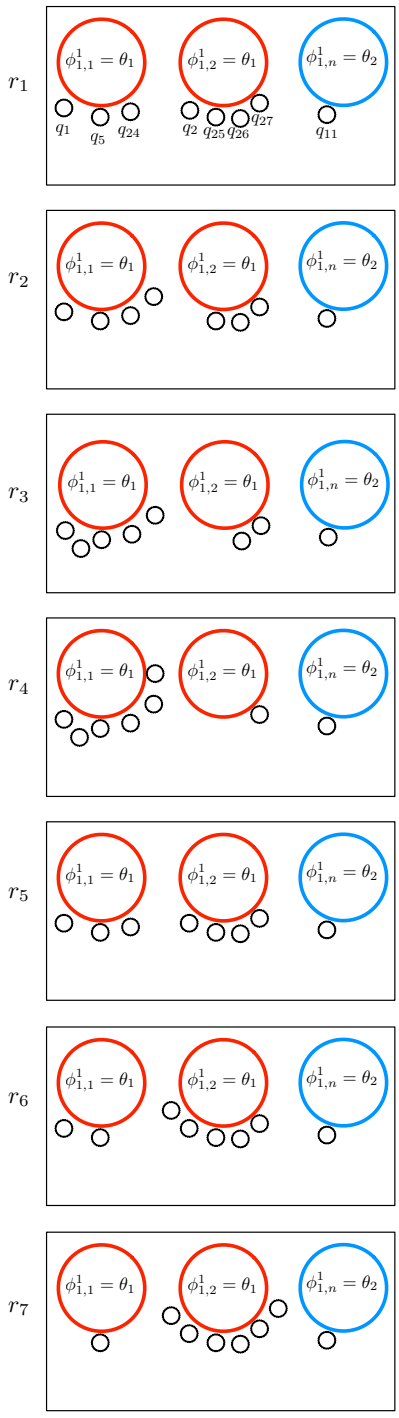


FIGURE 3.2 Possible customer and table configurations based on observations assigned to G_1^1. Each configuration $r \in R_{G_1^1\sigma} \in \mathcal{R}_{\mathcal{G}}$ for a given dish σ in sub-grammar G_1^1 is sampled from a partition distribution generated for $N = 8$ observations

maintaining a possible state of the model. The particle filter inference procedure is implemented in SYNACX using a single particle. Updates in this scheme can be understood in the following way: for every new observation associated with the collective grammar, counts are incremented for some subset of the grammars in $State_{\mathcal{G}_{\mathbf{q}_{1:i}}}$. Although using more than one particle can improve the accuracy of the posterior distribution, this improvement comes at the cost of algorithmic complexity, which increases linearly with the number of particles used. For online predictive modeling tasks performed in a non-distributed computing environement, we recommend using a single particle implementation of the SYNACX algorithm. In the Chinese Restaurant representation, this particle represents the state of the collective grammar, $State_{\mathcal{G}_{\mathbf{q}_{1:i}}}$, and maintains the current estimate of the posterior distribution from which the predictive probability of a primitive σ given a grammar $G_{\mathbf{q}} \in \mathcal{G}_{\mathbf{q}_{1:i}}$, $P(\sigma|G_{\mathbf{q}}, State_{\mathcal{G}_{\mathbf{q}_{1:i}}})$, can be computed. In view of a new observation q_{i+1}, $State_{\mathcal{G}_{\mathbf{q}_{1:i}}}$ is updated by drawing samples from it, such that $P(State_{\mathcal{G}_{\mathbf{q}_{1:i}}}, \mathbf{q}_{1:i})$ becomes $P(State_{\mathcal{G}_{\mathbf{q}_{1:i+1}}}, \mathbf{q}_{1:i+1})$.

Following the $i+1^{\text{th}}$ observation, q_{i+1}, we can obtain the configuration of the hierarchical Pitman-Yor processes (Figure 3.1) associated with the collective grammar $\mathcal{R}_{\mathcal{G}_{\mathbf{q}_{1:i+1}}} \sim \mathcal{E}_{|\mathbf{q}_{1:i+1}|}$, where $\mathcal{E}_{|\mathbf{q}_{1:i+1}|}$ describes the two-parameter Ewen's $\mathcal{E}S_N(d,\alpha)$ distribution of $N = |\mathbf{q}_{1:i+1}|$ observations over K partitions, for all possible N and K [135]. In order to sample such a partition distribution, we follow Ewen's sampling formula [136]:

$$P(m_1, ..., m_n) = \frac{n!}{\prod_{i=1}^{n}(i!)^{m_i} m_i!} \mu(m_1, ..., m_n)$$

with

$$\mu(m_1, ..., m_n) = E\Big[\sum \prod_{i=1}^{n} \prod_{j=1}^{m_i} V_{n(i,j)}^{i}\Big].$$

We generalize this formula for the Pitman-Yor Process, with $\mu(m_1, ..., m_n) = \mu_{d,\alpha}(m_1, ..., m_n)$ describing the mean probability distribution of a partition of length k , via the formula:

$$\mu_{d,\alpha=0}(m_1, ..., m_n) = \frac{[d]_d^{k-1}}{[1]_1^{n-1}} \prod_{j=1}^{n}([1-d]_1^{j-1}) \tag{3.9}$$

For a hierarchy of Hierarchical Pitman-Yor processes, where the partition distribution is conditioned on the length of partitions being $t_{G_{\mathbf{q}}}$, the joint posterior distribution can be denoted as fol-

lows[1]:

$$P(State_{\mathcal{G}_{\mathbf{q}_{1:i+1}}}, \mathbf{q}_{1:i+1}) \;=\; \left(\prod_{\sigma\in\Sigma} H(\sigma)^{t_{G_\emptyset\sigma}}\right) \prod_{G_\mathbf{q}\in\mathcal{G}} \left(\frac{[d_{G_\mathbf{q}}]_{d_{G_\mathbf{q}}}^{t_{G_\mathbf{q}\cdot}-1}}{[1]_1^{c_{G_\mathbf{q}\cdot}-1}} \prod_{\sigma\in\Sigma} \prod_{r\in R_{G_\mathbf{q}\sigma}\in\mathcal{R}_\mathcal{G}} [1-d_{G_\mathbf{q}}]_1^{|r|-1}\right) \tag{3.10}$$

This update scheme is interleaved with the construction of the collective grammar and the prediction of the next primitive.

3.0.2 Implementation

For each observation in an input data sequence, **Algorithm 3** works to first compute the predictive probability distributions for the current observation by grammar induction. During this stage, the algorithm identifies a grammar $G_{K_C} \in \mathcal{G}$ whose context-complex intersects the most with the collection of observed simplex-chains stored in the memory-complex $\mathcal{M}$. The predictive probability distribution of this grammar is computed in **Algorithm 7** according to (3.8) and subsequently used to update the state of a subset of grammars in the collective, $\mathcal{G} = \{G_\mathbf{q}\}$, whose context-complex intersect the most with G_{K_C}.

Incremental construction of $\mathcal{G}$ is handled by **Algorithm 4** and results in the creation of two or fewer sub-grammar for every observation. To do this either a new sub-grammar, G_K, that is a direct extension of an existing sub-grammar, $G_{\pi(K)}$, with overlapping context complex is created via the generateGrammar() subroutine, or an intermediate sub-grammar is created instead via **Algorithm 5**. This new intermediate sub-grammar is linked to both $G_{\pi(K)}$ and the new grammar G_K . In both cases, the counts of grammar G_K are used to estimate the predictive distribution. Following construction, the current observation is integrated into the underlying probability model via **Algorithm 8** by adjusting counts in all sub-grammars related to G_K. During this process, **Algorithm 10** computes the partial stochastic gradient of (3.8) with respect to the discount hyperparameters $d_\mathbf{q}$ of each sub-grammar with a context-complex intersecting that of G_K. Starting at grammar G_K and progressing down to the root sub-grammar $G_\emptyset$, **Algorithm 8** makes a stochastic decision to

[1]Here we use Kramp's general notation to concisely express the product of the factors of an arithmetic progression as $[c]_b^a \equiv \prod_{i=0}^{a-1} c + ib$

increment c_σ if t_σ was incremented in the sub-grammar below. This follows the Chinese Restaurant process metaphor where each observation of σ in grammar $\mathcal{G}_{\mathbf{q}_{1:i-1}}$ corresponds to a customer in a restaurant $G_{\mathbf{q}}$ who is served dish σ, and each table in each restaurant $G_{\mathbf{q}}$ being a draw from its base measure, $G_{\pi(\mathbf{q})}$, also corresponds to a customer in the parent restaurant $G_{\pi(\mathbf{q})}$. This relation is a consequence of the hierarchy of Pitman-Yor measures and expressed as follows:

$$c_{G_{\pi(\mathbf{q})}\sigma} = c_{G_{\pi(\mathbf{q})}\sigma} + \sum_{G_{\mathbf{q}}} t_{G_{\mathbf{q}}\sigma}, \begin{cases} c_{G_{\pi(\mathbf{q})}\sigma} & = 1 \text{ if dish } \sigma \text{ is observed in } G_{\pi(\mathbf{q})} \\ c_{G_{\pi(\mathbf{q})}\sigma} & = 0 \text{ otherwise.} \end{cases} \tag{3.11}$$

In addition, if c is larger than θ_{max} in any sub-grammar, the counts c_σ and potentially t_σ are reduced. The gradients for the set of discount parameters $\vec{\delta} = [\delta_0, \delta_1, ..., \delta_\omega]$ are then updated based on the calculated gradients using learning rate η by defining $\delta_n = \delta_\omega^{\gamma^{n-\omega}}$ for $n \geq \omega$, and δ_n for $n \leq \omega$. Lastly, the current observation is added to the memory-complex $\mathcal{M}$.

3.0.3 Algorithm Pseudocode

Algorithm 3 SYNACX

procedure SYNACX(**q**)
  $\mathcal{M} \leftarrow \{\}$; ▷ Initialize memory-complex
  $State_{\mathcal{G}} \leftarrow \{\}$; ▷ Initialize state of collective grammar $\mathcal{G}$
  $G \leftarrow \{\}$; ▷ Initialize root grammar of $\mathcal{G}$
  $\vec{\delta} \leftarrow [\delta_0, \delta_1, \ldots, \delta_\omega]$ ▷ Initialize $\mathcal{HPYP}$ discount hyperparameters
  $N \leftarrow 1$ ▷ number of grammars in $\mathcal{G}$
  ζ_{max} ▷ max length of simplex chains
  Υ_{max} ▷ max number of grammars in $\mathcal{G}$
  θ_{max} ▷ max number of observations associated with each grammar
  M_{max} ▷ max number of observations associated with memory-complex

  for $i = 0 : |\mathbf{q}| - 1$ **do**

    $\nabla \leftarrow \vec{0}$ ▷ hyperparameter gradient
    while $|\mathcal{M}| \geq M_{max}$ **do** ▷ Maintenance operation enforcing bound on $\mathcal{M}$
      $\mathcal{M}\leftarrow$deleteAndUpdateMemory($\mathcal{M}$)
      $\mathcal{G}\leftarrow$deleteAndUpdateGrammar(G_*,$\mathcal{G}$)
    end while
    while $N > (\Upsilon_{max} - 2)$ **do** ▷ Maintenance operation enforcing bound on $\mathcal{G}$
      $N \leftarrow N - 1$
      $\mathcal{G}\leftarrow$deleteAndUpdateGrammar(G_{random},$\mathcal{G}$)
    end while
    $\mathcal{G}_{\mathbf{q}_{1:i+1}}, G_{K_C} \leftarrow$ grammarInduction($\mathcal{M}_{\mathbf{q}_{1:i}}, \mathcal{G}_{\mathbf{q}_{1:i}}$)
    $P_{K_C} \leftarrow$ computePredictiveDistro($G_{K_C}, P = \vec{0}, m_f = 1.0$)
  ▷ Update probability of relevant grammar rules following incorporation of observation $\mathbf{q}_{i+1}$
    updateGrammarState($G_{K_C}, P_{K_C}, \mathbf{q}_{i+1}$, TRUE)
    $\vec{\delta} \leftarrow \vec{\delta} + \frac{\eta \nabla}{P_{K_C}(\mathbf{q}_{i+1})}$
    $\mathcal{M} \leftarrow \mathcal{M} + \mathbf{q}_{i+1}$
  end for
end procedure

Algorithm 4 grammarInduction

function $\mathcal{G}_{\mathbf{q}_{1:i+1}}, G_{K_C} \leftarrow$grammarInduction($\mathcal{M}_{\mathbf{q}_{1:i}}, \mathcal{G}_{\mathbf{q}_{1:i}}$)

    Find the grammar $G_{K_C^*}$ in $\mathcal{G}$ whose context-complex K_C^* intersects the most with $\mathcal{M}_{\mathbf{q}_{1:i}}$.
    if K_C^* of $G_{K_C^*}$ is a subcomplex of $\mathbf{q}_{1:i+1}$ and $|K_C^*| < \zeta_{max}$ **then**
        $\mathcal{G}_{\mathbf{q}_{1:i+1}}$,$G_{\mathbf{q}_{1:i+1}} \leftarrow$generateGrammar($\mathbf{q}_{1:i+1}, G_{K_C^*}, \mathcal{G}_{\mathbf{q}_{1:i}}$)
        $G_{K_C} \leftarrow G_{\mathbf{q}_{1:i+1}}$
        $N \leftarrow N + 1$
        return $\mathcal{G}_{\mathbf{q}_{1:i+1}}, G_{K_C}$
    else
        $\mathcal{G}_{\mathbf{q}_{1:i+1}}, G_{\mathbf{q}_{1:i+1}} \leftarrow$subGrammarInduction($\mathbf{q}_{1:i+1}, G_{K_C^*}, \mathcal{G}_{\mathbf{q}_{1:i}}$)
        $G_{K_C} \leftarrow G_{\mathbf{q}_{1:i+1}}$
        $N \leftarrow N + 1$
        return $\mathcal{G}_{\mathbf{q}_{1:i+1}}, G_{K_C}$
    end if
end function

Algorithm 5 subGrammarInduction

function $\mathcal{G}_{\mathbf{q}_{1:i+1}}, G_{\mathbf{q}_{1:i+1}} \leftarrow$subGrammarInduction($\mathbf{q}_{1:i+1}, G_{K_C^*}, \mathcal{G}_{\mathbf{q}_{1:i}}$)
    $d = 1.0$
    if $G_{K_C^*}$ is the root grammar **then** ▷ i.e. K_C^* = null complex
        $d \leftarrow d\delta_0$
    else
        for $i = (|\pi(K_C^*)| + 1) : |K_C^*|$ **do**

▷ $\pi(K_C^*)$ denotes the context-complex of an existing sub-grammar $G_{\pi(K_C^*)}$, such that $\pi(K_C^*)$ is the largest subcomplex of K_C^* learned thus far.

            if $i \leq \omega$ **then**
                $d \leftarrow d\delta_i$
            else
                $d \leftarrow d\delta_\omega^\gamma{}^i$
            end if
        end for
    end if
    $\mathcal{L} \leftarrow$largest subcomplex in common with K_C^* and $\mathcal{M}$
    $\mathcal{G}_{\mathbf{q}_{1:i}}, G_{\mathcal{L}} \leftarrow$ generateGrammar($\mathcal{L}, \pi(K_C^*), \mathcal{G}_{\mathbf{q}_{1:i}}$)
    $N \leftarrow N + 1$
    $\pi(G_{\pi(K_C^*)}) \leftarrow G_{\mathcal{L}}$

▷ $\pi(G_{\pi(K_C^*)})$ denotes the largest sub-grammar, such that $\pi(K_C^*) \leftarrow \mathcal{L}$

Algorithm 6 subGrammarInduction (continued)

$d = 1.0$
if $G_{\mathcal{L}}$ is the root grammar **then**
  $d \leftarrow d\delta_0$
else
  for $i = (|\pi(\mathcal{L})| + 1) : |\mathcal{L}|$ **do**
    if $i \leq \omega$ **then**
      $d \leftarrow d\delta_i$
    else
      $d \leftarrow d\delta_\omega^i$
    end if
  end for
end if
for $\sigma \in \Sigma$ **do**
  $\varrho \leftarrow$ createPartition $(c_\sigma^{G_{K_C^*}}, t_\sigma^{G_{K_C^*}}, d^{G_{K_C^*}})$
  $t_\sigma^{G_{\mathcal{L}}} \leftarrow t_\sigma^{G_{K_C^*}}$
  $t_\sigma^{G_{K_C^*}} \leftarrow 0$
  for i= 1 : $|\varrho|$ **do**
    $\tau \leftarrow 1$
    for j = 2 : ϱ[i] **do**
      $h \leftarrow 0$
      $h \leftarrow 1$ w.p. $\frac{\tau \frac{d^{G_{K_C^*}}}{d^{G_{\mathcal{L}}}} - d^{G_{K_C^*}}}{j-1-d^{G_{K_C^*}}}$
      $\tau \leftarrow \tau + h$
    end for
    $t_\sigma^{G_{K_C^*}} \leftarrow t_\sigma^{G_{K_C^*}} + \tau$
  end for
  $c_\sigma^{G_{\mathcal{L}}} \leftarrow t_\sigma^{G_{K_C^*}}$
end for
$\mathcal{G}_{\mathbf{q}_{1:i+1}}, G_{\mathbf{q}_{1:i+1}} \leftarrow$ generateGrammar$(\mathbf{q}_{1:i+1}, G_{\mathcal{L}}, \mathcal{G}_{\mathbf{q}_{1:i}})$
$N \leftarrow N + 1$
return $\mathcal{G}_{\mathbf{q}_{1:i+1}}, G_{\mathbf{q}_{1:i+1}}$
end function

Algorithm 7 computePredictiveDistro

function $P_{K_C} \leftarrow$ computePredictiveDistro(G_{K_C}, P, m_f)
  if $c^{G_{K_C}} > 0$ **then**
    for $\sigma \in \Sigma$ **do**
      $P(\sigma) \leftarrow P(\sigma) + m_f(\frac{c_\sigma^{G_{K_C}} - t_\sigma^{G_{K_C}} d^{G_{K_C}}}{c^{G_{K_C}}})$
    end for
  end if
  if $\pi(G_{K_C})$ exists **then**
    $P \leftarrow$ computePredictiveDistro($\pi(G_{K_C}), P, d^{G_{K_C}} \cdot m_f$)
  else
    $P \leftarrow (1 - d^{G_{K_C}} \cdot m_f)P + d^{G_{K_C}} \cdot m_f P_\Sigma$
    $P_{K_C} \leftarrow P$
    return P_{K_C}
  end if
end function

Algorithm 8 updateGrammarState

function updateGrammarState(G_K, P_K, σ, $boolean_{addC}$)
  $P'_K(\sigma) \leftarrow P_K(\sigma)$
  if $c^{G_K} > 0$ **then**
    $P'_K(\sigma) \leftarrow (P_K(\sigma) - \frac{c_\sigma^{G_K} - t_\sigma^{G_K} d^{G_K}}{c^{G_K}})(\frac{c^{G_K}}{t^{G_K} d^{G_K}})$
    $f_n \leftarrow t^{G_K} d^{G_K}$
    $f_d \leftarrow c_\sigma^{G_K} + d^{G_K}(t^{G_K} P'_K(\sigma) - t_\sigma^{G_K})$
  end if
  if $boolean_{addC}$ and $c^{G_K} > 0$ **then**
    $c_\sigma^{G_K} \leftarrow c_\sigma^{G_K} + 1$
    $boolean_{addC} \leftarrow$FALSE
    $boolean_{addC} \leftarrow$TRUE w.p. $P'_K(\sigma)(\frac{f_n}{f_d})$
    if $boolean_{addC}$ **then**
      $t_\sigma^{G_K} \leftarrow t_\sigma^{G_K} + 1$
    end if
  else if $boolean_{addC}$ **then**
    $c_\sigma^{G_K} \leftarrow c_\sigma^{G_K} + 1$
    $t_\sigma^{G_K} \leftarrow t_\sigma^{G_K} + 1$
  end if

Algorithm 9 updateGrammarState (continued)

updateGrammarGradient($G_K, c^{G_K}, t_\sigma^{G_K}, t^{G_K}, P_K^{'}, d^{G_K}, m_f$)
updateGrammarState($\pi(G_K), P_K^{'}, \sigma$, $boolean_{addC}$)
while $c^{G_K} > \theta_{max}$ **do**
  $B = [\frac{c_{e_0}^{G_K}}{c^{G_K}}, \frac{c_{e_1}^{G_K}}{c^{G_K}}, \ldots, \frac{c_{e_{|\Sigma|}}^{G_K}}{c^{G_K}}]$
  $e_* \leftarrow$ sample(B) ▷ $B_{e_j} = \frac{c_{e_j}}{c}$
  $\varrho \leftarrow$ createPartition$(c_{e_*}^{G_K}, t_{e_*}^{G_K}, d^{G_K})$
  $\psi \leftarrow (\frac{1}{c_{e_*}^{G_K}})\varrho$
  $i \leftarrow$ sample(ψ)
  if $\varrho_i = 1$ **then**
    $t_{e_*}^{G_K} \leftarrow t_{e_*}^{\mathcal{N}} - 1$
  end if
  $c_{e_*}^{G_K} \leftarrow c_{e_*}^{\mathcal{N}} - 1$
end while
end function

