Computability Theory for Neuroscience

by Doug Rubino

Abstract

Neuronal circuits are ubiquitously held to be the substrate of computation in the brain, information processing in single neurons is though of in terms of encoding, and neurons themselves are modeled with dynamical systems. The notions of computation, information and dynamics are three terms that are formally defined, for the most part, in historically independent fields. Dynamical systems are not a traditional model of computation, and the theory of dynamical systems plays little to no role in the history of computers. The temporal non-linearities observed in biological systems make them fundamentally difficult to describe in an information theoretic framework, which requires the base signal to be approximately stationary or ergodic [2]. Non-linear systems are best thought of as performing computations, and yet the notion of computation is best-studied in frameworks that are recursive, or discrete. This work formalizes that intuition by introducting a discrete model of computation called streaming automata. I demonstrate that these automata are an appropriate model of computation for neurons, a fact that requires the construction of a bijection between streaming automata and biological dynamical systems that holds to an error that is arbitrarily small. For this reason, streaming automata are a universal and non-parametric non-linear model [3].

1 Introduction

In the face of tremendous diversity in ion channels, ion channel densities and dendritic geometry, it is clear that finding appropriate models to represent neurons is no easy task [4]. The functional characteristics of any particular neuron presumably exist to subserve one or several specific computational roles, and there is no universal error metric that captures this notion of neuronal function nor is there a well-defined notion of computation in neuroscience.

Computability theory is field jointly created by several historically important theorists including Alan Turing, Alonzo Church and Stephen Kleene [5]. Among other things, the formalism of computability theory has yielded a definition for computation. This definition is fundamentally is model dependent, and the Church-Turing thesis is the statement that all models of computation will be equivalent to the ones that have already studied. The seminal universal models of computation include the Turing Machine, presented in the work of Alan Turing [6], and the Lambda Calculus presented in the work of Alonzo

Church [1]. Because the word *computation* is only well-defined with respect to a chosen model of computation, it is important that we fix a particular model of computation in neuroscience that is powerful enough to capture all possible neuronal dynamics, and one that has no extraneous theoretical baggage to obscure the link between theory and data. This will serve to make model fitting no more difficult than it has to be.

A particular problem with neuronal systems in neuroscience is that all elements engage in real-time computation. The classical models in computability theory are called acceptor models of computation. They take an input string, flip some bits, and then (hopefully) halt. This is not an appropriate model for neuroscience, and finding an appropriate model will account for some of the work in this project. I am presenting the streaming automata, a real-time version of the well precedented finite automata, to play the role of a computational model in neuroscience. I intend to define this model and prove that it bijects onto computational dynamical systems.

2 The Streaming Automata

A streaming automata M is a five-tuple $(Q, \Sigma, \delta, q_0, w)$ where

1. $Q \subset \mathbb{N}$ is a finite set of states.
2. $\Sigma \subset \mathbb{R}$ is a finite input alphabet.
3. $\delta : Q \times \Sigma \to Q$ is a transition function.
4. $q_0 \in Q$ is a start state.
5. $w : Q \to \mathbb{R}$ is a transduction function.

Let $I(t)$ and $O(t)$ be input and output functions, $I(t) \in \Sigma$, and let $Q(t)$ be a function describing the current state of the streaming automata M.

$$Q(t) = \begin{cases} q0 & \text{if } t = 0 \\ \delta(Q(t-1), I(t)) & \text{otherwise} \end{cases}$$

$O(t) = w(Q(t))$. The transition is both state and input dependent, giving M its limited memory. Because the input is streaming, this form of memory is the only form that can be supported without the machine simply ignoring its input in order to access memory.

3 An Example from Neuroscience

The goal of this section is to present an example streaming automata that models the reduced Hodgkin-Huxley equations (Figure 1). The automata outputs