Algorithm 10 updateGrammarGradient

function updateGrammarGradient($G_K, c^{G_K}, t_\sigma^{G_K}, t^{G_K}, P_K^{'}, d^{G_K}, m_f$)
  if $c^{G_K} > 0$ **then**
    if $|K| = 0$ **then**
      $\psi \leftarrow \frac{1.0}{\delta_0}$
      $\nabla_0 \leftarrow \nabla_0 + (d^{G_K}(t^{G_K} * P_K^{'} - t_\sigma^{G_K})\psi / c^{G_K})m_f$
    else
      $z \leftarrow |\pi(K)| + 1$
      while $z \leq |K|$ and $z < \omega$ **do**
        $\psi \leftarrow \frac{1.0}{\delta_z}$
        $\nabla_z \leftarrow \nabla_z + (d^{G_K}(t^{G_K} * P_K^{'} - t_\sigma^{G_K})\psi / c^{G_K})m_f$
      end while
      if $|K| \geq \omega$ **then**
        $a \leftarrow z - \omega$
        $b \leftarrow |K| - z + 1$
        $\psi \leftarrow \gamma^a(1-\gamma^b)/((1-\gamma)\delta_\omega)$
        $\nabla_\omega \leftarrow \nabla_\omega + (d^{G_K}(t^{G_K} * P_K^{'} - t_\sigma^{G_K})\psi / c^{G_K})m_f$
        $\psi \leftarrow \log(\delta_\omega)(a\gamma^{a-1} - (a+b)\gamma^{a+b-1})/(1-\gamma) + (\gamma^a - \gamma^{a+b})/(1-\gamma)^2$
        $\nabla_{\omega+1} \leftarrow \nabla_{\omega+1} + (d^{G_K}(t^{G_K} * P_K^{'} - t_\sigma^{G_K})\psi / c^{G_K})m_f$
      end if
    end if
  end if
end function

CHAPTER 4

Predictive Analytics for Biomedical Datastreams

To determine if the current implementation of the SYNACX framework is sufficient for learning and building predictive models from biomedical datastreams we performed a series of experiments in which the performance of the framework and various neural network architectures were evaluated under two different scenarios. In both scenarios, we evaluated the performance of predictive models constructed directly from non-annotated complex biological system/process data in an online and unsupervised setting. Thus, the model structure and its parameters were learned in an incremental manner directly from empirical data. Since many of these systems and processes are dynamic, it is desirable to learn from time-series/sequence data as opposed to static datasets. However, unlike the case in some domain applications where a finite length sequence or time-series dataset is sufficient for learning, in many practical scenarios, it is also necessary to learn from a datastream consisting of a constant flow of new datapoints with no definitive start or end point. The SYNACX framework uniquely posesses the ability to construct predictive models in an unsupervised and online setting through a Bayesian inference process that extracts the variable-order temporal dependencies found in sequence data and models them in a data abstraction that is generalizable for multi-scale modeling.

4.1 Primary Aim

As new data modalities become available for learning, they may introduce datastreams whose structural and statistical properties differ from past observations due to variability in sampling procedure, measurement noise, storage constraints, etc. Furthermore, when a given biomedical datastream consists of a sequence of measurements taken from a biological process, the underlying event or sub-process to which they may correspond is often unknown. Thus, a data-sequence sampled from one biomedical datastream may differ from another, in terms of its length, dimension or information content. The primary aim of this chapter is to apply the SYNACX framework and other machine learning models to biomedical datastreams in order to evaluate their capacity to perform general-purpose learning from sequence data. To achieve this, the SYNACX framework utilizes an approximate inference procedure that exploits the Bayesian nonparametric probabilistic structure of the simplicial grammar modeling formalism and allows for density estimation without necessitating extensive retraining. Recall, the key assumption underyling a simplicial grammar and each of its simplicial production rules, is the existence of a set of random variables drawn from some unknown probability distribution. This unknown probability distribution is itself drawn from some prior distribution. To allow uncertainty in distributional assumptions and to avoid critical dependence on parametric assumptions, each simplicial rewrite-rule is based on a Hierarchical Pitman-Yor process prior [137] inferred during the incremental learning process. Our goal is to predict the macro-level behavior and properties of a complex system using a set of local spatio-temporal patterns describing the relations between lower-level elements. To achieve this, patterns are abstractly represented using simplicial grammar. Put simply, we are representing the behavior of a complex system by a set of grammar rules which can generate many of the system's possible spatial and temporal patterns. The syntactic and nonparametric analysis of a complex system begins with the decomposition of the collective simplicial grammar $\mathcal{G}$ of a complex system into G_j sub-grammars. This decomposition process can be thought of as a partition of the complex system behavior into its functional components. Using the spatio-temporal patterns of the system, we try to deduce the decomposition by treating the G_j as latent random distributions. While Markov chains and Hidden Markov models continue to be among the most popular methodologies for modeling time-series

and sequence data, the focus of SYNACX algorithm is to model sequences directly, without the use of hidden states or strict Markov assumptions. Since we are interested in incremental learning from data sequences of unbounded length over time, the use of a Hierarchical Pitman-Yor process prior enables us to construct and integrate predictive models with conditional probabilities that are coupled hierarchically. This allows for the number of variables and the order of temporal dependencies to grow with sequence length rather than require explicit specification *a priori* as is the case in most Markov models. Thus, the conditional probabilities used to describe simplicial grammar rules, with operating dimension $d \geq 0$, are parameterized using random probability measures $G_{\mathbf{q}}$:

$$P(\sigma_{1:T}) = \prod_{i=1}^{T} P(\sigma_i|\sigma_{i-1}) = \prod_{i=1}^{T} G_{\sigma_{1:i-1}}(\sigma_i) \tag{4.1}$$

Each random probability measure, $G_{\mathbf{q}}(\sigma)$, models the probability of observing a simplex $\sigma \in K$, conditioned on a simplex chain of previously observed simplices $\mathbf{q} \in K^{(d)}$, given a hierarchical Pitman-Yor Process prior:

$$G_{\emptyset} \sim PYP(\alpha_{\emptyset}, d_{\emptyset}, H)$$
$$G_{\mathbf{q}}|G_{\pi(\mathbf{q})} \sim PYP(\alpha_{\mathbf{q}}, d_{\mathbf{q}}, G_{\pi(\mathbf{q})}) \;\; \forall \, \mathbf{q} \in C_d \backslash \{\emptyset\}$$

where $\emptyset$ denotes the empty chain and $\pi(\mathbf{q})$ represents a variable-length truncation of chain $\mathbf{q}$. The treatment of each sub-grammar as a latent variable allows us to express our prior assumptions about the form of each G_j and the dependence among the G_j for different j.

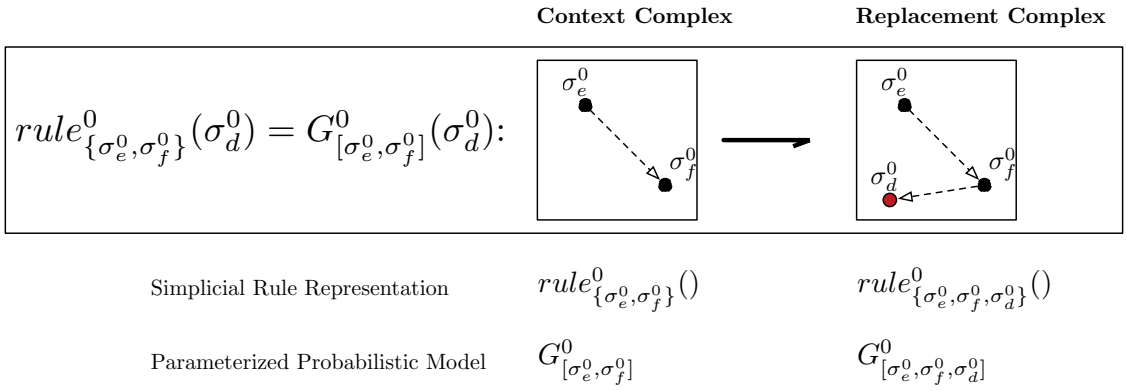


FIGURE 4.1 0-dimensional simplicial rewrite rule depicting a temporal dependency between 0-dimensional simplices (vertices)

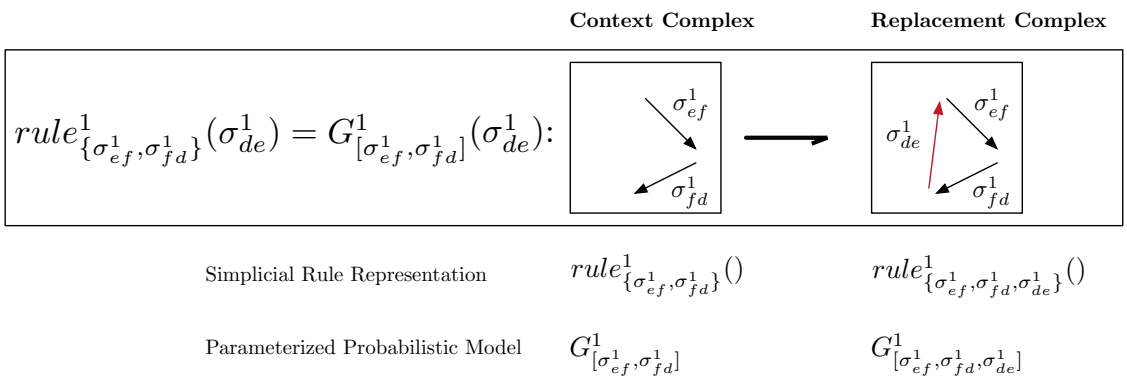


FIGURE 4.2 1-dimensional simplicial rewrite rule depicting temporal dependencies between 1-dimensional simplices (edges)

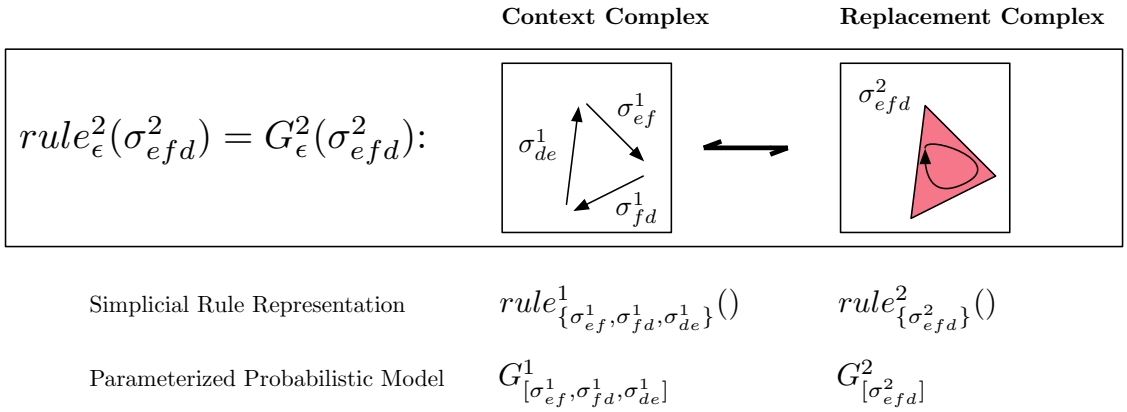


FIGURE 4.3 Depiction of boundary homomorphism between a sequence of binary relations (1-boundary cycle) and 2-dimensional simplicial complex

Thus, the collection of rules for each simplex $\sigma \in K^d$, following a context-complex $\mathbf{q}$ describes the simplicial sub-grammar $G_{\mathbf{q}}$ represented by the probability vector $G_{\mathbf{q}} = [G_{\mathbf{q}}(\sigma)]_{\sigma \in K^d}$. The collective (possibly unbounded) set of simplicial rules (latent variables) in the grammar is thereby denoted $\mathcal{G} = \{G_{\mathbf{q}}\}_{\mathbf{q} \in C_d}$. Examples of simplicial rules at various dimensions and a boundary homomorphism are illustrated in Figures 4.1,4.2,4.4 and 4.3, respectively. In this setup, given an input data sequence $\vec{x}_{1:i}$, the joint probability of its simplex chain representation $\sigma_{1:i}$ and $\mathcal{G}$ is given as:

$$P(\mathbf{q}, \mathcal{G}) = P(\mathcal{G}) \prod_{i=0}^{|\mathbf{q}|-1} G_{\sigma_{1:i}}(\sigma_{i+1}) \tag{4.2}$$

where the rightmost term is the probability of each simplex conditioned on the simplex chain of observations thus far, and $P(\mathcal{G})$ the prior over the unbounded set of latent variables for the collective grammar.

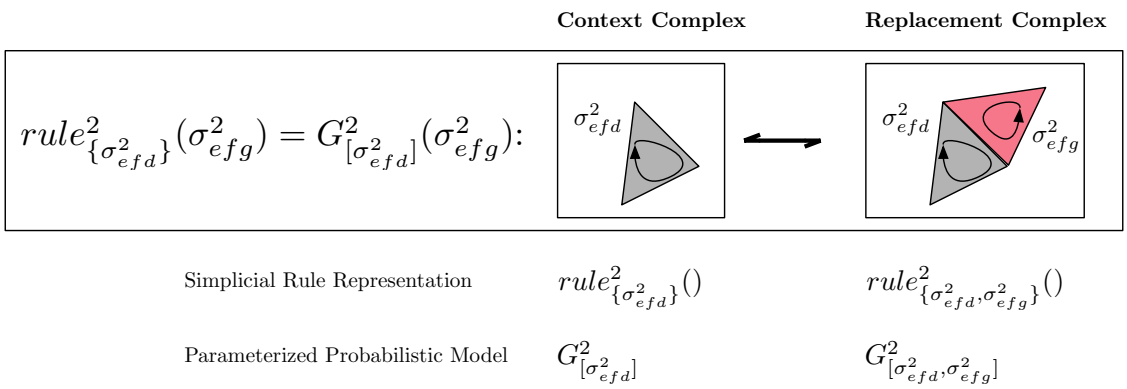


FIGURE 4.4 2-dimensional simplicial rewrite rule depicting temporal dependencies between 2-dimensional simplices; each 2-d simplex, σ^2, representative of the existence of a ternary relation between (3) 0-d simplices as well as the ordered sequence of (3) 1-d simplices and their binary relations

4.2 Scenario 1: Modeling a myocardial infarction from electrocardiography data

In this scenario we evaluate the applicability of the SYNACX framework and neural network architectures in learning predictive models of cardiac electrical activity during myocardial infarction from electrocardiography data. In addition, we highlight why learning algorithms designed to handle uncertainty become necessary when building comprehensive predictive models of biological phenomena from data that may span multiple spatial and temporal resolutions.

4.2.1 Motivation

Electrocardiography (ECG) is a common diagnostic tool for cardiovascular disease. The finite-length waveform and the continuous datastream from which it is sampled, can be used to approximate the structure of the human heart and the function of its electrical conduction system over time. Specifically, when electrodes are attached to the skin, the currents produced by the heart's electrical activity can be measured and transformed into waveforms that can be used to evaluate the heart's depolarization and repolarization cycle. The ECG waveform illustrates the sequence of electric potentials occurring in cardiac cells throughout this cycle. As this sequence of electric potentials is studied over time, various spatial and temporal patterns can be recognized and used to generalize characteristic features of ECG recordings at multiple levels of granularity. As shown in Figure 4.5a, an ECG can be described in terms of the sequence of:

1) electric potentials providing a fine-grain representation of the heart's electrical activity,
2) distinct waveform deflections (P,Q,R,S,T waves) indicating the overall direction of depolarization and repolarization,
3) segments and intervals (PR interval, PR segment, QRS complex, QT interval, ST segment, RR interval) providing a coarse-grain representation that is temporally related to distinct phases of the cardiac conduction cycle (Figure 4.6a).

Considering the ECG sequence as a sequence of segments and intervals is appealing since limits can be set on these from which to diagnose deviations from normality. As a result, an ECG can provide valuable information about a patient's cardiac status and/or the adverse affects of acute and chronic conditions. For example, an ECG can aid in the diagnosis of an ischemic cardiac event in a patient presenting difficulty breathing and chest discomfort. In addition, an ECG may also be used to indicate how workload increases on the myocardium may be a compensatory response to chronic dysfunction of another body system (e.g. respiratory system). Given the dynamic nature of the heart's electrical activity over both short and long periods of time, the continuous interpretation of a patient's ECG is critical to assessing the severity of many effects (e.g., myocardial infarction, ventricular hypertrophy, or abnormal heart rhythms) as well as the monitoring of the heart's response to treatment [1]. However, recognition of the various statistical and structural properties of an ECG waveform, in relation to the heart's physiology, often requires *a priori* knowledge about the complex systems and processes that exist at multiple levels (genomic, proteomic, metabolic, cellular, etc) and give rise to the events of the cardiac cycle (i.e. isovolumetric ventricular contraction, ventricular ejection, isovolumetric relaxation, ventricular filling, arial systole). In subsequent sections, we outline how uncertainty about these underlying complex systems and processes, their interdependencies and domain-specific datastreams can be accounted for in online and unsupervised learning settings. To demonstrate how implicit knowledge about multi-level complex biological systems and processes can inform our decision-making ability in regards to the ECG waveform, we highlight the significance of the complex interdependencies that underly action potential conduction in cardiomyocytes (cardiac muscle cells) during the cardiac cycle (Figure 4.6).

Each iteration of the cardiac cycle, includes two phases: ventricular diastole (relaxation) and

[1]Note: An ECG surface recording does not measure the mechanical pumping ability of the heart.

ventricular systole (contraction). During diastole, the left and right ventricles relax, both atria contract, and blood is forced through the open tricuspid and mitral valves. The aortic and pulmonic valves are closed. During systole, the atria relax and fill with blood. The mitral and tricuspid valves are closed. As the ventricular pressure rises, the aortic and pulmonic valves are forced open. Both ventricles subsequently contract, and blood flows through the circulatory system. However, the heart cannot pump unless an electrical impulse is generated and transmitted.[2] Cardiac cells undergo cycles of depolarization and repolarization during impulse transmission. When at rest, cardiac cells are considered polarized (i.e. no electrical activity is occuring). The cell membranes of these cardiac cells separate different concentrations of ions, such as calcium, potassium and sodium (Ca^{2+}, K^{+}, Na^{+}), to create a more negative charge inside the cell (i.e resting potential). After an impulse is generated, ions cross the cell membrane and cause an action potential (i.e cell depolarization). When a cell becomes fully depolarized, it tries to return to its resting state in a process called repolarization. During this process, the electrical charges inside the cell reverse and return to normal. As shown in Figure 4.5, a temporal relation exists between the action potential curve (showing voltage changes during the five phases of the depolarization-polarization cycle) of an individual ventricular cardiomyocyte (Figure 4.5c) and the QRS complex and T wave of the ECG. This relation can be loosely generalized in terms of the ionic currents and channels that generate the distinct phases characterizing the action potential (Figure 4.5b) or scrutinized further in terms of the complex sequence of molecular events leading to cardiac muscle contraction/relaxation (Figure 4.7). After depolarization and repolarization occurs, the resulting electrical impulse travels through the heart along the cardiac conduction system. These impulses are initiated at the sinoatrial node and spread as a wave through the atrial muscle. The impulse travels through the internodal tracts and Bachmann's bundle to arrive at the atrioventricular node. When the impulse arrives at the atrioventricular node, there is a momentary delay that gives the atria time to squeeze blood into the ventricles before they fire. The impulse then continues to travel through the bundle of His, the bundle branches, and finally to the Purkinje fibers where it spreads through the walls of the ventricles. This sequence of electrical events occuring in cardiac cells throughout the conduction

[2] A small precentage of cardiac muscle cells possess the ability to spontaneously generate electric impulses. These cells are called autorhythmic cells. They include the pacemaker cells concentrated primarily in the sinoatrial node and the atrioventricular node.