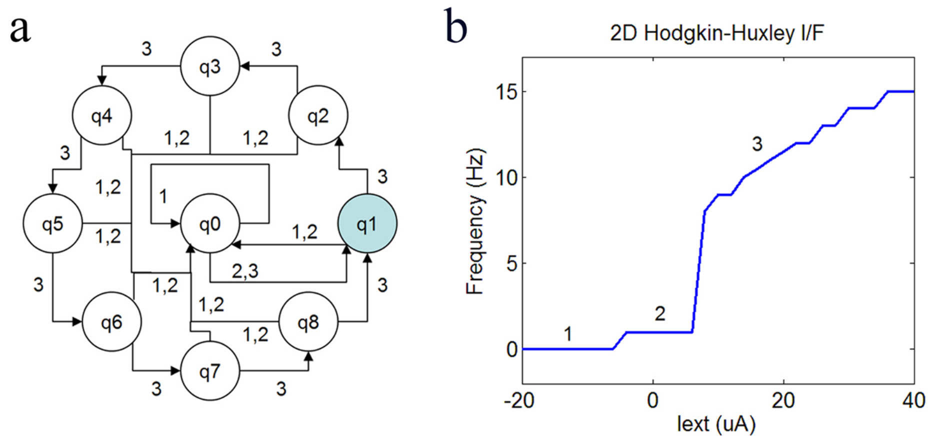

Figure 1: A Reduced Hodgkin-Huxley Streaming Automata. a) A Three input streaming automata that displays the major firing motifs of the reduced Hodgkin-Huxley equations. The system outputs a 1 when in a blue state and a 0 otherwise. b) An I/F plot of a 2D projection of the reduced Hodgkin-Huxley equations. The numbers represent the different firing motifs which are 1. subthreshold dynamics, 2. single spiking, and 3. rhythmic spiking.

in binary, where an output of 1 represents the presence of an action potential. There are three main types of behavior that are observed in the Hodgkin-Huxley reduction. They are: subthreshold dynamics, single spiking and rhythmic spiking. The $q_0 \to q_0$ loop for $I(t) = 1$ models the subthreshold dynamics, the $q_0 \to q_1 \to q_0$ loop for $I(t) = 2$ models single spiking behavior, and the $q_1 \to ... \to q_8 \to q_1$ loop for $I(t) = 3$ represents rhythmic spiking. Frequency resolution can be made arbitrarily high by increasing the number of states in the $q1 - q8$ loop and adding inputs that count the loop with various spacings. Additional frequencies were omitted from the figure to avoid visual clutter.

4 A Bijection Between Streaming Automata and Computational Dynamical Systems

The goal of this section is to demonstrate that there is at least one dynamical system that approximates any streaming automaton and there is at least one streaming automaton that can approximate any of a broad class of biologically plausible dynamical systems. I call these systems 'computational dynamical systems.' The first step to to define computational dynamical systems and to formally define the 'approximate' relation. After that background is done, I will prove that for any streaming automata M and any dynamical system Θ, $M^* \cong \Theta^*$, which is true when $Ap(M, \Theta)$ and $Ap(\Theta, M)$. Therefore, this work consists of two approximation proofs.

The proof sketch below is analogous to a standard technique of showing computational equivalence between models. This is done by showing that each can simulate the other. Computational models are recursive and their space is countable. This allows any two equivalent models to simulate each other absolutely. Because the set of streaming automata is countable and the set of computational dynamical systems is not, I must prove that streaming automata can simulate computational dynamical systems to an error ϵ. This error can be chosen to be arbitrarily small, allowing streaming automata to simulate computational dynamical systems in the limit.

The two sides of the equivalence proof will be presented along with other background steps as a series of lemmas. Many of these lemmas are intuitive and will be stated with only an intuitive proof. The idea is to capture only the least intuitive ideas with formality. This serves to keep the work straightforward and relatively clear.

Computational Dynamical Systems. A computational dynamical system is meant to represent a dynamical system that a user can control with an input signal. There are several requirements that a dynamical system must satisfy for it to have this desirable property. They are as follows:

1. The system is resettable.
2. The system is stable.
3. The system is insensitive to noise.

The system must be resettable so the user has a point of reference for control. The system must be stable because it must be physically realized in an analog machine. And the system must be insensitive to noise because noise cannot be controlled by the user by definition. If these fluctuations make the system unpredicatable, it cannot be used for computation. This last requirement is equivalent to the requirement that the system be non-chaotic.

Approximation. Let $\sigma(n)$ be an input string. For all $n \in \mathbb{N}$, $\sigma(n) \in \Sigma$, let $n\Delta t \leq t < (n+1)\Delta t$ and let $I(t) = \sigma(n)$. Let M be a streaming automaton and Θ be a computational dynamical system. Let $O_M(n, \sigma)$ be the output of M and let $O_\Theta(t, I)$ be the output of Θ.

Given any streaming automaton M, if there exists a computational dynamical system Θ such that $|O_M(n, \sigma) - O_\Theta(n\Delta t, I)| = 0$ we say Θ approximates M and write $Ap(\Theta, M)$.

Given any $\epsilon > 0$ and any computational dynamical system Θ, if there exists a streaming automaton M such that $|O_\Theta(n\Delta t, I) - O_M(n, \sigma)| < \epsilon$ we say M approximates Θ and write $Ap(M, \Theta)$.

Let M^* be the set of streaming automata and Θ^* be the set of computational dynamical systems. For all $M \in M^*$ if there exists $\Theta \in \Theta^*$ such that $Ap(\Theta, M)$, and for all $\Theta \in \Theta^*$ if there exists $M \in M^*$ such that $Ap(M, \Theta)$, we say streaming automata are *computationally equivalent* to computational dynamical systems and write $M^* \cong \Theta^*$.

Lemma 1.1. Let C be a finite set of non-overlapping, continuous and differentiable curves. Each $c \in C$ is of the form $c(r) = (x_1, ..., x_N)$ for bounded r and x_i. Let F^N be the space of continuous, differentiable functions from $\mathbb{R}^N \to \mathbb{R}^N$. Given any C, there exists $f \in F^N$ such that for all fixed α and $c \in C$, $\int_c f(x) \cdot dx = \alpha$.

This lemma states that one can find a function which has a constant value for line integrals along each in a set of bounded curves. This lemma will be used when embedding the streaming automata into the phase-plane of a computational dynamical system.

Lemma 1.2. Let G be a finite graph. $G = \{E, V\}$, where E is a set of edges and V is a set of vertices. For each $v \in V$, let $\vec{p}_v$ be a point in $\mathbb{R}^K$ where $K > 2$. For each $e \in E$, let c_e be a continuous, differentiable curve. If $e = (v_1, v_2)$, c_e has end-points $\vec{p}_v 1$ and $\vec{p}_v 2$. For all finite graphs G, there exist sets of points P and curves C, such that no two curves in C intersect each other. Let this graph embedding be called E_G, where $E_G = \{P, C\}$.

Curves are one-dimensional manifolds. The only N-dimensional space where the union of a finite set of curves can yield a set topologically equivalent to an N-sphere is $\mathbb{R}^2$. Therefore, one cannot enclose any point in $\mathbb{R}^3$ using a finite set of curves. Therefore, there must be a path that connects any two points and does not intersect any of the other curves. This lemma will be used to place a lower bound the number of dimensions needed to embed a streaming automata in the phase-plane of a computational dynamical system.

Lemma 1.3. Let M be a streaming automata. $\delta(q, \sigma)$ can be represented by an $N \times K$ matrix, where $|Q| = N$ and $|\Sigma| = K$. Let $V = Q$ and $E = \{(q, d(q, \sigma)) : q \in Q, \sigma \in \Sigma\}$. $G = \{V, E\}$ is a graph that represents δ. There exists a graph embedding $E_G = \{P, C\}$, where $P \in \mathbb{R}^4$ where first dimension of $\vec{p}_q$ equals $w(q)$ for all $\vec{p}_q \in P$. Consider the embedding $E_{G,\sigma}$ corresponding to the subgraph $G_\sigma = \{V, E_\sigma\}$, where $E_\sigma = \{(q, d(q, \sigma)) : q \in Q\}$. There exists a set of functions f_σ such that for all $c \in C_\sigma$, $\int_c f_\sigma(x) \cdot dx = \alpha$.

This lemma utilizes lemma 1.1 and lemma 1.2. The first dimension of f_σ is going to be the output of a computational dynamical system that represents M. This will ensure that $O_\Theta(n\Delta t) = w(\sigma(n))$. The other three dimensions are needed for the application of lemma 1.2. This will ensure that paths from connected points in P do not cross. Clearly, lemma 1.1 is used to prove that $\int_c f_\sigma(x) \cdot dx = \alpha$ for any constant $\alpha \in \mathbb{R}$.

Lemma 1. For all $M \in M^*$ and $\Delta t > 0$, there exists a computational dynamical system $\Theta(I, X) \in \mathbb{R}^4$ such that $Ap(\Theta, M)$.

Proof. For each $\sigma \in \Sigma$ choose f_σ from lemma 1.3. Choose $\alpha = \frac{1}{\Delta t}$ and let $\Theta(I(n\Delta t), X(t)) = f_\sigma(n)$. $\int_c \Theta(I(n\Delta t), x) \cdot dx = \frac{1}{\Delta t}$. If $\vec{X}_q^\sigma$ and $\vec{X}_{q'}^\sigma \in P$ are the boundaries of c then

$$\begin{aligned}
\Delta \vec{X}_q^\sigma &= \frac{1}{\Delta t}(\vec{X}_{q'}^\sigma - \vec{X}_q^\sigma) \Rightarrow \\
\frac{\Delta \vec{X}_q^\sigma}{\Delta t} &= \vec{X}_{q'}^\sigma - \vec{X}_q^\sigma.
\end{aligned}$$

$q' = \delta(q, \sigma)$ so Θ describes a continuous transition between $\vec{X}_q^\sigma$ and $\vec{X}_{q'}^\sigma$ in Δt. $c \in C$ is defined and only defined for $(q, d(q, \sigma))$ pairs. Therefore Θ transitions with M, and O_Θ transitions with O_M. By induction

$O_M(0, \sigma) = O_\Theta(0\Delta t, I)$
from the definition of Θ.

$O_M(n, \sigma) = O_\Theta(n\Delta t, I) \to O_M(n+1, \sigma) = O_\Theta((n+1)\Delta t, I)$
from above.