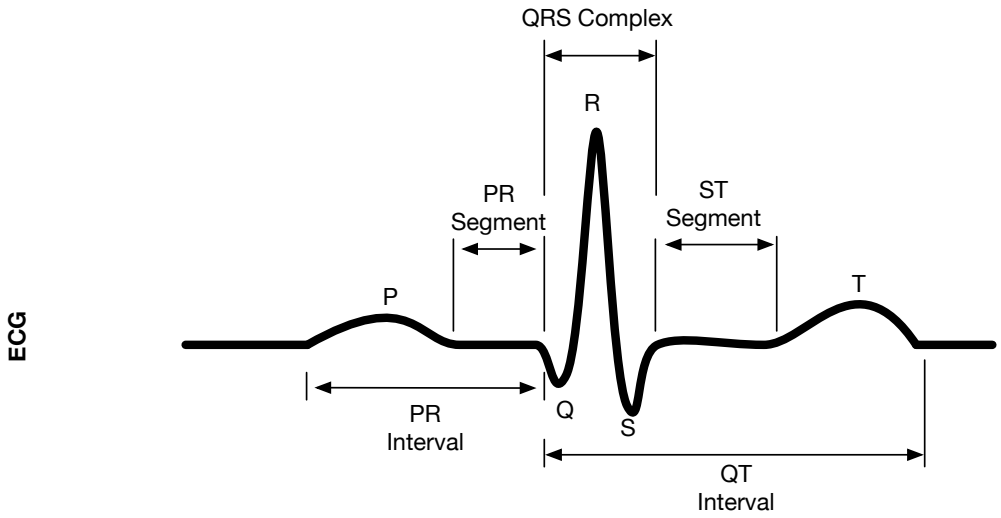
process, beginning with the depolarization of the atria (causing them to contract as a unit), and subsequently the depolarization of the ventricles (also causing them contract as a unit), is precisely shown in an ECG (Figure 4.6a). A comprehensive and dynamic model of the heart must include models of these various biological systems/processes along with their interdependencies. ECG data is one of many data types and sources that should be incorporated in such a comprehensive predictive model of cardiac activity. Thus, it is desirable for any learning algorithm to be able to learn and integrate models based on different data modalities (i.e. multi-modal learning). This process of multi-level model integration may enable researchers to quantify how and why events at the local and/or global level influence one another and possibly lead to insight about the effect and propagation of perturbations in biological pathways, as in the case of myocardial infarction.

In addition to the aforementioned challenge of multi-modal learning that we postulate as being necessary when building any predictive model of cardiac electrical activity, the non-stationary nature of cardiac signals and the effect of different noise sources (e.g. electrode contact noise, muscle movement artifacts) during the measurement process, can introduce additional uncertainty. This uncertainty is caused by discrepancies that exist between ECG recordings used during the learning process. Unlike other types of sequence data such as text which contains a natural segmentation defined by a distinct set of symbols (e.g. punctuation, end-of-line characters), many biomedical datastreams do not. In the case of an algorithm trying to learn a predictive model of cardiac electrical activity from an ECG data sequence, with limited *a priori* knowledge, it is difficult to know with absolute certainty at what moment in the cardiac cycle an ECG recording may have begun or ended. For example, without the assistance of expert knowledge or a data pre-processing procedure, we cannot ascertain whether all ECG data-sequences to be used during training of a predictive model, are of sufficient length and/or depict the same electrophysiological phenomenon we wish to model (i.e. Does each ECG recording begin with an electrical potential measurement of the same moment in the cardiac cycle? Does the sequence of electric potentials span the entire duration of a cardiac cycle?). In addition, regardless of whether the data is in an ideal form, it is desirable for a predictive model based on a given patient's ECG data to be resusable in prediction tasks when applied to a different patient's ECG datastream. Differences in the statistical properties between two patients cardiac profile make model reuse or transfer learning a challenging task. Assuming

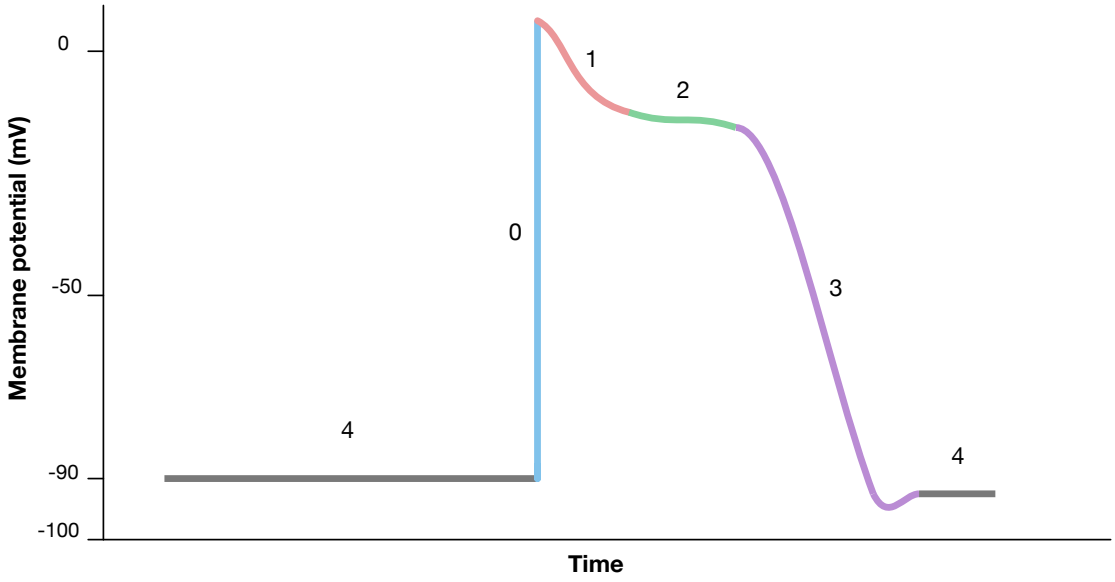
ideal data conditions, where all ECG data-sequences are of the same length and capture the same start and end events of a given electrophysiological phenomenon, two ECG data-sequences of two patients will not necessarily be similar. In such case, it is desirable for a learning algorithm to demonstrate the ability to generalize/reuse knowledge about the syntactic (structural) properties of previously learned patients' ECG data-sequences during learning of a new patient's data-sequence. Thus, the objective is to build general-purpose learning algorithms which can reuse and generalize knowledge about the distinct waveform deflections (P,Q,R,S,T) and their temporal dependencies to guide learning of a new ECG. These high-level characteristics (coarse-grain patterns) can be an alternative strategy to building a model from scratch in cases where limited ECG examples from a new patient are available for learning (i.e. one-shot learning).

4.2.2 Experiment Design

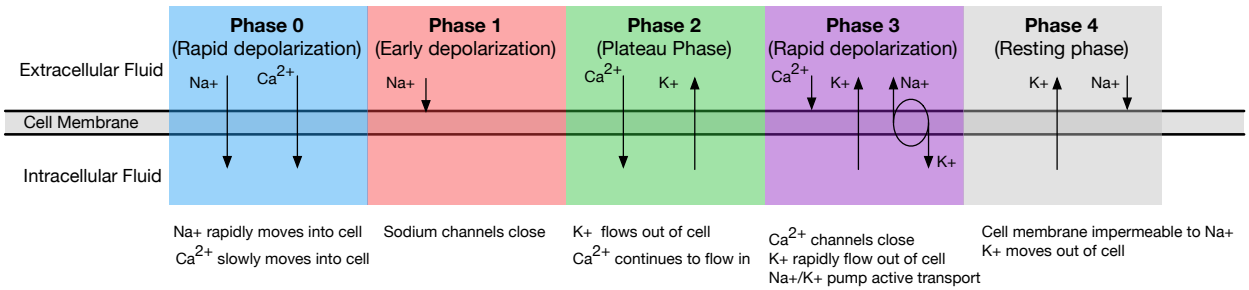
In this scenario, we evaluate online prediction ability in its most challenging form: after incremental exposure to one real-world complex biological system/process data sequence. As in the case of many other non-stationary complex biological system/process data sequences where sample size is limited, we aim to learn the spatio-temporal dynamics of a complex biological system/process incrementally without the explicit requirement of extensive retraining. The biomedical datastreams considered are discretized ECG data-sequences [138] derived from ECG data for 352 torso-surface sites across 4 human subjects with moderate to large myocardial infarctions. A single data-sequence sample represents a sequence of electric potentials captured over 1640 time-points from a given patient. Compared to most synthetic or idealized time-series datasets, these sequences display characteristics commonly observed in real-world biomedical datastreams, including data sparsity, multi-dimensionality and variable-order temporal dependencies. In a series of experiments, the SYNACX algorithm and various neural network models were evaluated under the constraints of one-shot learning. Specifically, in each experiment, a model was trained to ingest as input a training data-sequence and improve its output predictions as the data-sequence is processed in a feed-forward manner. All neural network models were implemented using Tensorflow [139]. Tensorflow is an open source software library, originally released by Google in 2015, that provides a framework with which deep learning models can be designed, built and trained.



(a) Simplified sketch of surface electrocardiography (ECG) recording

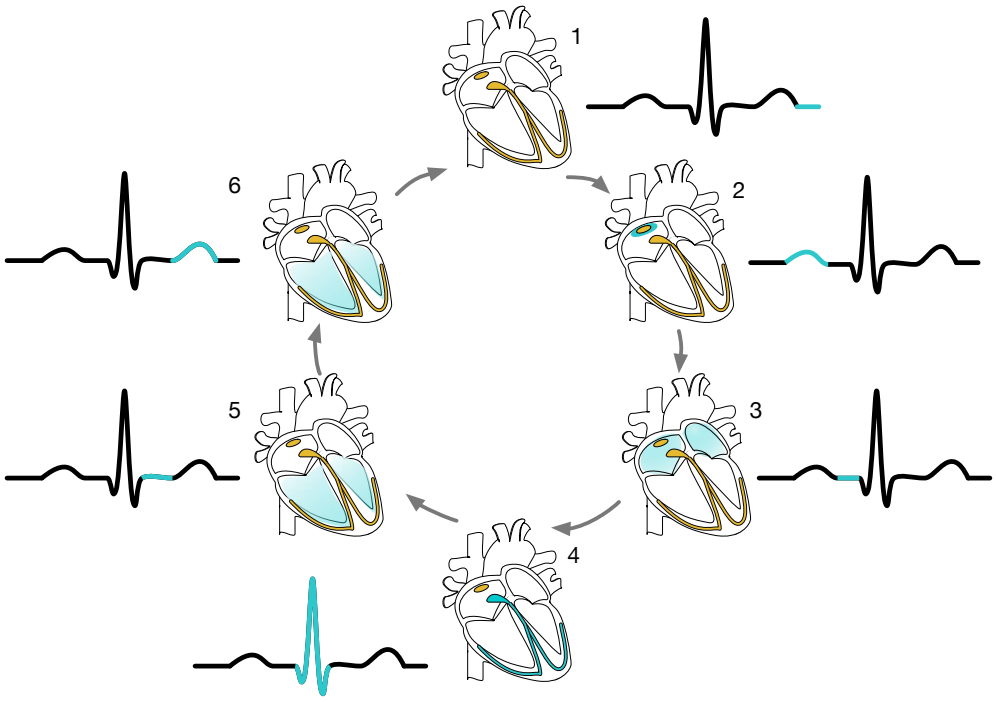


(b) Action potential of a cardiomyocyte.

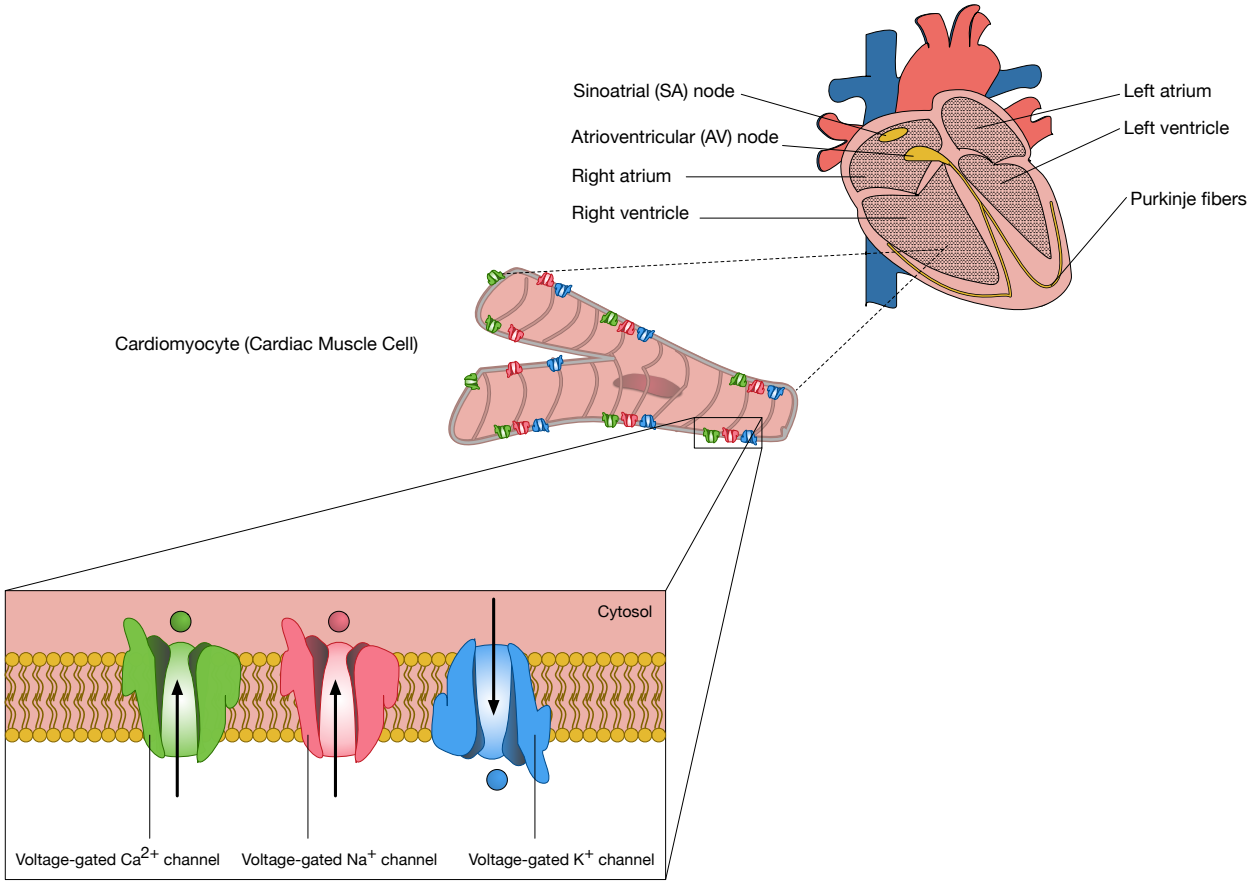


(c) Ionic currents and channels that generate the distinct phases characterizing the action potential in a cardiomyocyte.

FIGURE 4.5 Relationship between an individual ventricular cardiomyocyte action potential and an ECG. The action potential can be decomposed into five unique phases (0-4). The cardiac action actional and its phases are temporally correlated to the QRS complex and T wave of the ECG. The ECG also illustrates the anatomical orientation of the wave of depolarization through the heart and the location of recording electrodes.



(a) Cardiac Conduction Cycle with corresponding ECG tracings



(b) Multi-Level Perspective of Cardiac Muscle Cells

FIGURE 4.6 Illustration of the relationships between ECG waveform deflections and heart physiology during conduction.

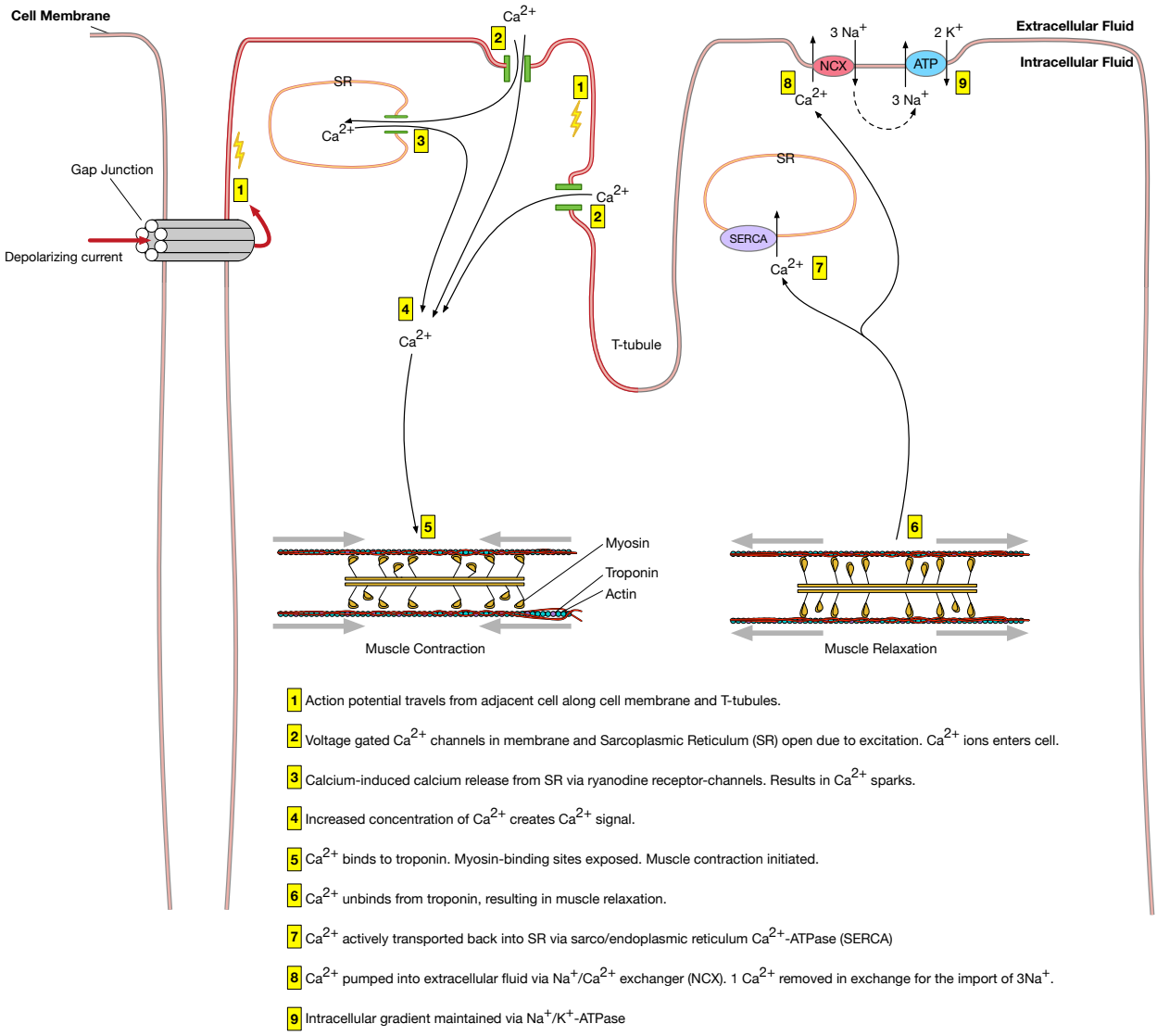


FIGURE 4.7 Sequence of molecular events leading to contraction and relaxation of a cardiomyocyte

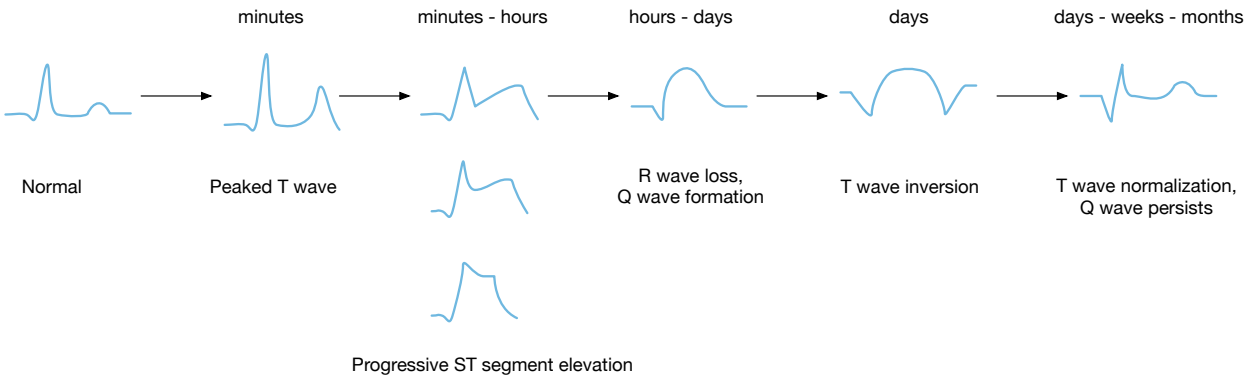


FIGURE 4.8 ECG Evolution of Mydocardial Infarction

In general, Tensorflow is a Python library that expresses arbitrary computation as a graph of data flows, where nodes depict mathematical operations and edges represent data (in the form of tensors) transmitted between nodes. Among the most appealing attributes offered by Tensorflow and the utilization of tensors in building neural networks is its computational efficiency by means of exploiting the acceleration of parallel tensor operations afforded by GPU hardware.