Therefore for all n
$O_M(n, \sigma) = O_\Theta(n\Delta t, I) \Rightarrow$
$|O_M(n, \sigma) - O_\Theta(n\Delta t, I)| = 0.$

Therefore for all M, there exists Θ such that $Ap(\Theta, M)$.

Lemma 2.1. Let Θ be an N-dimensional computational dynamical system. Θ is resettable, so there exists $\vec{X}(0) = (x_{0,0}, ..., x_{N-1,0}) \in \mathbb{R}^N$ that is the reset position of the phase-plane. Let $rng\{I\}$ be a finite set of values and fix $\sigma \in rng\{I\}$. Given any $\epsilon > 0$, there exists a finite set of points P_σ and $\vec{p} \in P_\sigma$ such that for all $\vec{X}(n\Delta t)$, $|\vec{X}(n\Delta t) - \vec{p}| < \epsilon$.

Proof. For all n, $\vec{X}(n\Delta t) \in P_\sigma$. $\Theta(\sigma, \vec{X})$ stable and insensitive to noise, which implies that $\vec{X}(t)$ must approach a limit cycle. Therefore, for all $n > N$, there exists $\vec{p}_N \in P_\sigma$ such that $|\vec{X}(n\Delta t) - \vec{p}_N| < \epsilon$. Therefore $|P_\sigma| \leq N$.

Lemma 2.2. Let $I(n\Delta t) = \sigma$ and $I((n+1)\Delta t) = \sigma'$. There exists a set of points $P \in X^N$ such that $\vec{X}(n\Delta t) \in P$ implies that there is a $\vec{p} \in P$ such that $|\vec{X}(n+1) - \vec{p}| < \epsilon$.

Proof. $P = \bigcup_{\sigma \in rng\{I\}} P_\sigma$.

Lemma 2. For all $\Theta \in \Theta^*$ and $\Delta t > 0$, there exists a streaming automaton $M \in M^*$ such that $Ap(M, \Theta)$.

Proof. Let Q be any enumeration f of P where $q_0 = f(\vec{X}(0))$. Let $\Sigma = rng\{I\}$. If $I(n\Delta t) = \sigma$, $f(\vec{X}(n\Delta t)) = q$ and $f(\vec{X}((n+1)\Delta t)) = q'$, $\delta(q, \sigma) = q'$. Let $q = f(\vec{p})$ and $w(q) = p_0$, the output dimension of P. $M = (Q, \Sigma, \delta, q_0, w)$.

The Bijection Theorem. $M^* \cong \Theta^*$.

Proof. This result follows directly from lemma 1 and lemma 2.

Corollary. Any computational dynamical system Θ can be approximated with a computational dynamical system Θ' in $\mathbb{R}^4$.

Proof. For every Θ and $\epsilon > 0$, there exists a streaming automata M that approximates it, from lemma 2. Choose such an M. From lemma 1, there exists a dynamical system $\Theta' \in \mathbb{R}^4$ that approximates M. Choose such a Θ'. For all $n \in \mathbb{N}$, $|O_\Theta(n\Delta t) - O_{\Theta'}(n\Delta t)| < \epsilon$.

5 Discussion

A model of computation is a powerful tool. The logical circuit, the register machine and the Turing machine are all models of computation [5]. Demonstration of the equivalence of these three models of computation provided the theoretical foundation for today's modern computers. Perhaps most important of all, a model of computation gives a construct in which an algorithm can be rigorously defined. I hope that this work is a first step in towards developing a generalized methodology to convert known neuronal circuits into known algorithms. This would be a valuable tool for the field of computational neuroscience.

A more humble and immediate goal for this theory is to provide a universal metric for model fitting. There are many ways to estimate the parameters for a particular set of a dynamical systems and many ways to collect the data for the fit. By converting both the data and the model into streaming automata, one can develop a metric based on computational means. Insofar as neuronal circuits subserve computation, this should serve a universal error metric for parameter estimation in neuroscience.

References

[1] A. Church. The calculi of lambda-conversion. *Annals of Mathematics Studies*, 1941.

[2] T. M. Cover and J. A. Thomas. *Elements of Information Theory, Second Edition.* John Wiley & Sons, Inc., Hoboken, NJ, 2006.

[3] M. Gold. System identification via state characterization. *Automatica*, 8:621–636, 1972.

[4] E. M. Izhikevich. *Dynamical Systems in Neuroscience.* The MIT Press, Cambridge, MA, 2007.

[5] M. Sipser. *Introduction to the Theory of Computation, Second Edition.* Thompson Course Technology, Boston, MA, 2006.

[6] A. M. Turing and J. E. Copeland. *The Essential Turing.* Oxford University Press, Oxford, England, 2004.