4.2.2.1 Tasks Evaluated

The set of online predictive modeling experiments conducted for this scenario can be divided into three distinct tasks. In Task 1a, models with no prior training are constructed from a single data-sequence from Patient 1, S_1. Models are trained on the first 10% of S_1 and subsequently tested on the latter 90%. In Task 1b, the top 5 models generated during Task 1a experiments are evaluated on a repeat exposure of S_1 under the same train/test ratio (10/90). In Task 1c, models with no prior training are initially trained on the first 99% of S_1 and subjected to further training on the first 10% of a new data sequence from Patient 2, S_2, and then tested on the latter 90%.

4.2.3 Results/Discussion

For Task 1a, we evaluated the MLP, RNN and LSTM neural network architectures using various parameter configurations to provide breadth and depth to our experiments. To begin, architectures with only one hidden layer and of variable size (1,10, 50, 100, 150 units) were evaluated. In addition, each neural network configuration was applied with one of three optimization procedures (Stochastic Gradient Descent (SGD) [40] , Adaptive Moment Estimation (ADAM) [41], Root Mean Square Propagation (RMSPROP) [42]). ADAM and RMSPROP are popular variants of the standard stochastic gradient descent algoritm which also adapt the learning rate for each parameter. Each optimization algorithm was implemented in accordance with their original published form, with an initial learning rate $\eta = 0.001$ used across all methods and decay variables $\beta_1 = 0.9$, $\beta_2 = 0.999$ used for the ADAM method. The performance results for each of these 1-hidden layer neural network models is given in Tables 4.1, 4.2, 4.3. Figure 4.9 depicts the performance of a SYNACX-derived model relative to these neural network models. This plot illustrates the problem of 'model selection' that exists in neural network models but avoided in SYNACX due

		SGD		ADAM		RMSPROP	
Model	**# hidden units**	**Training MSE**	**Test MSE**	**Train MSE**	**Test MSE**	**Train MSE**	**Test MSE**
LSTM_1L_1	1	0.03	0.53	2.34	3.07	1.05	1.67
LSTM_1L_2	10	0.01	0.51	0.01	0.28	0.01	0.29
LSTM_1L_3	50	0.02	0.72	0.01	0.19	0.01	0.19
LSTM_1L_4	100	0.02	0.64	0.01	0.16	0.01	0.17
LSTM_1L_5	150	0.02	0.63	0.01	0.14	0.02	0.17

TABLE 4.1 Task 1a Performance for 1-Hidden Layer LSTM Networks

its use of Bayesian nonparametric modeling methods which are less dependent on specifying the model complexity *a priori*. Using the top five single hidden layer neural network models and the SYNACX model, listed in Table 4.4, we evaluate each model's prediction ability in Task 1b (Table 4.5). Task 1b simply serves as a sanity check against overfitting. In order to evaluate the effect of increasing neural network depth on model performance, we repeat experiments for Tasks 1a and 1b with models consisting of 2 hidden layers. Each neural network model was once again constructed with hidden layers consisting of a variable number of hidden units. Given the negligible difference in results between optimization algorithms for experiments with 1-layer models, only the ADAM optimization procedure was implemented in these 2-hidden layer neural network models. In addition, each 2-hidden layer model was evaluated with an implementation that included the use of the dropout algorithm [140]. Similar to the previous plot depicting the performance of a SYNACX-derived model relative to neural network models with a single hidden layer (Figure 4.9), Figure 4.10 further illustrates how the problem of 'model selection' in neural network models persists, despite adding more layers to create a deeper neural network model. Whereas Tasks 1a and 1b both evaluate the ability to perform prediction in the presence of data and model uncertainty, Task 1c begins our evaluation of the the benefits of utilizing a modeling formalism, such as simplicial grammar, with which a new pattern class (Patient ECG) can be modeled and predicted by reusing knowledge learned from other pattern classes. In Tables 4.11 and 4.12, Task 1c performance results for the top five neural network models with 1-hidden layer and 2-hidden layer architectures are shown, respectively. Figure 4.12 illustrates how the performance of the SYNACX model is on the same order of magnitude to neural network models with the best optimized model configuration for Task 1c.

The number of machine learning algorithms being developed and applied to computational

		SGD		ADAM		RMSPROP	
Model	**# hidden units**	**Training MSE**	**Test MSE**	**Train MSE**	**Test MSE**	**Train MSE**	**Test MSE**
RNN_1L_1	1	0.03	0.91	30	32.74	25.65	28.23
RNN_1L_2	10	0.01	0.14	0.01	0.13	0	0.12
RNN_1L_3	50	0.01	0.41	0.01	0.15	0.03	0.15
RNN_1L_4	100	0.01	0.32	0.01	0.13	0.17	0.27
RNN_1L_5	150	0.01	0.39	0.01	0.13	0.01	0.11

TABLE 4.2 Task 1a Performance for 1-Hidden Layer RNNs

		SGD		ADAM		RMSPROP	
Model	**# hidden units**	**Training MSE**	**Test MSE**	**Train MSE**	**Test MSE**	**Train MSE**	**Test MSE**
MLP_1L_1	1	0.02	0.53	0.02	0.5	0.02	0.5
MLP_1L_2	10	0.04	1.19	0.03	1.07	0.03	0.98
MLP_1L_3	50	0.02	0.78	0	0.28	0	0.23
MLP_1L_4	100	0.01	0.41	0	0.25	0	0.21
MLP_1L_5	150	0.01	0.47	0	0.29	0	0.23

TABLE 4.3 Task 1a Performance for 1-Hidden Layer MLP Networks

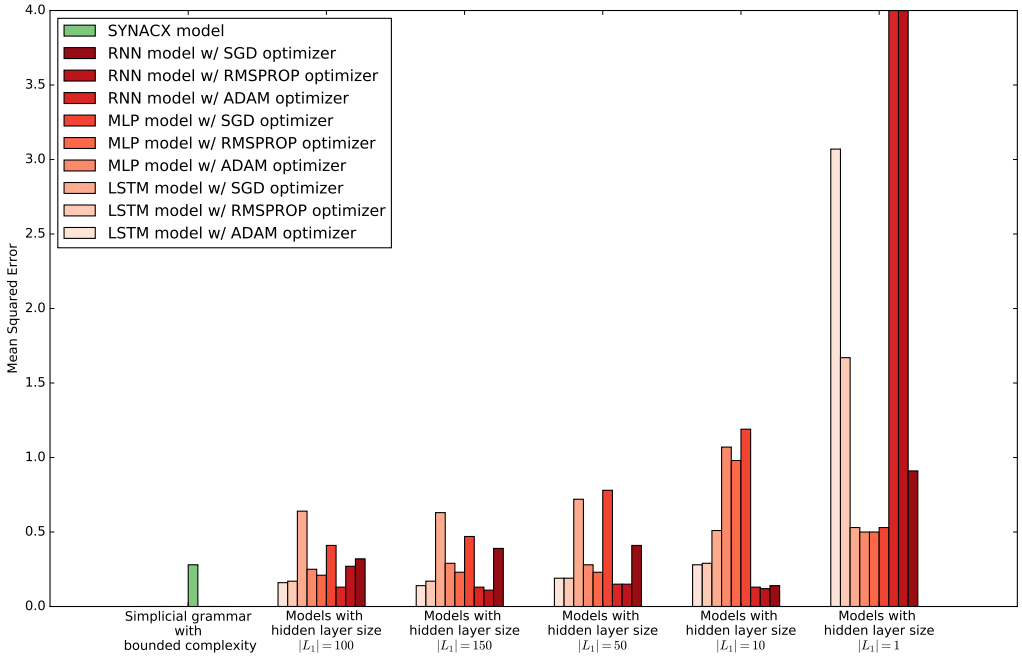


FIGURE 4.9 Comparing performance in Task1a for a SYNACX-derived model relative to neural network models. The neural network models that were evaluated use architectures consisting of a single hidden layer across various predefined sizes.

Model	# hidden units	Train MSE	Test MSE
RNN_1L_2-ADAM	10	0.01	0.13
RNN_1L_4-ADAM	100	0.01	0.13
RNN_1L_5-ADAM	150	0.01	0.13
LSTM_1L_5-ADAM	150	0.01	0.14
LSTM_1L_4-ADAM	100	0.01	0.16
SYNACX		0.02	0.28

TABLE 4.4 Task 1a Performance comparison of top 1-Hidden Layer Architectures vs. SYNACX

Model	# hidden units	Train MSE	Test MSE
RNN_1L_2-ADAM	10	0.01	**0.08**
RNN_1L_4-ADAM	100	0	**0.08**
RNN_1L_5-ADAM	150	0.06	0.13
LSTM_1L_5-ADAM	150	0.07	0.16
LSTM_1L_4-ADAM	100	0.07	0.17
SYNACX		0.02	0.11

TABLE 4.5 Task 1b Performance comparison of top 1-Hidden Layer Architectures vs. SYNACX

			ADAM	
Model	Layer1 # hidden units	Layer2 # hidden units	Train MSE	Test MSE
LSTM_2L_1	10	10	0.01	0.31
LSTM_2L_2	10	150	0.02	0.25
LSTM_2L_3	150	10	0.02	0.23
LSTM_2L_4	150	150	0.01	0.13
LSTM_2L_5_DO	10	10	0.17	0.64
LSTM_2L_6_DO	10	150	0.05	0.37
LSTM_2L_7_DO	150	10	0.47	0.8
LSTM_2L_8_DO	150	150	0.03	0.15

TABLE 4.6 Task 1a Performance for 2-Hidden Layer LSTM Networks

			ADAM	
Model	Layer1 # hidden units	Layer2 # hidden units	Train MSE	Test MSE
RNN_2L_1	10	10	0.03	0.66
RNN_2L_2	10	150	0.01	0.2
RNN_2L_3	150	10	0.01	0.29
RNN_2L_4	150	150	0.01	0.14
RNN_2L_5_DO	10	10	0.76	1.18
RNN_2L_6_DO	10	150	0.01	0.2
RNN_2L_7_DO	150	10	0.77	0.96
RNN_2L_8_DO	150	150	0.4	0.56

TABLE 4.7 Task 1a Performance for 2-Hidden Layer RNNs

			ADAM	
Model	**Layer1 # hidden units**	**Layer2 # hidden units**	**Train MSE**	**Test MSE**
MLP_2L_1	10	10	0	0.46
MLP_2L_2	10	150	0.01	0.54
MLP_2L_3	150	10	0	0.38
MLP_2L_4	150	150	0.01	0.54
MLP_2L_5_DO	10	10	0	0.38
MLP_2L_6_DO	10	150	0	0.44
MLP_2L_7_DO	150	10	0	0.28
MLP_2L_8_DO	150	150	0.01	0.5

TABLE 4.8 Task 1a Performance for 2-Hidden Layer MLP Networks

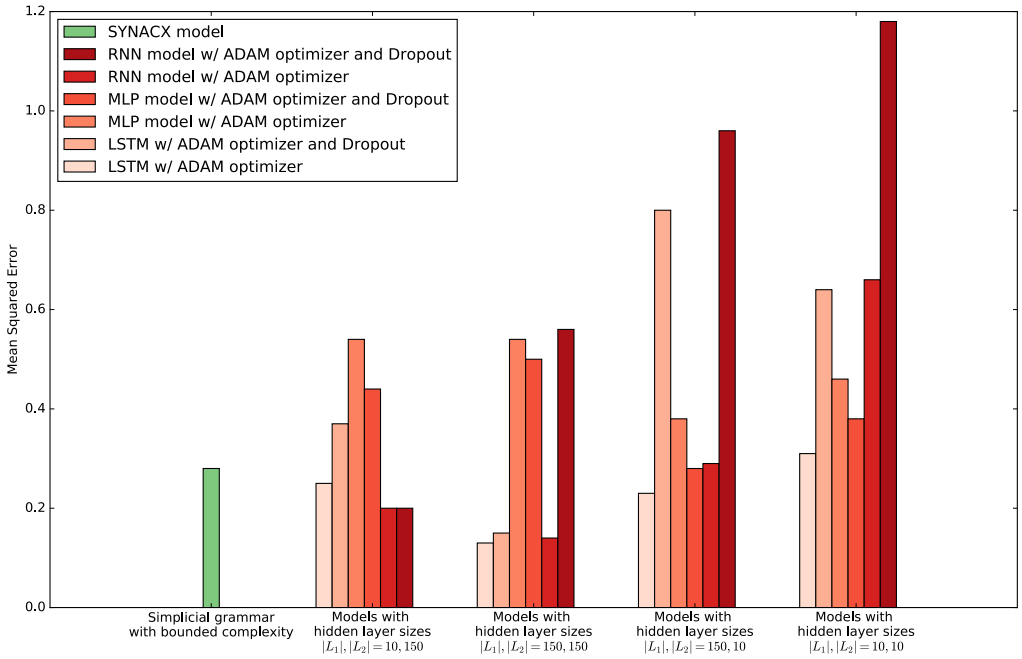


FIGURE 4.10 Comparing performance in Task1a for a SYNACX-derived model relative to neural network models. The neural network models that were evaluated use architectures consisting of two hidden layers across various predefined sizes.

Model	**Layer1 # hidden units**	**Layer2 # hidden units**	**Train MSE**	**Test MSE**
LSTM_2L_4-ADAM	150	150	0.01	0.13
RNN_2L_4-ADAM	150	150	0.01	0.14
LSTM_2L_8_DO-ADAM	150	150	0.03	0.15
RNN_2L_6_DO-ADAM	10	150	0.01	0.2
RNN_2L_2-ADAM	10	150	0.01	0.2
SYNACX			0.02	0.28

TABLE 4.9 Task 1a Performance comparison of top 2-Hidden Layer Architectures vs. SYNACX

Model	Layer1 # hidden units	Layer2 # hidden units	Train MSE	Test MSE
LSTM_2L_4-ADAM	150	150	0.01	0.1
RNN_2L_4-ADAM	150	150	0	0.09
LSTM_2L_8_DO-ADAM	150	150	0.07	0.17
RNN_2L_6_DO-ADAM	10	150	0.2	0.3
RNN_2L_2-ADAM	10	150	0.01	**0.08**
SYNACX			0.02	0.11

TABLE 4.10 Task 1b Performance comparison of top 2-Hidden Layer Architectures vs. SYNACX

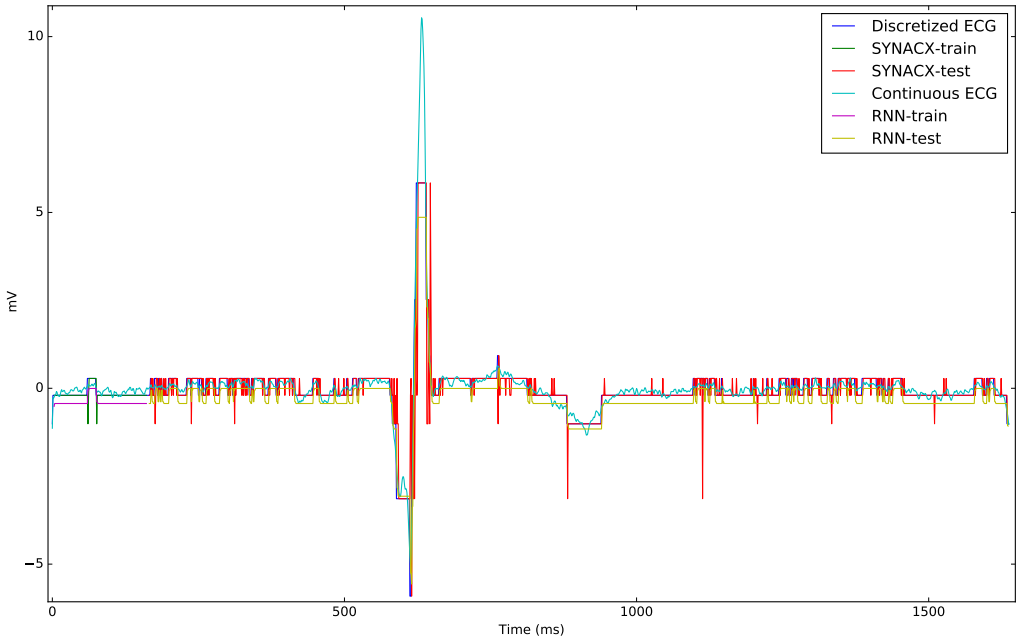


FIGURE 4.11 Overall Top-Performing Neural Architecture vs. SYNACX for Task 1a

Model	# hidden units	Train MSE	Test MSE
RNN_1L_2-ADAM	10	0.01	0.07
RNN_1L_4-ADAM	100	0.01	0.09
RNN_1L_5-ADAM	150	0.01	**0.05**
LSTM_1L_5-ADAM	150	0.01	0.09
LSTM_1L_4-ADAM	100	0.01	0.09
SYNACX		0.01	0.23

TABLE 4.11 Task 1c Performance comparison of top 1-Hidden Layer Architectures vs. SYNACX

Model	Layer1 # hidden units	Layer2 # hidden units	Train MSE	Test MSE
LSTM_2L_4-ADAM	150	150	0.01	0.11
RNN_2L_4-ADAM	150	150	0.03	0.14
LSTM_2L_8_DO-ADAM	150	150	0.03	0.19
RNN_2L_6_DO-ADAM	10	150	0.01	0.17
RNN_2L_2-ADAM	10	150	0.01	**0.08**
SYNACX			0.01	0.23

TABLE 4.12 Task 1c Performance comparison of top 2-Hidden Layer Architectures vs. SYNACX

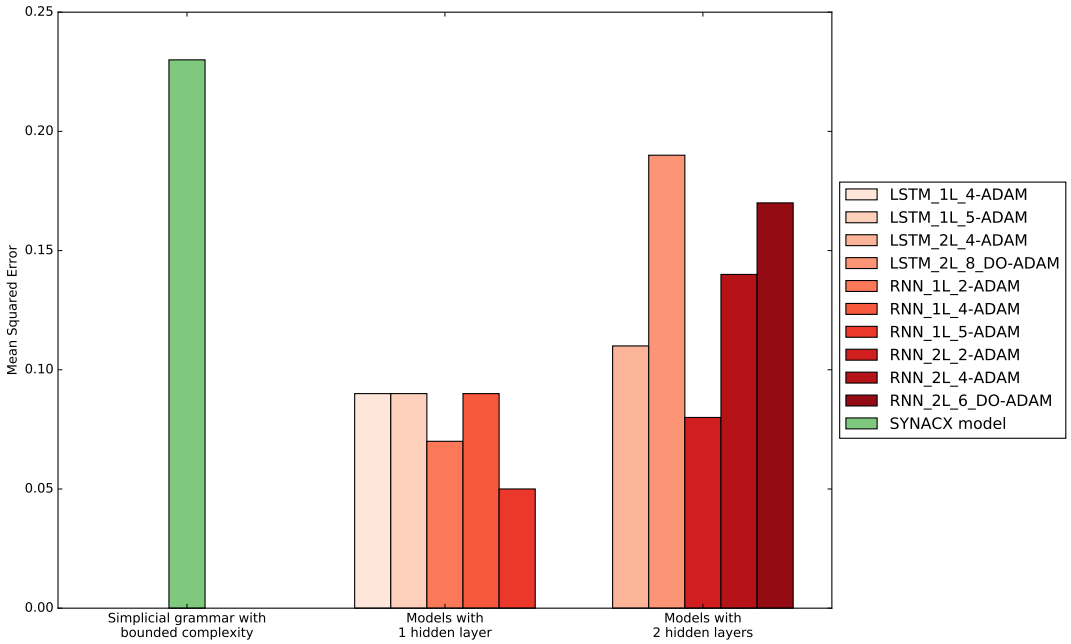


FIGURE 4.12 Comparing performance in Task1c for a SYNACX-derived model relative to the top performing neural network models.

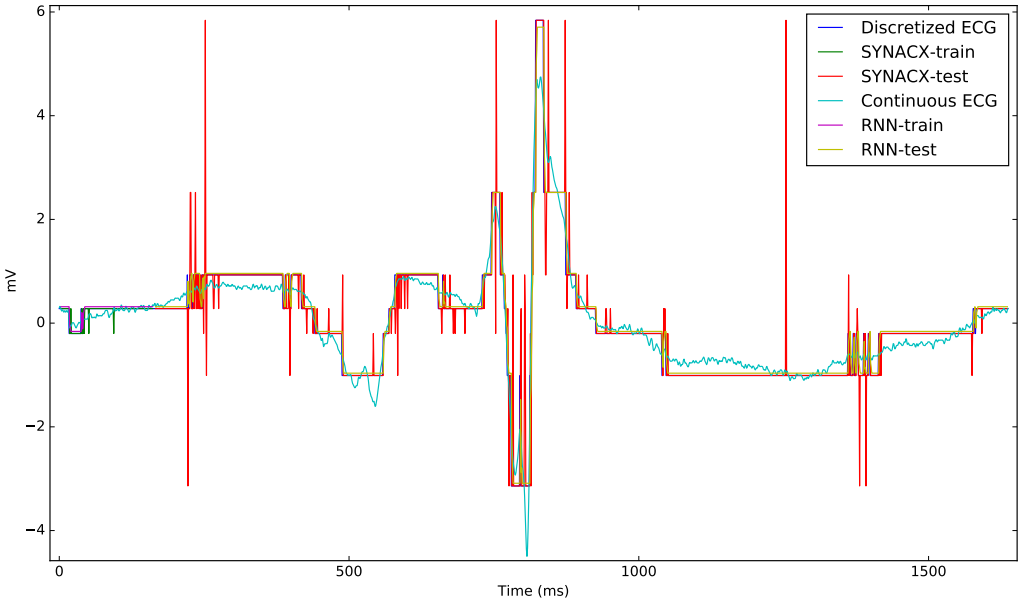


FIGURE 4.13 Overall Top-Performing Neural Architecture vs. SYNACX for Task 1c

modeling tasks continues to grow over time. This growth is partially due to advances in the technology that have increased the abundance and availability of large-scale datasets. As a result, algorithms, new and old, have found use in a wide range of new domain applications. The set of models constructed in our experiments, using LSTMs, RNNs and MLPs, represent only a small subset of algorithms applicable to the designated tasks. We acknowledge that each neural-network architecture can exhibit superior performance, given additional data and further optimization. However, the purpose of the experiments performed in this chapter is not to determine whether the SYNACX algorithm matches or exceeds the performance outcomes of state-of-the-art methods optimized to perform a specific task. The primary focus of these preliminary studies is to provide a holistic evaluation of the functionality of the SYNACX framework, as a proof of concept, and to highlight the intrinsic difficulties associated with learning directly from data of a complex biological system/process. While not immediately evident in the results shown in Tables 4.1-4.3, training on only 10% of the data reflects two inherent challenges that any machine learning model must anticipate when trying to learn from data of a complex biological system/process. In the case of sampling from novel biomedical datastreams, the challenge is not being able to control the *length* or the content of the data-subsequences with which a machine learning model is trained. Put simply,

without *a priori* knowledge and/or the use of pre-processing methods, it is difficult to anticipate

1) *if* a given data-sequence contains sufficient information about the phenomenon we wish to model.
2) *where* in a given data-sequence pertinent information about the phenomenon we wish to model exists.

As can be seen in Tables 4.1, 4.2 and 4.3, the SYNACX algorithm performs generally well against neural-network architectures implemented with standard stochastic gradient descent (SGD) optimization. However, the performance of these neural network models begins to surpass that of SYNACX as we begin to manipulate the model parameters, make application-specific assumptions, and implement more modern optimization procedures. For example, when working with standard neural-network architectures, several implicit and explicit assumptions regarding the model parameters (i.e. number of hidden layers, type of neuron used, number of neurons used per layer, choice of loss function), input data (dimension, length, distribution) and choice of optimization algorithm, must be made during its design, and before it is implemented. These assumptions induce a learning bias, based on a user's judgement and prior knowledge, that ultimately affects how well a model can learn the target function and generalize beyond training data. In the case of time-series or sequence data, algorithms are often designed and/or optimized to work with data of a known structure (stationary or nonstationary), length, and dimension. When the data is variable in length, additional pre-processing methods such as data-padding are utilized so as to fulfill the assumptions used by the algorithm. The result of these workarounds is a trained model that is superior to other models for a given task but less intuitive or true to the system from which its data was sampled. Although we may find and/or train a neural-network prediction model with very high accuracy and precision from a given dataset, if the system from which data is sampled undergoes an unforeseen shift in behavior, the trained model may no longer be the most optimal. Thus, it is desirable that the underlying neural-network model be adaptive, such that it can anticipate changes in observed spatio-temporal patterns (Figure 4.8) and the occurence of system perturbations and black swan events [141, 142] in a complex biological system/process.

4.3 Scenario 2: Modeling cell morphology dynamics from live-cell imaging data

In this section we evaluate the applicability of the SYNACX framework and neural network architectures in learning predictive models of cell morphology features from time-lapse microscopy image sequences.

4.3.1 Motivation

The purpose of evaluating various predictive models in this scenario is to contribute a new method by which to learn and analyze the spatio-temporal dynamics of single-cell properties directly from empirical data. Despite the substantial knowledge about cellular dynamics obtained from bulk population-level studies, the heterogeneous nature of many cell-lines requires single-cell profiling techniques in order to quantify the dynamic processes from which further insights can be derived. To date, many single-cell approaches have been used in spatio-temporal dynamic studies, including single-cell RNA-seq for measuring gene expression [143], sorting methods such as Fluorescence-activated cell sorting (FACS) and single cell mass cytometry (CyTOF) [144] and microfluidic chip-electrospray ionization mass spectrometry for protein and metabolite analysis at the single-cell level [145]. Since each of these modalities often introduces its own novel data type, it is desirable for their respective predictive models to be built in an integrative and general-purpose manner. To explore this further, we evaluate the feasiblity of using the SYNACX framework and neural network architectures in building predictive models of single-cell dynamics using pre-processed image feature data obtained from live-cell imaging experiments.

Advancements in time-lapse microscopy have enabled researchers to visualize and evaluate the variety of complex patterns governing dynamic cellular processes in real-time [146]. One such cellular process, known as mitosis (cell division), is the process by which genetic material of a eukaryotic cell is equally distributed between its descendants through nuclear division, resulting in the birth of daughter cells. Cellular events such as mitosis are frequently accompanied by changes in cell morphology on a range of spatial and temporal scales, which can be recorded in the form of

image data via a time-lapse microscopy method known as live-cell imaging. Live-cell imaging is an essential tool for deciphering the dynamics of individual cells and cell populations by capturing and correlating cell morphology and gene-expression profiles at multiple scales of resolution. As new data modalities become available and models derived from their respective datasets are generated, it is desirable to integrate them into a single comprehensive model. Insights obtained from these models of cell morphology can be used to enhance current models of cellular processes, leading to integrated cellular/gene-expression models. For example, in Figure 4.14 we illustrate the sequence of events involved in actin based endothelial cell migration. Cell motility experiments performed on a live-cell imaging platform allow researchers to construct computational models [146–149] describing changes in morphology during the migration process (Figure 4.14.A). It is desirable for such models to be subsequently integrated with other quantitative models describing its major signaling events (Figure 4.14.C) using new empirical data or information found in published literature [150]. Our motivation reflects that of the previous scenario, where the objective is to provide a universal and intuitive modeling formalism in which probabilistic generative models of a complex biological system/process can be derived from multiple data modalities. This can be achieved in an unsupervised and/or semi-supervised manner via SYNACX's inductive inference algorithm in the presence of sequence data or via manual rule creation by domain experts, respectively.

4.3.2 Experiment Design

4.3.2.1 Cell Line

The data used consists of time-lapse microscopy images of MDA-MB231 human breast cancer epithelial cells and NCTC clone 929 [L cell, L929, derivative of Strain L] mouse fibroblast cells. Both cell lines were separately maintained in Dulbecco's Modified Eagle Medium (DMEM, Life Technologies) supplemented with 10% (v/v) fetal bovine serum (FBS, Hyclone), 100 units/ml penicillin and 100 μ/ml streptomycin (Life Technologies). Cells were maintained in an air incubator at 5% CO2 and 37°C. Twenty four hours prior to imaging, cells were seeded and allowed to adhere in a 6-well cell culture plate (Corning) at a density of $2\text{x}10^4$ cells/well.

4.3.2.2 Large-Scale Digital Cell Analysis System

The Large-Scale Digital Cell Analysis System (LSDCAS) is an automated live-cell imaging system capable of examining thousands of living cells for up to one month during a single experiment [147]. LSDCAS is designed for the quantitative study of cell culture populations grown under conditions that mimic a traditional incubator used in routine biochemical/molecular in vitro experiments. The system is comprised of hardware for controlled environment image acquisition that allows for unperturbed cell growth. The main components include an Olympus IX-71 inverted phase microscope, a Hamamatsu ORCA-285 cooled CCD camera, and an Okolab stage incubator. The hardware is controlled with software designed in-house to direct stage movement, image autofocus, and modification of optical parameters. The input data acquired through this framework consists of a set of time-lapse image sequences from multiple microscope fields per experimental sample. An example image sequence for a single microscope field is shown in Figure 4.15. LSDCAS can segment and track multiple single-cells, as they grow and divide, to generate individual cell spatial trajectories as a function of time. To determine these individual cell trajectories, advanced image processing and machine learning methods are used to determine the borders around individual cells and cell clusters in each frame of an image sequence. A feature of the resulting cell borders, the border centroid, is then used to track cell motion frame-to-frame. Subsequent statistical analysis using the spatial and temporal features extracted from the large amount of cell border data in a given data set, provide a description of the total distribution of cell speeds in the sample population, as well as a temporal description of the change in mean cell motility as a function of experiment time. In a given experiment, 20 microscope fields are acquired for each sample, at 20 min interframe intervals, for three days. With 215 frames acquired per microscope field, each sample consists of approximately 4300 frames. Thus, the resultant dataset contains approximately 4,000 cell borders associated with 200 single cells explictly modeled and analyzed throughout the duration of a given experiment.

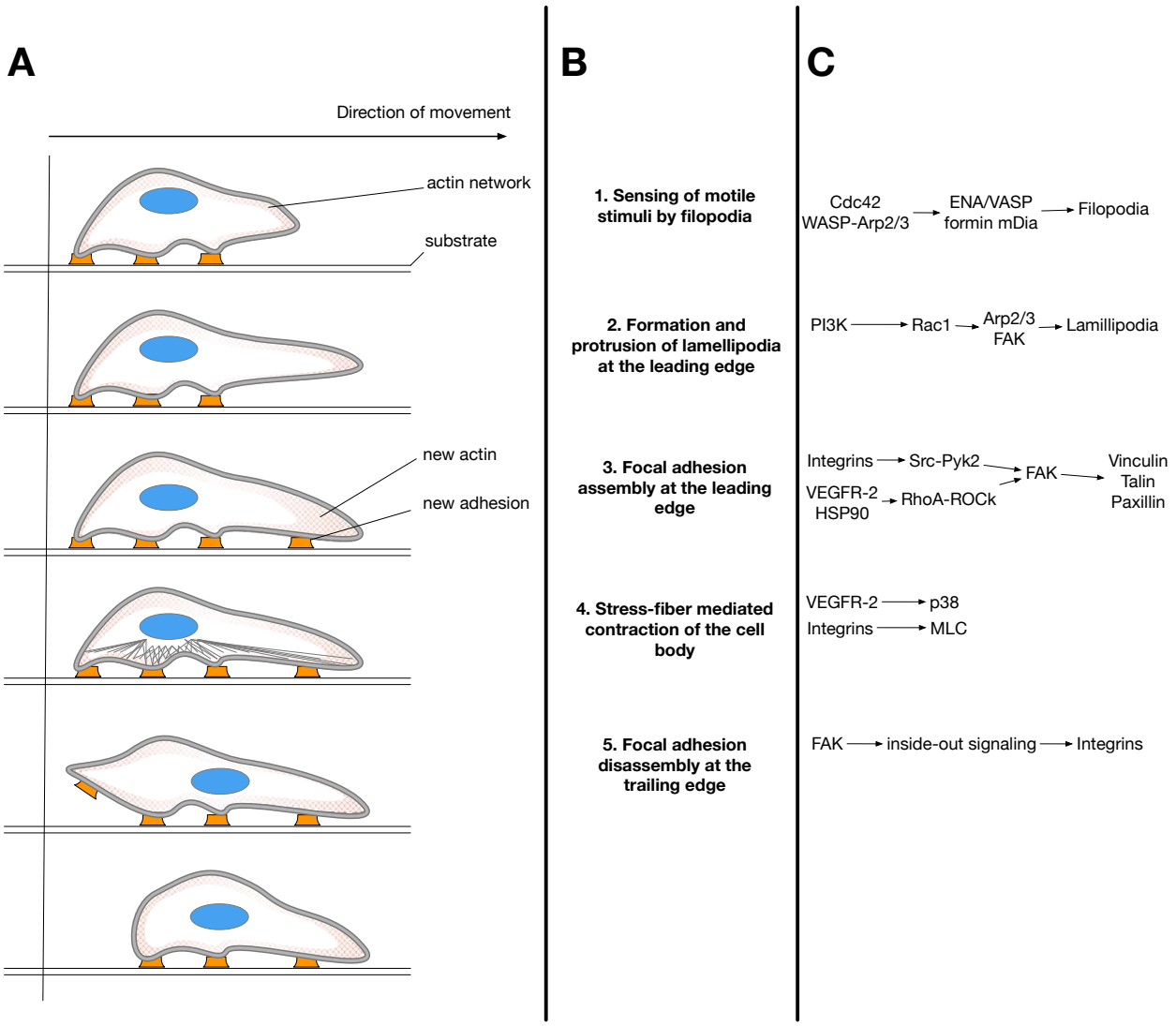
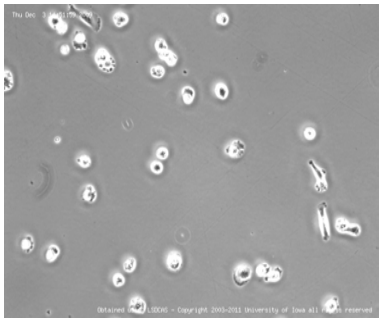
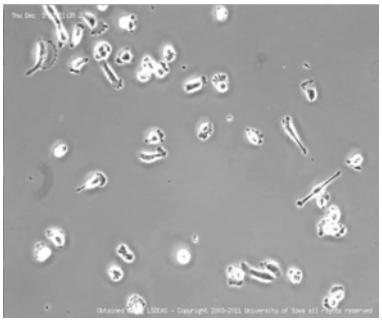


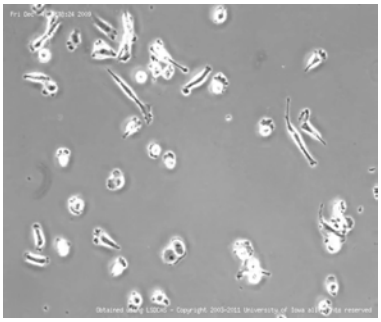
FIGURE 4.14 Overview of endothelial cell migration. A, The sequence of changes in morphology observed during the migration process. The sequence of steps and signaling events typically associated with these changes in morphology are indicated in B and C, respectively.



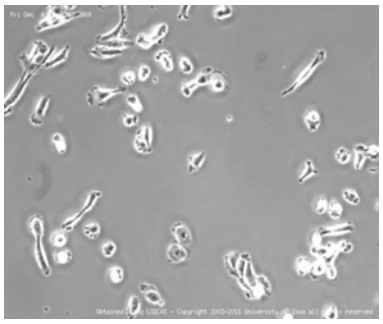
(a) Cells within microscope field of view at time t_0



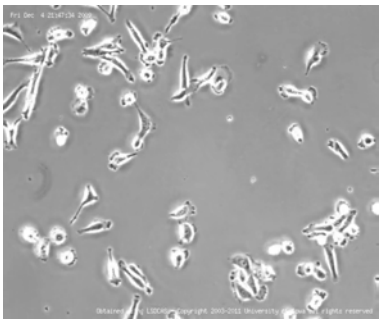
(b) Cells within microscope field of view at time t_1



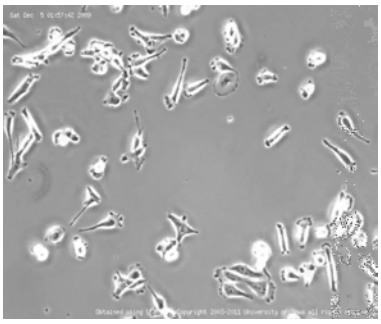
(c) Cells within microscope field of view at time t_2



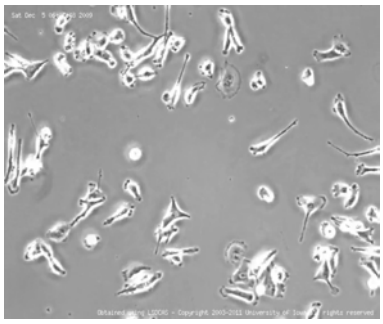
(d) Cells within microscope field of view at time t_3



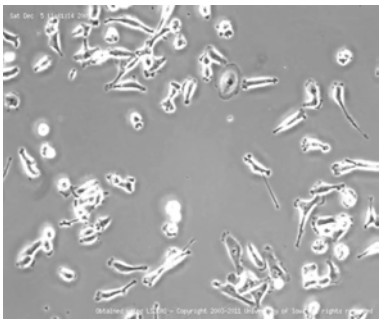
(e) Cells within microscope field of view at time t_4



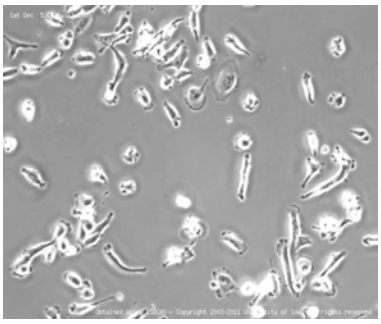
(f) Cells within microscope field of view at time t_5



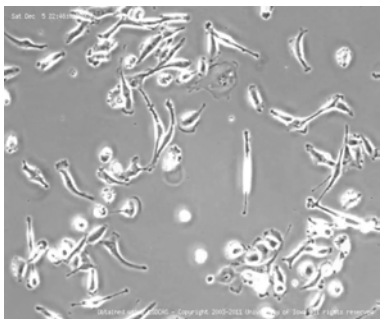
(g) Cells within microscope field of view at time t_6



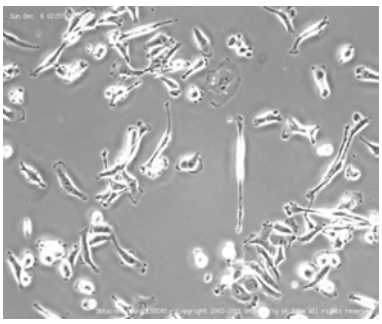
(h) Cells within microscope field of view at time t_7



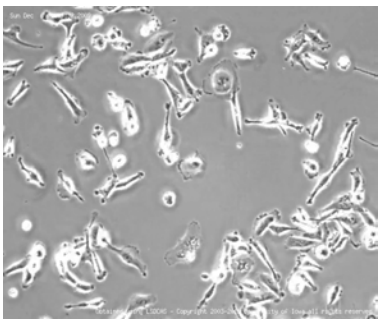
(i) Cells within microscope field of view at time t_8



(j) Cells within microscope field of view at time t_9



(k) Cells within microscope field of view at time t_{10}



(l) Cells within microscope field of view at time t_{11}

FIGURE 4.15 Time-lapse image sequence captured from single field within cell culture well

Discrete Cellular Event	Spatio-Temporal Characterization
Late Interphase	Cell is spread out over cell-culture plate surface, internal structure is visible. Cell has grown to its largest size before mitosis. Identified based on temporal proximity to beginning of mitosis.
Metaphase	Chromosomes are aligned at metaphase plate; cell has become round and borders are bright.
Anaphase/Early Telophase	Chromosomes have migrated to opposite poles, nuclear envelope may be reforming but cytokinesis not yet evident.
Late Telophase/Cytokinesis	Cytoplasmic division is complete or nearing completion; two daughter cells are now visible
Early Interphase (Gl)	Identified based on temporal proximity to end of mitosis. Internal cellular structure becoming more visible; cells begin to flatten out over plate surface and growing in size.

TABLE 4.13 Discrete cellular events associated with mitosis

4.3.2.3 Morphology Discretization

A single cell's progression through its life cycle can be observed in a sequence of time-lapse microscopy images. Morphological and optical changes in the cell can be used to determine the approximate location of the cell in the cell cycle. Using adherent cell lines and live-cell imaging techniques, cells can be visually classified throughout the duration of their cell cycle via a finite set of commonly observed morphological changes (discrete cellular events). An adherent cell can typically be observed spread out over a cell-culture plate surface, thus allowing visualization of its internal structure, most notably the cell nucleus. At the time of division, as a cell becomes more compact and spherical in shape a resultant change in optical properties is observed in digital image sequences in the form increased pixel intensity. Table 4.13 provides a subset of discrete cellular events related to mitosis that can be visibly detected using geometric and optical changes in a time-lapse microscopy image sequence. These discrete events are subsequently used to characterize the morphology of a cell at a given moment in time. Following the LSDCAS data acquisition process [146], a dataset, M, describing the morphology of imaged cells is generated. This dataset consists of data sequences providing a quantitative description of each cell's morphology over time (Figure 4.16) in terms of its optical and geometric properties. An element, $m \in M$, of a single data sequence provides a 4-dimensional representation of a distinct cell at a given time during imaging. In addition to its timestamp, this data element describes the morphology of a cell in terms of the parameters: Mean Intensity (i.e. measurement of cell brighthness), Area and Shape Factor (i.e. measurement of cell's spherical shape). For each morphology parameter, a vector quantization procedure is performed in order to obtain a finite *alphabet* of discrete morphological

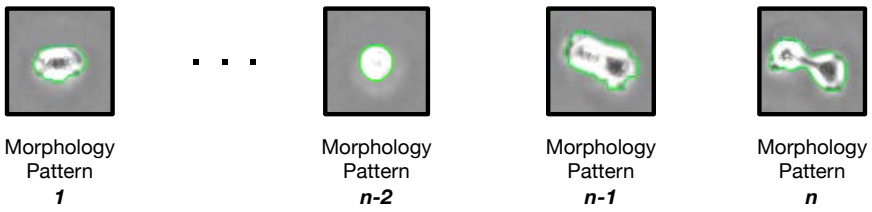


FIGURE 4.16 Distinct morphology shapes captured from live-cell imaging data

states. This alphabet is subsequently used to discretize the spatio-temporal trajectory of a given cell into a discrete sequence of morphological events. The collection of such sequences for a given experiment serves as the dataset used in our predictive modeling experiments.

4.3.2.4 Tasks Evaluated

We continue our evaluation of the SYNACX framework in online predictive modeling tasks by considering discrete sequences of morphology data objects derived from live-cell imaging experiments. In addition to generating predictive models under the constraints of one-shot learning as was the case in the previous scenario, Tasks 2a-2d evaluate the difference in prediction performance under one-shot learning and incremental learning conditions. In Tasks 2a and 2b, predictive models are generated by a given neural-network architecture under various model parameter configurations (i.e. number of hidden layers, size of hidden layer) and training conditions (i.e. size of training/prediction datasets). In Task 2a predictive models, training is performed (90/10 train/prediction) on two consecutive training datasets D_1,D_2, each of which contains variable-length data sequences for N distinct MDA-MB231 cells derived from time-lapse microscopy image sequences in a given microscope field. The resultant training models' prediction ability is subsequently evaluated on a new dataset D_3, also consisting of N variable-length data sequences, with a 10/90 training/prediction ratio. Task 2b predictive models use the same model configuration as those in Task 2a but are not subjected to the same two-part consecutive training process. Thus, predictive models are immediately evaluated following a 10/90 train/prediction ratio. In Tasks 2c and 2d, we evaluate the transfer learning ability of the predictive models generated in Task 2a and 2b but interchange the prediction dataset D_3 with the novel dataset D_4. D_4 corresponds to the set of variable-length data sequences for N distinct L929 cells derived from time-lapse microscopy image sequences in a given microscope field. Finally, in Task 2e, neural network architectures similar to

those constructed for previous tasks are applied in the predictive classification of cells undergoing mitosis. Classification, which is the task of assigning objects to one of several predefined classes, is a pervasive problem that encompasses many applications in biology and medicine. The input data for this binary classification task is the sequence of records corresponding to the morphology of a single cell in a given microscope field, sampled every 20 minutes over 72 hours. Each record is characterized by a tuple $(\mathbf{x}, y)$, where $\mathbf{x}$ is the attribute set and y is a special attribute, designated as the class label. The attribute set in this task corresponds to the 4-dimensional data element m for a single cell and the class label is a binary attribute indicating whether or not the cell is undergoing mitosis at the given timestep. The goal during this task is to learn a target function f (or classification model), under one-shot learning conditions, that maps each attribute set $\mathbf{x}$ to one of the predefined class labels y. This classification model is used to predict the class label of unknown records. Thus, for each neural network architecture, a training set consisting of the sequence of records (for a single cell) whose class labels are known is used to build a classification model. This classification model is subsequently applied to the test set, consisting of the sequences of records (for all cells in a given microscope field) with unknown class labels. Six neural network models are evaluated at increasing hidden-layer sizes for each of the Tasks 2a-2e. In these experiments, models are trained with a recurrent layer consisting of either a GRU or LSTM.

4.3.3 Results/Discussion

Despite higher than expected RMSE values, all recurrent neural network models were found to perform significantly better than the SYNACX model. Without undergoing the iterative process commonly used to train neural networks, no particular model within the set of predictive models generated from live-cell imaging data sequences for MDA-MB231 cells under incremental learning (Table 4.14) or one-shot learning (Table 4.15) demonstrated strong predictive performance results (Figure 4.18). Although all predictive models trained with incremental learning on two consecutive datasets demonstrated modest gains in predictive performance relative to those that did not, subsequent experiments must be conducted in order to determine if and with what size of larger dataset such predictive models can match the performance of neural networks subjected to extensive iterative training. In Tasks 2c and 2d where transfer learning ability was loosely evaluated on

predictive models intially trained on MDA-MB231 cell sequence data and further trained on L929 cell sequence data, predictive performance was higher than initially expected. As with Tasks 2a,2b, predictive models trained under incremental learning (Table 4.16) performed better than those under one-shot learning (Table 4.18). Despite not demonstrating the highest predictive performance in Task 2c, the SYNACX model was able to outperform all other models under the constraints of one-shot learning in Task 2d, while also slightly decreasing it MSE from Task 2c (Figure 4.19). As with Tasks 2a and 2b, further experimentation is necessary in order to determine the optimal training dataset size for all predictive models.

As previously mentioned in Section 2.0.2, the Simplicial Grammar modeling formalism provides an inherent method for data dimensionality reduction in modeling applications involving complex biological systems and processes. Similar to other rule-based approaches for biological system modeling [151–154], the use of a finite set of rules provides an alternative approach to the traditional modeling paradigm based on systems of differential equations [155,156] that is beset by the problem of combinatorial complexity. This problem naturally arises in modeling applications where the interdependencies between system/process components can result in a combinatorial explosion of possible model configurations, making enumeration computationally expensive. In Task 2e, we were interested in the predictive classification of each cell undergoing mitosis within a given time interval and microscope field.[3] While the classification models evaluated in Table 4.18 were applied to the same test dataset, all except for the SYNACX model were trained with the training dataset. Given the computational costs associated with generating a new classification model from scratch and applying it to the large number of cells for this task, we sought to reuse prior knowledge (collective grammar learned during Task 2d) for purposes of building the SYNACX classifer and reducing the model search space to include only a subset of the cells (data sequences) necessary for the testing (prediction) phase. Using the simplicial grammar inferred from prior predictive modeling tasks, mitosis could be classified in terms of:

1) the sequence of pattern primitives (Figure 4.17a) corresponding to discrete cellular events observed during mitosis within the live-cell imaging data (Table 4.13),

[3] Since mitosis is a dynamic event characterized by both spatial and temporal patterns, the accuracy and computational cost of performing this classification task for a large population of cells (each varying in their migratory behavior over time) is highly dependent on the accuracy of border segmentation and image sampling procedure.

2) the sequence of variable-order temporal dependencies corresponding to simplicial production rules (Figure 4.17b) associated with the transitons from one phase of mitosis to another,
3) the cyclic spatio-temporal patterns (corresponding to recurrent sequences of primitives and/or simplicial rules), observed throughout the lifetime of each cell.

As shown in Figure 4.20, the application of inferred simplicial production rules associated with a specific system behavior (i.e. mitosis) can reduce the number of relevant system components prior to performing any further modeling tasks. In Figures 4.20a and 4.20b, images of cells present in a given microscope field at timestep t_0 are depicted in their original and pre-processed (with border segmentation) forms, respectively. Model search space reduction was achieved by first performing a quantization step, in which all cells at a given time-step were mapped to the 0-simplex pattern primitive (Figure 4.20c) that best resembled their morphology. Subsets of simplicial production rules (Figures 4.20e and 4.20g) from the existing sub-grammar encapsulating patterns associated with mitosis were sequentially applied to the field of 0-simplices to obtain a reduced modeling space containing only specific components of interest (i.e cells predicted to undergo cell division) (Figure 4.20h). It is with this subset of cells and their associated data sequences that all classification models were evaluated during the testing (prediction) phase. Whereas the neural-network classification models required an initial training phase the SYNACX model did not. In Figure 4.21, we illustrate the incremental process by which SYNACX infers the appropriate predictive model for a given system component (L929 cell) based on its corresponding sequence of observations in the reduced model search space. At each time-step, SYNACX evaluates the current data element associated with a system component, given the sequence of previously observed data elements and the collective grammar from Task 2c. The accuracy of SYNACX and the collective grammar in its new role as a classification model is evaluated by quantifying how often it correctly interprets the current input data element as a simplicial production rule from a previous inferred sub-grammar whose context-complex encapsulates the primitives, variable-order temporal dependencies and cyclic pattern associated with mitosis. Despite this flexibility in shared representation that allows for SYNACX and the simplicial grammar modeling formalism to train a universal model that can be used for different but related tasks, its classification and predictive performance is highly dependent on the quality of the live-cell imaging data used during the initial learning pro-

	L1		L2		L3		Train MDA-MB-231	Test MDA-MB-231
Model	Neuron type	# units	Neuron type	# units	Neuron type	# units	RMSE	RMSE
LSTM 1L	LSTM	150	-	-	-	-	303.897894	**499.3848716**
LSTM 2L	LSTM	150	LSTM	150	-	-	301.6125495	507.7005909
LSTM 3L	LSTM	150	LSTM	150	LSTM	150	305.1120614	509.2072466
GRU 1L	GRU	150	-	-	-	-	304.7398235	499.7216325
GRU 2L	GRU	150	GRU	150	-	-	307.4978049	500.6681336
GRU 3L	GRU	150	GRU	150	GRU	150	313.9821332	511.9859568
SYNACX	-	-	-	-	-	-	438.4395397	655.8316781

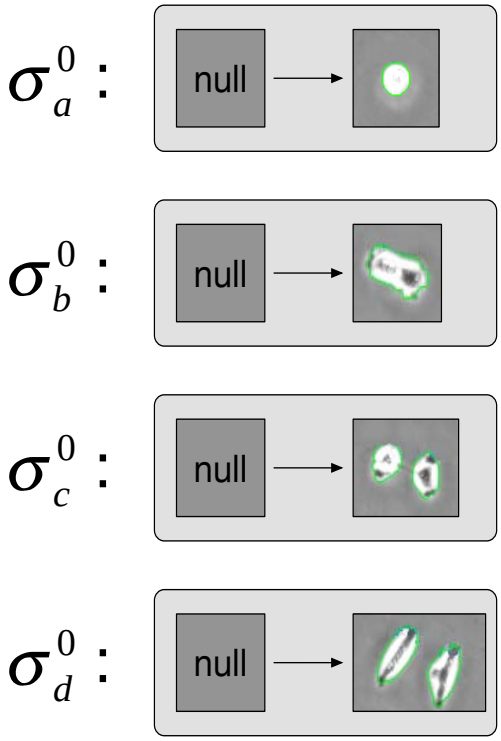
TABLE 4.14 Task 2a: Online Prediction for MDA-MB231 cells with Incremental Learning

	L1		L2		L3		Train MDA-MB-231	Test MDA-MB-231
Model	Neuron type	# units	Neuron type	# units	Neuron type	# units	RMSE	RMSE
LSTM 1L	LSTM	150	-	-	-	-	329.2765251	534.1024808
LSTM 2L	LSTM	150	LSTM	150	-	-	358.6660843	559.2743334
LSTM 3L	LSTM	150	LSTM	150	LSTM	150	464.5311184	724.6649916
GRU 1L	GRU	150	-	-	-	-	315.487892	**517.4750912**
GRU 2L	GRU	150	GRU	150	-	-	344.2288192	550.7346366
GRU 3L	GRU	150	GRU	150	GRU	150	363.6992026	567.9408596
SYNACX	-	-	-	-	-	-	430.6698271	677.1494665

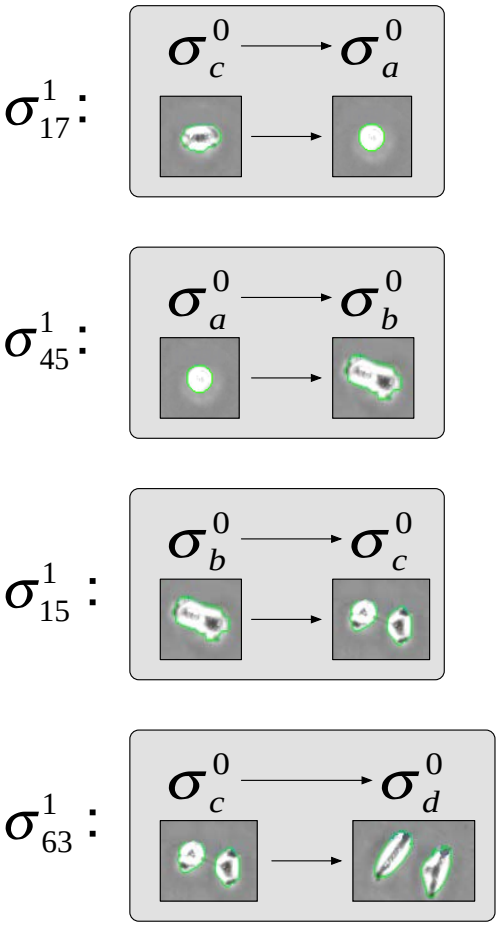
TABLE 4.15 Task 2b: Online Prediction for MDA-MB231 cells with One-Shot Learning

cess [4]. This dependence relation between data quality and classification performance is reflected in the results of Task 2e. As shown in Table 4.18, the accuracy of the SYNACX classification model is significantly lower than all neural-network configurations. Upon further investigation, it was discovered that in addition to the non-stationary behavior of migratory cells, errors that arose during image sampling and segmentation caused SYNACX to learn grammar rules for phantom phenomena (spatio-temporal patterns of noise and image artifacts observed in a data sequence). The presence of sub-grammars corresponding to these phantom phenomena within the collective grammar were found to distort the predictive models of other grammars. Thus, subsequent experiments must explore the use of alternative border segmentation methods that can overcome some of the limitations of the current approach, primarily image occlusion [157] caused by the growth in cell population within a fixed volume cell culture well.

[4]Data quality in this context refers to how well the phenomenon being modeled (cell morphology of a single cell) is represented by the data sequence used during the learning process. Choice of sampling rate and image segmentation method are very influential in this respect.



(a) 0-simplicial rules corresponding to cell morphology pattern-primitives.



(b) 1-simplicial rules corresponding to first-order temporal dependencies between cell morphology pattern-primitives.

FIGURE 4.17 Simplicial Grammar production rules inferred from experiment dataset

	L1		L2		L3		Train L929	Test L929
Model	Neuron type	# units	Neuron type	# units	Neuron type	# units	RMSE	RMSE
LSTM 1L	LSTM	150	-	-	-	-	97.26700365	131.9648059
LSTM 2L	LSTM	150	LSTM	150	-	-	90.33769977	126.0110709
LSTM 3L	LSTM	150	LSTM	150	LSTM	150	73.27885097	115.9963361
GRU 1L	GRU	150	-	-	-	-	91.69362028	127.8651243
GRU 2L	GRU	150	GRU	150	-	-	56.62261032	**105.6557145**
GRU 3L	GRU	150	GRU	150	GRU	150	63.66757416	114.2783882
SYNACX	-	-	-	-	-	-	0	128.6685276

TABLE 4.16 Task 2c: Online Prediction for L929 cells with Incremental Learning

	L1		L2		L3		Train L929	Test L929
Model	Neuron type	# units	Neuron type	# units	Neuron type	# units	RMSE	RMSE
LSTM 1L	LSTM	150	-	-	-	-	463.0643476	468.0598252
LSTM 2L	LSTM	150	LSTM	150	-	-	408.5169764	414.7995781
LSTM 3L	LSTM	150	LSTM	150	LSTM	150	425.1505145	431.0111483
GRU 1L	GRU	150	-	-	-	-	368.4471468	375.9465388
GRU 2L	GRU	150	GRU	150	-	-	362.0095164	369.6934406
GRU 3L	GRU	150	GRU	150	GRU	150	268.3646586	280.487005
SYNACX	-	-	-	-	-	-	0	**127.2681421**

TABLE 4.17 Task 2d: Online Prediction for L929 cells with One-Shot Learning

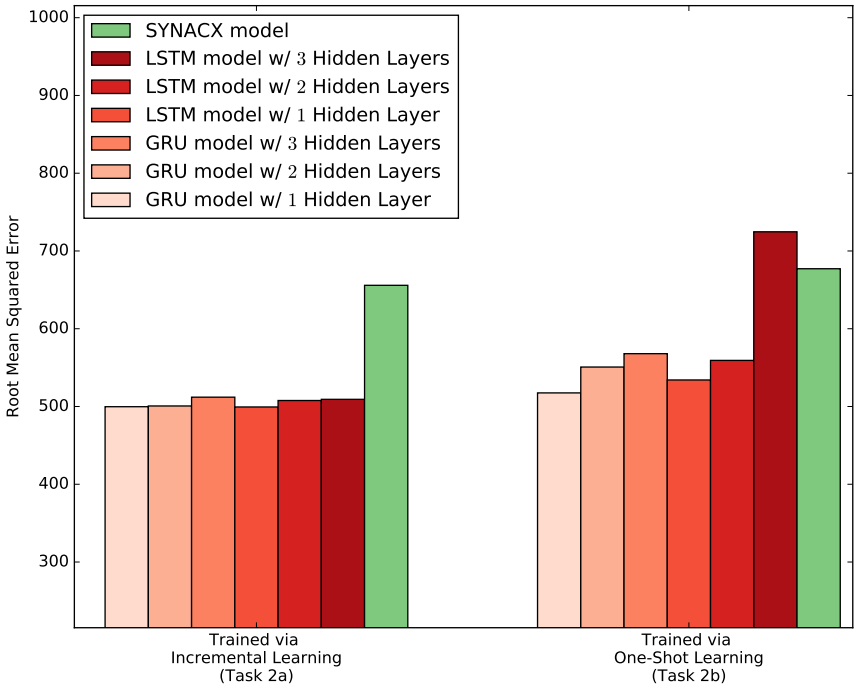


FIGURE 4.18 Evaluation of model prediction performance using MDA-MB231 cell sequence data under different training conditions.

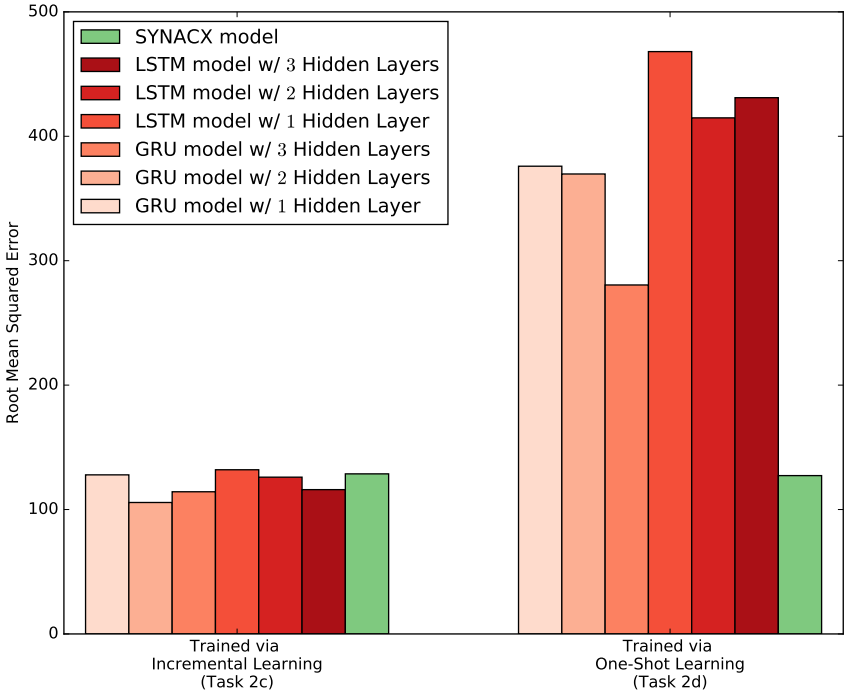
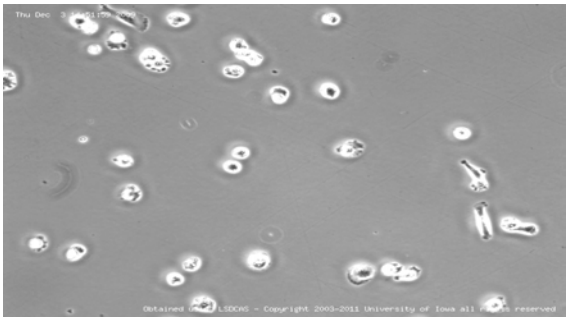
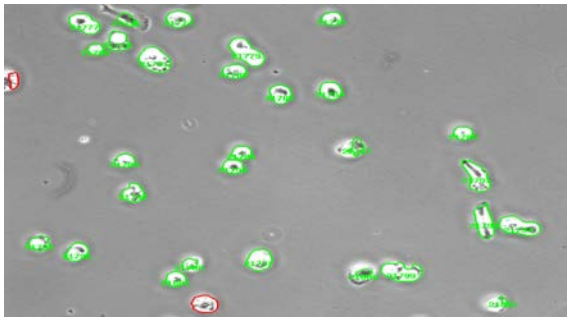


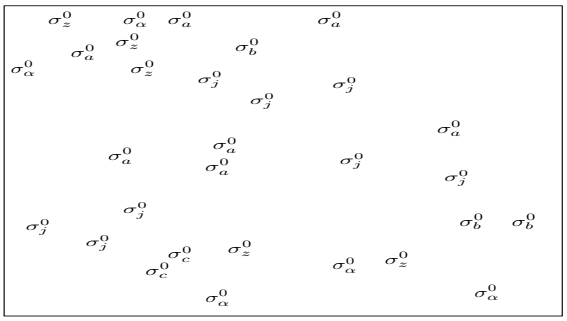
FIGURE 4.19 Comparing prediction and transfer-learning performance of the SYNACX model relative to neural network models using L929 cell sequence data under different training conditions.



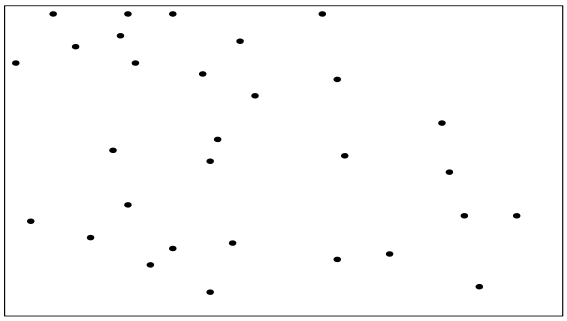
(a) Cells within microscope field of view at time t_0



(b) Cells within microscope field of view at time t_0 with automated border segmentation

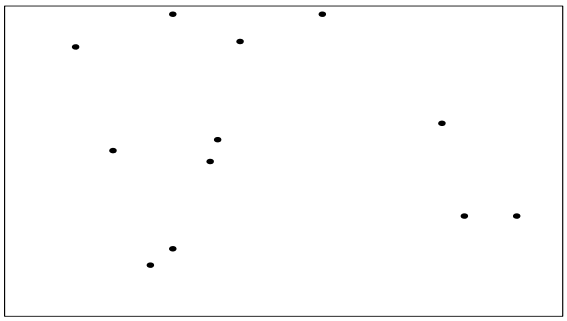


(c) Cells within microscope field of view at time t_0 labeled by collection of 0-simplices depicting cell morphology pattern-primitives

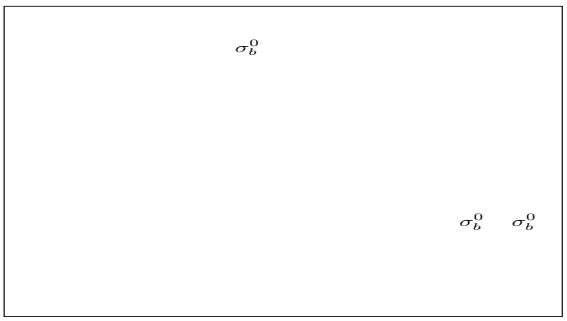


(d) Cells within microscope field of view at time t_0 as depicted graphically by vertices

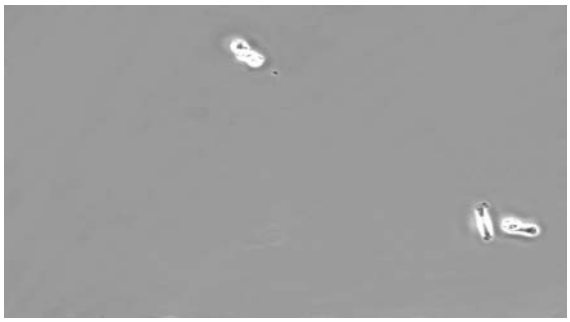
(e) Dimension reduction via application of the simplicial grammar consisting of $\{\sigma_a^0, \sigma_b^0, \sigma_c^0\}$ to Figure 4.20c.



(f) Figure 4.20e as depicted graphically by vertices



(g) Further dimension reduction via application of the simplicial grammar consisting of $\{\sigma_{17}^1, \sigma_{45}^1, \sigma_{15}^1, \sigma_{63}^1\}$ to Figure 4.20e.



(h) Illustration of microscope field at t_0 following dimension reduction via application of the simplicial grammar consisting of $\{\sigma_{17}^1, \sigma_{45}^1, \sigma_{15}^1, \sigma_{63}^1\}$

FIGURE 4.20 Application of simplicial grammar to reduce model space complexity.

		L1		L2		L3		
Train/Test	Model	Neuron type	# units	Neuron type	# units	Neuron type	# units	Accuracy
10%/90%	LSTM Binary classifier	LSTM	150	-	-	-	-	86.59
10%/90%	LSTM Binary classifier	LSTM	150	LSTM	150	-	-	86.59
10%/90%	LSTM Binary classifier	LSTM	150	LSTM	150	LSTM	150	86.59
10%/90%	GRU Binary classifier	GRU	150	-	-	-	-	86.59
10%/90%	GRU Binary classifier	GRU	150	GRU	150	-	-	86.59
10%/90%	GRU Binary classifier	GRU	150	GRU	150	GRU	150	86.59
	SYNACX	-	-	-	-	-	-	73

TABLE 4.18 Task 2e: Predictive Classification of Mitosis in L929 cell seqeuence following One-Shot Learning

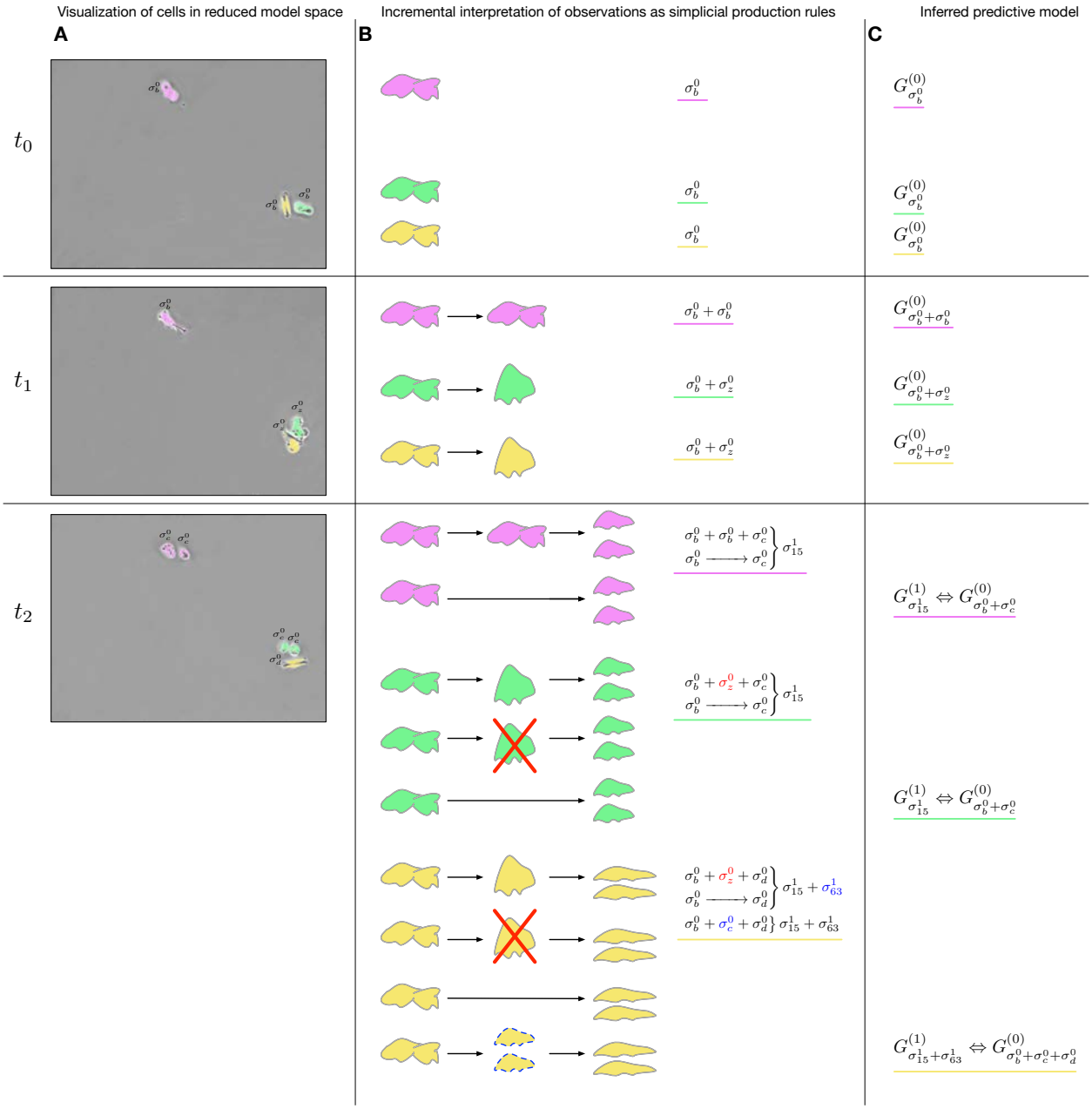


FIGURE 4.21 Application of simplicial grammar to reduce combinatorial complexity during predictive modeling of mitosis

CHAPTER 5

Summary and Future Work

5.1 Summary

In this thesis we have described the development and application of the simplicial grammar modeling formalism and SYNACX framework for learning and building probabilistic generative models of complex biological systems/processes directly from real-world data sequences. This new modeling formalism and approximate inference scheme provides a unified probabilistic framework to decompose complex system behavior into modular grammar rules which parsimoniously describe the spatial/temporal structure and dynamics of patterns inferred from time-series data. The key advantage of simplicial grammars which makes them ideal for general-purpose learning is their ability to appropriately represent and recognize the large intra-class variablities between patterns in terms of their compositionality (i.e. decomposition of a pattern into a distinct configuration of sub-patterns) and reconfigurablity (i.e. representation of a pattern using multiple distinct configurations). In addition to allowing the number of components to vary during inference, as in traditional Bayesian nonparametric models, the use of simplicial grammar in the SYNACX model provides a flexible approach to modeling higher-order representations of the state and dynamics of a complex system despite model and data uncertainty. As a result, the simplicial grammar modeling formalism and the SYNACX framework provide an alternative software platform for extracting, consolidating and visualizing knowledge in applications where multi-modal, transfer, multi-task and one-shot learning are to be combined.

Our preliminary experiments using SYNACX in online prediction tasks demonstrate the utility

of this novel approach as a possible alternative to popular gradient-based neural-network architectures in one-shot learning applications where subtle and latent patterns are to be inferred from various data sources over time, given limited *a priori* knowledge. Given the rapid pace of technological advancement in the field of deep learning, we look to develop extensions of the SYNACX framework incorporating more elements from memory-augmented networks [110, 113] in an effort to achieve greater performance results in all evaluated tasks. Subsequent experiments involving heterogeneous datatypes generated across multiple spatio-temporal scales will be used to further investigate the use of simplicial grammar and its topological invariants as a means for integrating models of global behavior with those of local interactions. Building such models of complex biological systems and processes from the molecular level up to the entire organism will allow researchers to close the computational gap in systems biology. The challenge is to construct and integrate all these models in a data-driven manner that enables discovery and enhances our understanding of the processes, sub-systems and interdependencies that exist across multiple spatial and temporal scales.

5.2 Future Work

Many of the modeling applications in systems and computational biology have focused on intracellular dynamics. We strive to build computational models that encapsulate how emergent global behavior and functionality is heavily influenced by the interactions and local dynamics of lower-level sub-systems, and vice-versa by also incorporating quantitative models of behaviors occuring at the population level. To achieve this, we look to extend our work in Artificial General Intelligence to building agent-based models of a general class of complex systems, called Complex Adaptive Systems [158]. Such systems encompass a broad range of biological phenomena. A hallmark of complex adaptive systems is the existence of heterogeneous components, often called agents, which adapt and evolve as they interact in their environment. The combination of such a diverse array of agents, each interacting with other agents gives rise to a system that is complex due to the conditional interactions that dictate their behavior and adaptive because of an agent's inherent ability to learn from each interaction. Due to the non-linearity that arises from these conditional

interactions, a complex adaptive system does not have a single governing equation or rule by which the system is controlled. Although its many distributed, interacting agents display little if any central control, each is governed by a generative process that can be described by a set of local rules. If defined in terms of the spatial and temporal patterns of the system, this collection of rules may be regarded as a grammar expressing patterns as a composition of subpatterns and pattern primitives. The unsupervised learning of these rules and their underlying patterns falls under the class of factorial learning problems in which the goal is to identify the multiple independent causes or factors that accurately characterize the observed data. This learning problem often arises in response to the actual process by which the data was generated. For high-throughput, biological time-series data that may result from interactions across multiple scales, the goal is to invert the data generation process, and identify a representation that concisely describes the data and reflects its underlying causes and interdependencies. Agent-based models depict populations of autonomous agents, each following a set of internal rules and interacting with each other within a shared virtual environment. With this approach, researchers can move beyond modeling approaches based a single level of representation towards multi-scale simulations and representations which may enable further insights about the relationship between microscopic rules of the agents and macroscopic behaviors of the population. By using the aforementioned SYNACX framework and incorporating elements of reinforcement learning, we seek to infer the set of internal rules that define agent behavior directly from biological sequence data. The aim of this future work is build agent-based models of biological phenomena with which researchers can explore and discover the scope of novel targets useful for therapeutic intervention *in-silico*, with minimal time, computational resources or expert supervision.

Bibliography

[1] O. Z. Kraus, B. T. Grys, J. Ba, Y. Chong, B. J. Frey, C. Boone, and B. J. Andrews, "Automated analysis of high-content microscopy data with deep learning." *Molecular systems biology*, vol. 13, p. 924, Apr. 2017.

[2] C. Angermueller, T. Pärnamaa, L. Parts, and O. Stegle, "Deep learning for computational biology." *Molecular systems biology*, vol. 12, p. 878, Jul. 2016.

[3] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[4] T. C. A. Amin, R. P. D. M. Kamel, and D. de Ridder, "Structural, syntactic, and statistical pattern recognition," 2008.

[5] K.-S. Fu, "A step towards unification of syntactic and statistical pattern recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, no. 3, pp. 398–404, 1986.

[6] S. K. Pal and A. Pal, *Pattern recognition: from classical to modern approaches*. World Scientific, 2001.

[7] J. McCarthy, M. Minsky, and N. Rochester, "A proposal for the dartmouth summer research project on artificial intelligence," 1955.

[8] S. Russell, P. Norvig, and A. Intelligence, "A modern approach," *Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs*, vol. 25, p. 27, 1995.

[9] *The Goal of Artificial Intelligence*. Dordrecht: Springer Netherlands, 2006, pp. 3–27. [Online]. Available: https://doi.org/10.1007/1-4020-5045-3_1

[10] R. V. Yampolskiy and J. Fox, "Artificial general intelligence and the human mental model," in *Singularity Hypotheses*. Springer, 2012, pp. 129–145.

[11] J. McCarthy and P. J. Hayes, "Some philosophical problems from the standpoint of artificial intelligence," *Readings in artificial intelligence*, pp. 431–450, 1969.

[12] B. Goertzel and C. Pennachin, *Artificial general intelligence*. Springer, 2007, vol. 2.

[13] P. Wang and B. Goertzel, "Introduction: Aspects of artificial general intelligence," in *Proceedings of the 2007 conference on Advances in Artificial General Intelligence: Concepts, Architectures and Algorithms: Proceedings of the AGI Workshop 2006*. IOS Press, 2007, pp. 1–16.

[14] C. Pennachin and B. Goertzel, "Contemporary approaches to artificial general intelligence," *Artificial general intelligence*, pp. 1–30, 2007.

[15] M.-T. Luong, Q. V. Le, I. Sutskever, O. Vinyals, and L. Kaiser, "Multi-task sequence to sequence learning," *ArXiv e-prints*, Nov. 2015.

[16] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *Journal of Machine Learning Research*, vol. 12, no. Aug, pp. 2493–2537, 2011.

[17] J. Weston, A. Bordes, S. Chopra, A. M. Rush, B. van Merriënboer, A. Joulin, and T. Mikolov, "Towards ai-complete question answering: A set of prerequisite toy tasks," *arXiv preprint arXiv:1502.05698*, 2015.

[18] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[19] Y. Bengio, "Deep learning of representations for unsupervised and transfer learning," in *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, 2012, pp. 17–36.

[20] L. Fei-Fei, R. Fergus, and P. Perona, "One-shot learning of object categories," *IEEE transactions on pattern analysis and machine intelligence*, vol. 28, no. 4, pp. 594–611, 2006.

[21] R. Salakhutdinov, J. Tenenbaum, and A. Torralba, "One-shot learning with a hierarchical nonparametric bayesian model," in *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, 2012, pp. 195–206.

[22] D. R. Rhodes, S. A. Tomlins, S. Varambally, V. Mahavisno, T. Barrette, S. Kalyana-Sundaram, D. Ghosh, A. Pandey, and A. M. Chinnaiyan, "Probabilistic model of the human protein-protein interaction network," *Nature biotechnology*, vol. 23, no. 8, pp. 951–959, 2005.

[23] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum, "Human-level concept learning through probabilistic program induction," *Science*, vol. 350, no. 6266, pp. 1332–1338, 2015.

[24] J. Rajendran, M. M. Khapra, S. Chandar, and B. Ravindran, "Bridge correlational neural networks for multilingual multimodal representation learning," *arXiv preprint arXiv:1510.03519*, 2015.

[25] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng, "Multimodal deep learning," in *Proceedings of the 28th international conference on machine learning (ICML-11)*, 2011, pp. 689–696.

[26] N. Srivastava and R. R. Salakhutdinov, "Multimodal learning with deep boltzmann machines," in *Advances in neural information processing systems*, 2012, pp. 2222–2230.

[27] K. Sohn, W. Shang, and H. Lee, "Improved multimodal deep learning with variation of information," in *Advances in Neural Information Processing Systems*, 2014, pp. 2141–2149.

[28] P. Wu, S. C. Hoi, H. Xia, P. Zhao, D. Wang, and C. Miao, "Online multimodal deep similarity learning with application to image retrieval," in *Proceedings of the 21st ACM international conference on Multimedia*. ACM, 2013, pp. 153–162.

[29] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[30] D. A. Ferrucci, E. W. Brown, J. Chu-Carroll, J. Fan, D. Gondek, A. Kalyanpur, A. Lally, J. W. Murdock, E. Nyberg, J. M. Prager, N. Schlaefer, and C. A. Welty, "Building watson: An overview of the deepqa project." *AI Magazine*, vol. 31, no. 3, pp. 59–79, 2010. [Online]. Available: http://dblp.uni-trier.de/db/journals/aim/aim31.html#FerrucciBCFGKLMNPSW10

[31] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[32] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[33] F. Rosenblatt, "Principles of neurodynamics. perceptrons and the theory of brain mechanisms," CORNELL AERONAUTICAL LAB INC BUFFALO NY, Tech. Rep., 1961.

[34] M. Minsky and S. Papert, "Perceptrons." 1969.

[35] *Feedforward Networks I: Generalities and LTU Nodes*. New York, NY: Springer New York, 1999, pp. 53–80. [Online]. Available: https://doi.org/10.1007/0-387-22649-4_3

[36] T. Mikolov, A. Joulin, S. Chopra, M. Mathieu, and M. Ranzato, "Learning longer memory in recurrent neural networks," *ArXiv e-prints*, Dec. 2014.

[37] J. R. Chung, J. Kwon, T. A. Mann, and Y. Choe, *Evolution of Time in Neural Networks: From the Present to the Past, and Forward to the Future*. Boston, MA: Springer US, 2012, pp. 99–115. [Online]. Available: https://doi.org/10.1007/978-1-4614-0724-9_6

[38] T. Jebara, *Machine learning: discriminative and generative*. Springer Science & Business Media, 2012, vol. 755.

[39] Y. Bengio *et al.*, "Learning deep architectures for ai," *Foundations and trends® in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[40] Y. LeCun, L. Bottou, G. B. Orr, and K. R. Müller, *Efficient BackProp*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 9–50. [Online]. Available: https://doi.org/10.1007/3-540-49430-8_2

[41] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[42] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural networks for machine learning*, vol. 4, no. 2, pp. 26–31, 2012.

[43] J. Chung, S. Ahn, and Y. Bengio, "Hierarchical multiscale recurrent neural networks," *ArXiv e-prints*, Sep. 2016.

[44] Z. C. Lipton, J. Berkowitz, and C. Elkan, "A critical review of recurrent neural networks for sequence learning," *arXiv preprint arXiv:1506.00019*, 2015.

[45] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997. [Online]. Available: http://dx.doi.org/10.1162/neco.1997.9.8.1735

[46] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[47] Z. Che, S. Purushotham, K. Cho, D. Sontag, and Y. Liu, "Recurrent neural networks for multivariate time series with missing values," *arXiv preprint arXiv:1606.01865*, 2016.

[48] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent, "Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription," *arXiv preprint arXiv:1206.6392*, 2012.

[49] R. Jozefowicz, W. Zaremba, and I. Sutskever, "An empirical exploration of recurrent network architectures," in *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, 2015, pp. 2342–2350.

[50] S. Crespi Reghizzi, *Formal Languages and Compilation*, 2009.

[51] M. A. Harrison, *Introduction to formal language theory*. Addison-Wesley Longman Publishing Co., Inc., 1978.

[52] P. P. Perez Velasco, "Matrix graph grammars," *ArXiv e-prints*, Jan. 2008.

[53] K. Etessami, A. Stewart, and M. Yannakakis, "Stochastic context-free grammars, regular languages, and newton's method," *ArXiv e-prints*, Feb. 2013.

[54] J. Engelfriet, "Tree automata and tree grammars," *ArXiv e-prints*, Oct. 2015.

[55] G. Vishnu Murthy, C. Pavan Kumar, and V. Vijaya Kumar, "A new model of array grammar for generating connected patterns on an image neighborhood," *ArXiv e-prints*, Jul. 2014.

[56] C. Pennycuff, S. Aguinaga, and T. Weninger, "A temporal tree decomposition for generating temporal graphs," *ArXiv e-prints*, Jun. 2017.

[57] M. Marcolli and A. Port, "Graph grammars, insertion lie algebras, and quantum field theory," *ArXiv e-prints*, Feb. 2015.

[58] H. Zenil, "Algorithmic data analytics, small data matters and correlation versus causation," *ArXiv e-prints*, Sep. 2013.

[59] A. B. Patel, T. Nguyen, and R. G. Baraniuk, "A probabilistic theory of deep learning," *ArXiv e-prints*, Apr. 2015.

[60] F. Saad and V. Mansinghka, "Probabilistic data analysis with probabilistic programming," *ArXiv e-prints*, Aug. 2016.

[61] T. Ferguson, "A Bayesian analysis of some nonparametric problems," *The Annals of Statistics*, vol. 1, no. 2, pp. 209–230, 1973.

[62] R. B. Ash and C. Doleans-Dade, *Probability and measure theory*. Academic Press, 2000.

[63] R. Durrett, *Probability: theory and examples*. Cambridge university press, 2010.

[64] W. M. Bolstad and J. M. Curran, *Introduction to Bayesian statistics*. John Wiley & Sons, 2016.

[65] A. Gelman, J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin, *Bayesian data analysis*. CRC press Boca Raton, FL, 2014, vol. 2.

[66] B. P. Carlin and T. A. Louis, *Bayes and empirical Bayes methods for data analysis*. Chapman & Hall/CRC Boca Raton, FL, 2000, vol. 17.

[67] C. C. Aggarwal and C. K. Reddy, *Data clustering: algorithms and applications*. CRC press, 2013.

[68] K. J. Cios, R. W. Swiniarski, W. Pedrycz, and L. A. Kurgan, "Unsupervised learning: clustering," in *Data Mining*. Springer, 2007, pp. 257–288.

[69] A. K. Jain, "Data clustering: 50 years beyond k-means," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2008, pp. 3–4.

[70] S. E. Schaeffer, "Graph clustering," *Computer science review*, vol. 1, no. 1, pp. 27–64, 2007.

[71] F. Pernkopf and D. Bouchaffra, "Genetic-based em algorithm for learning gaussian mixture models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 8, pp. 1344–1348, 2005.

[72] S. Lloyd, "Least squares quantization in pcm," *IEEE transactions on information theory*, vol. 28, no. 2, pp. 129–137, 1982.

[73] S. C. Johnson, "Hierarchical clustering schemes," *Psychometrika*, vol. 32, no. 3, pp. 241–254, 1967.

[74] V. Melnykov, R. Maitra *et al.*, "Finite mixture models and model-based clustering," *Statistics Surveys*, vol. 4, pp. 80–116, 2010.

[75] G. McLachlan and D. Peel, *Finite mixture models*. John Wiley & Sons, 2004.

[76] C. Fraley and A. E. Raftery, "Model-based clustering, discriminant analysis, and density estimation," *Journal of the American statistical Association*, vol. 97, no. 458, pp. 611–631, 2002.

[77] D. Pelleg, A. W. Moore *et al.*, "X-means: Extending k-means with efficient estimation of the number of clusters." in *ICML*, vol. 1, 2000, pp. 727–734.

[78] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *Kdd*, vol. 96, no. 34, 1996, pp. 226–231.

[79] N. L. Hjort, C. Holmes, P. Müller, and S. G. Walker, *Bayesian nonparametrics*. Cambridge University Press, 2010, vol. 28.

[80] Y. W. Teh and M. I. Jordan, "Hierarchical bayesian nonparametric models with applications," *Bayesian nonparametrics*, vol. 1, 2010.

[81] D. Blackwell and J. MacQueen, "Ferguson distributions via Polya urn schemes," *The Annals of Statistics*, vol. 1, pp. 353–355, 1973.

[82] J. Pitman and M. Yor, "The two-parameter Poisson-Dirichlet distribution derived from a stable subordinator," *The Annals of Probability*, vol. 25, pp. 855–900, 1997.

[83] Y. W. Teh, "A hierarchical Bayesian language model based on Pitman-Yor processes," in *Proceedings of the Association for Computational Linguistics*. Association for Computational Linguistics, 2006, pp. 985–992.

[84] F. Wood and Y. Teh, "A hierarchical, hierarchical Pitman Yor process language model," in *ICML/UAI Nonparametric Bayes Workshop*, 2008.

[85] M. Johnson, T. Griffiths, and S. Goldwater, "Adaptor grammars: A framework for specifying compositional nonparametric Bayesian models," in *Advances in Neural Information Processing Systems*, vol. 19, 2007, p. 641.

[86] S. J. Gershman and D. M. Blei, "A tutorial on bayesian nonparametric models," *Journal of Mathematical Psychology*, vol. 56, no. 1, pp. 1–12, 2012.

[87] T. S. Ferguson, "Bayesian density estimation by mixtures of normal distributions," in *Recent advances in statistics*, M. Rizvi, J. Rustagi, and D. Siegmund, Eds. New York: Academic Press, 1983, pp. 287–302.

[88] J. Sethuraman, "A constructive definition of dirichlet priors," DTIC Document, Tech. Rep., 1991.

[89] J. Pitman, "Some developments of the Blackwell-MacQueen urn scheme," in *Statistics Probability and Game Theory; Papers in honor of David Blackwell*, ser. Lecture Notes – Monograph Series, T. S. Ferguson, L. S. Shapley, and J. B. MacQueen, Eds. Institute of Mathematical Statistics, 1996, pp. 245–267.

[90] Y. W. Teh, "Dirichlet process," in *Encyclopedia of machine learning*. Springer, 2011, pp. 280–287.

[91] D. M. Blei, M. I. Jordan *et al.*, "Variational inference for dirichlet process mixtures," *Bayesian analysis*, vol. 1, no. 1, pp. 121–143, 2006.

[92] J. Pitman, "Coalescents with multiple collisions," *The Annals of Probability*, vol. 27, pp. 1870–1902, 1999.

[93] Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei, "Hierarchical dirichlet processes," *Journal of the american statistical association*, vol. 101, no. 476, 2006.

[94] P. Lum, G. Singh, A. Lehman, T. Ishkanov, M. Vejdemo-Johansson, M. Alagappan, J. Carlsson, and G. Carlsson, "Extracting insights from the shape of complex data using topology," *Scientific reports*, vol. 3, p. 1236, 2013.

[95] G. Carlsson, "Topology and data," *Bulletin of the American Mathematical Society*, vol. 46, no. 2, pp. 255–308, 2009.

[96] T. Y. Lin and N. Cercone, *Rough sets and data mining: Analysis of imprecise data*. Springer Science & Business Media, 2012.

[97] H. Adams, T. Emerson, M. Kirby, R. Neville, C. Peterson, P. Shipman, S. Chepushtanova, E. Hanson, F. Motta, and L. Ziegelmeier, "Persistence images: a stable vector representation of persistent homology," *Journal of Machine Learning Research*, vol. 18, no. 8, pp. 1–35, 2017.

[98] A. Hatcher, *Algebraic topology*, 2002.

[99] T. Becker and V. Weispfenning, *Gröbner bases*. Springer, 1993.

[100] B. Buchberger and F. Winkler, *Gröbner bases and applications*. Cambridge University Press, 1998, vol. 251.

[101] B. Buchberger, "Applications of gröbner bases in non-linear computational geometry," in *Trends in computer algebra*. Springer, 1988, pp. 52–80.

[102] J. R. Munkres, *Elements of algebraic topology*. Addison-Wesley Reading, 1984, vol. 2.

[103] B. Buchberger, "Gröbner bases: An algorithmic method in polynomial ideal theory," in *Multidimensional Systems Theory–Progress, Directions and Open Problems in Multidimensional Systems*, 1985, pp. 184–232.

[104] H. Hassani, X. Huang, and M. Ghodsi, "Big data and causality," *Annals of Data Science*, p. 1, Aug. 2017. [Online]. Available: http://dx.doi.org/10.1007/s40745-017-0122-3

[105] K. B. Athreya and S. N. Lahiri, *Measure theory and probability theory*. Springer Science & Business Media, 2006.

[106] Z. Ghahramani, "An introduction to hidden markov models and bayesian networks," *International journal of pattern recognition and artificial intelligence*, vol. 15, no. 01, pp. 9–42, 2001.

[107] M. Längkvist, L. Karlsson, and A. Loutfi, "A review of unsupervised feature learning and deep learning for time-series modeling," *Pattern Recognition Letters*, vol. 42, pp. 11–24, 2014.

[108] J. Kalantari and M. A. Mackey, "One-shot ontogenetic learning in biomedical datastreams," in *International Conference on Artificial General Intelligence*. Springer, 2017, pp. 143–153.

[109] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom, "Models and issues in data stream systems," in *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. ACM, 2002, pp. 1–16.

[110] Ł. Kaiser and I. Sutskever, "Neural gpus learn algorithms," *ArXiv e-prints*, Nov. 2015.

[111] M. Ranzato, S. Chopra, M. Auli, and W. Zaremba, "Sequence level training with recurrent neural networks," *ArXiv e-prints*, Nov. 2015.

[112] F. Kemény and B. Meier, "Multimodal sequence learning." *Acta psychologica*, vol. 164, pp. 27–33, Feb. 2016.

[113] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *ArXiv e-prints*, Sep. 2014.

[114] D. Floreano and C. Mattiussi, *Bio-inspired artificial intelligence: theories, methods, and technologies*. MIT press, 2008.

[115] C. A. Knoblock, "Learning abstraction hierarchies for problem solving." in *AAAI*, 1990, pp. 923–928.

[116] L. Darden, "Artificial intelligence and philosophy of science: Reasoning by analogy in theory construction," in *PSA: Proceedings of the biennial meeting of the Philosophy of Science Association*, vol. 1982, no. 2. Philosophy of Science Association, 1982, pp. 147–165.

[117] G. F. Luger, "Artificial intelligence: Structures and strategies for complex problem solving," 2008.

[118] T. M. Mitchell, R. M. Keller, and S. T. Kedar-Cabelli, "Explanation-based generalization: A unifying view," *Machine learning*, vol. 1, no. 1, pp. 47–80, 1986.

[119] L. R. Ye and P. E. Johnson, "The impact of explanation facilities on user acceptance of expert systems advice," *Mis Quarterly*, pp. 157–172, 1995.

[120] I. B. Ji-Ye Mao, "The use of explanations in knowledge-based systems: Cognitive perspectives and a process-tracing analysis," *Journal of Management Information Systems*, vol. 17, no. 2, pp. 153–179, 2000.

[121] A. Zomorodian and G. Carlsson, "Computing persistent homology," *Discrete & Computational Geometry*, vol. 33, no. 2, pp. 249–274, 2005.

[122] H. Poincaré, *Analysis situs*. na, 1895.

[123] M. Kirby, *Geometric data analysis: an empirical approach to dimensionality reduction and the study of patterns*. John Wiley & Sons, Inc., 2000.

[124] V. D. Silva and J. B. Tenenbaum, "Global versus local methods in nonlinear dimensionality reduction," in *Advances in neural information processing systems*, 2003, pp. 721–728.

[125] R. Ghrist, "Barcodes: the persistent topology of data," *Bulletin of the American Mathematical Society*, vol. 45, no. 1, pp. 61–75, 2008.

[126] F.-O. Schreyer, "Die berechnung von syzygien mit dem verallgemeinerten weierstraßschen divisionssatz und eine anwendung auf analytische cohen-macaulay stellenalgebren minimaler multiplizität," Ph.D. dissertation, 1980.

[127] G. Singh, "Algorithms for topological analysis of data," 2008.

[128] J. Catlett, "On changing continuous attributes into ordered discrete attributes," in *Machine learning EWSL-91*. Springer, 1991, pp. 164–178.

[129] Y. Linde, A. Buzo, and R. M. Gray, "An algorithm for vector quantizer design," *Communications, IEEE Transactions on*, vol. 28, no. 1, pp. 84–95, 1980.

[130] Y. W. Teh, "A bayesian interpretation of interpolated kneser-ney," 2006.

[131] N. Bartlett, D. Pfau, and F. Wood, "Forgetting counts: Constant memory inference for a dependent hierarchical Pitman-Yor process," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, June 2010, pp. 63–70. [Online]. Available: http://www.icml2010.org/papers/549.pdf

[132] F. Wood, C. Archambeau, J. Gasthaus, L. James, and Y. W. Teh, "A stochastic memoizer for sequence data," in *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, 2009, pp. 1129–1136.

[133] F. Wood, J. Gasthaus, C. Archambeau, L. James, and Y. W. Teh, "The sequence memoizer," *Communications of the ACM*, vol. 54, no. 2, pp. 91–98, 2011.

[134] D. Crisan, "Particle filters—a theoretical perspective," in *Sequential Monte Carlo methods in practice*. Springer, 2001, pp. 17–41.

[135] H. Crane *et al.*, "The ubiquitous ewens sampling formula," *Statistical Science*, vol. 31, no. 1, pp. 1–19, 2016.

[136] S. Tavaré, "The ewens multivariate distribution," *Multivariate discrete distributions*, 1997.

[137] J. Pitman, "Combinatorial stochastic processes," 2002, notes for Saint Flour Summer School.

[138] Y. Chen, E. Keogh, B. Hu, N. Begum, A. Bagnall, A. Mueen, and G. Batista, "The ucr time series classification archive," Jul. 2015.

[139] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, and e. Devin, Matthieu, "Tensorflow: large-scale machine learning on heterogeneous systems, software available from tensorflow. org," *Available: http://tensorflow.org*, 2015.

[140] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting." *Journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[141] N. N. Taleb, "The black swan: The impact of the highly improbable (the incerto collection)," 2007.

[142] A. Bellouquid, E. De Angelis, and D. Knopoff, "From the modeling of the immune hallmarks of cancer to a black swan in biology," *Mathematical Models and Methods in Applied Sciences*, vol. 23, no. 05, pp. 949–978, 2013.

[143] D. Grün and A. van Oudenaarden, "Design and analysis of single-cell sequencing experiments," *Cell*, vol. 163, no. 4, pp. 799–810, 2015.

[144] T. Kalisky, P. Blainey, and S. R. Quake, "Genomic analysis at the single-cell level," *Annual review of genetics*, vol. 45, pp. 431–445, 2011.

[145] S. S. Rubakhin, E. V. Romanova, P. Nemes, and J. V. Sweedler, "Profiling metabolites and peptides in single cells," *Nature methods*, vol. 8, no. 4s, pp. S20–S29, 2011.

[146] P. J. Davis, E. A. Kosmacek, Y. Sun, F. Ianzini, and M. A. Mackey, "The large-scale digital cell analysis system: an open system for nonperturbing live cell imaging," *Journal of microscopy*, vol. 228, no. 3, pp. 296–308, 2007.

[147] M. A. Mackey, K. Anderson, L. Bresnahan, F. Domann, G. Gallardo, F. Ianzini, E. Kosmacek, Y. Li, M. Sonka, D. Spitz *et al.*, "The large scale digital cell analysis system: a unique tool for the study of molecular and cellular phenomena in living cell populations," *Mol Imaging*, vol. 2, p. 226, 2003.

[148] F. Ianzini, L. Bresnahan, L. Wang, K. Anderson, and M. Mackey, "The large scale digital cell analysis system and its use in the quantitative analysis of cell populations," in *Microtechnologies in Medicine & Biology 2nd Annual International IEEE-EMB Special Topic Conference on*. IEEE, 2002, pp. 470–475.

[149] F. Ianzini and M. A. Mackey, "Development of the large scale digital cell analysis system," *Radiation protection dosimetry*, vol. 99, no. 1-4, pp. 289–293, 2002.

[150] L. Lamalice, F. Le Boeuf, and J. Huot, "Endothelial cell migration during angiogenesis," *Circulation research*, vol. 100, no. 6, pp. 782–794, 2007.

[151] L. A. Chylek, L. A. Harris, C.-S. Tung, J. R. Faeder, C. F. Lopez, and W. S. Hlavacek, "Rule-based modeling: a computational approach for studying biomolecular site dynamics in cell signaling systems," *Wiley Interdisciplinary Reviews: Systems Biology and Medicine*, vol. 6, no. 1, pp. 13–36, 2014.

[152] J. Wilson-Kanamori, V. Danos, T. Thomson, and R. Honorato-Zimmer, "Kappa rule-based modeling in synthetic biology," *Computational Methods in Synthetic Biology*, pp. 105–135, 2015.

[153] L. A. Chylek, E. C. Stites, R. G. Posner, and W. S. Hlavacek, "Innovations of the rule-based modeling approach," in *Systems Biology*. Springer, 2013, pp. 273–300.

[154] V. Danos, J. Feret, W. Fontana, R. Harmer, and J. Krivine, "Rule-based modelling of cellular signalling," in *International conference on concurrency theory*. Springer, 2007, pp. 17–41.

[155] B. Schoeberl, C. Eichler-Jonsson, E. D. Gilles, and G. Müller, "Computational modeling of the dynamics of the map kinase cascade activated by surface and internalized egf receptors," *Nature biotechnology*, vol. 20, no. 4, pp. 370–375, 2002.

[156] R. J. Orton, O. E. Sturm, V. Vyshemirsky, M. Calder, D. R. Gilbert, and W. Kolch, "Computational modelling of the receptor-tyrosine-kinase-activated mapk pathway," *Biochemical Journal*, vol. 392, no. 2, pp. 249–261, 2005.

[157] W. C. Mankowski, M. R. Winter, E. Wait, M. Lodder, T. Schumacher, S. H. Naik, and A. R. Cohen, "Segmentation of occluded hematopoietic stem cells from tracking," in *Engineering in Medicine and Biology Society (EMBC), 2014 36th Annual International Conference of the IEEE*. IEEE, 2014, pp. 5510–5513.

[158] J. H. Holland, "Studying complex adaptive systems," *Journal of Systems Science and Complexity*, vol. 19, no. 1, pp. 1–8, 2006.