Entropy Rate and Statistical Complexity for Dynamical Neurons

Solveig Næss
solveig.nass@nmbu.no

June 2014

Abstract

ϵ-machines can be used to study the dynamics of neural spike trains and reveal spiking patterns. By constructing ϵ-machines we quantify the randomness and structure of three dynamical neuron models: The Linear Integrate and Fire neuron, the Quadratic Integrate and Fire neuron and the Izhikevich neuron.

1 Introduction

The brain's communication system consists of billions of neurons divided into large, complicated networks. The communication between neurons and neural networks appears to be fast, accurate and structured. But what exactly is structure? And can a sequence of action potentials fired from a neuron be said to be a structured process, even though the neuron's input seems to stochastic? And how can structure be measured?

Computational mechanics provides tools for inferring structure from stochastic processes.[4] Specifically we want to describe the neuron spiking models in terms of ϵ-machines, and compute the entropy rate and statistical complexity for each model. Section 2 and 3 describe the dynamics of spiking neurons and give an overview of three specific neuron models: the Linear Integrate and Fire model, the Quadratic Integrate and Fire Model and the Izhikevich model. Section 2 will also explain what we mean by ϵ-machines, entropy rate and statistical complexity. Furthermore two methods for constructing ϵ-machines are considered and used to find entropy rate and statistical complexity for the three neuron models. In short, ϵ-machines are constructable for all 3 neuron models and the Izhikevich neuron model appears to have the most structure.

The Python/ CMPy code used in the project can be found in the shared Sage worksheets.

2 Background

In most neuron models we're interested in capturing the dynamics of the neuron's membrane potential. I.e. the voltage over the cell membrane due to differences in ion concentrations inside vs. outside the cell. These dynamics are determined by complicated chemical processes in the neuron, and can be described very accurately by the Hodgkin Huxley model. Here, the focus will be on simplified models: the Linear Integrate and Fire Model, the Quadratic Integrate and Fire Model and the more complicated Izhikevich Model. These models all capture how a neuron's membrane potential is changing because of inputs from other neurons. When the membrane potential crosses a threshold value, the neuron fires an action potential, it "spikes", and the membrane potential decays back to a resting potential.[1]

The goal for this project is to find ϵ-machines for the dynamical neuron models. The ϵ-machine is a type of unifilar, hidden Markov model of a process, given by the processes' causal states and the state-to-state transition probabilities. Causal states are constructed by grouping together histories that lead to the same prediction of futures. The e-machine is a minimal, unique, optimal predictor and its graphical representation gives a good intuition about a process' dynamics in terms of randomness and structure.[4]

By computing the ϵ-machines for these neuron dynamics we can also directly quantify the randomness and the structure of spiking processes. We'll calculate the randomness or uncertainty of the spike trains in terms of entropy rate h_μ:

$h_\mu = -\sum_{\sigma \in S} \ Pr(\sigma) \ \sum_{\{x\}} Pr(x|\sigma) \ log_2 Pr(x|\sigma)$[3]

Where $\mathcal{S}$ is the set of the causal states σ, $\{x\}$ is the symbol alphabet, in our case $\{x\} = \{0, 1\}$. $Pr(\sigma)$ is the probability distribution of causal states and $Pr(x|\sigma)$ is the probability of a causal state σ emitting the symbol x.

The statistical complexity C_μ is a measure of how structured the process is, in terms of the amount of information stored in the causal states.

$C_\mu = -\sum_{\sigma \in S} \ Pr(\sigma) \ log_2 Pr(\sigma)$ [3]

3 Dynamical System

3.1 Linear Integrate and Fire Neuron

The Linear Integrate and Fire Model, LIF, is a simple model of a spiking neuron, easy to understand by thinking of the membrane as an electric circuit, like the one in figure 1. Here C_m is the membrane capacitance, R_m is the membrane resistance, E_m is the resting potential and I is the input current representing the input from other cells in the neural network.[1, 6] Below threshold the membrane potential V is described by the following differential equation:

$$\begin{aligned} &C_m \frac{dV}{dt} = -\frac{V - E_m}{R_m} + I \\ &\text{if} \quad V \geq V_{threshold}, \quad V \rightarrow E_m[1] \end{aligned}$$

Whenever V reaches the threshold potential $V_{threshold}$, V is reset to E_m. We can think of this as closing the switch in the circuit in figure 1. The input current is a constant value with white noise added to describe the input from thousands of neurons. The voltage curve is shown in figure 2. See caption for parameters.

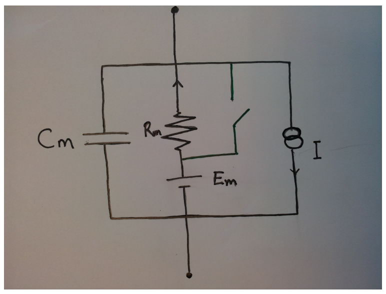


Figure 1: Electric circuit representing the membrane in a Linear Integrate and Fire Neuron. C_m is the membrane capacitance, E_m is the resting potential, R_m is the membrane resistance and I is the input current. [1]

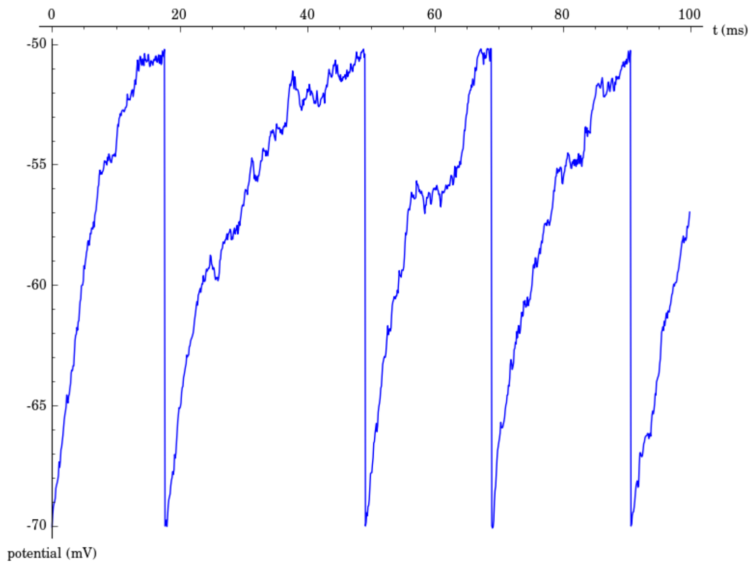


Figure 2: Membrane potential V for the Linear Integrate and Fire neuron with noisy input current I calculated by adding zero-mean white noise with standard deviation= 2 to constant mean current $= 2.2\mu A$. The constants used are: $C_m = 1\mu A$, $R_m = 10k\Omega$, $E_m = -70mV$, $V_{threshold} = -50mV$. [1]

3.2 Quadratic Integrate and Fire Neuron

The Quadratic Integrate and Fire neuron, QIF, is very similar to the LIF, the only difference being the dynamics of the subthreshold potential, as you can see in the equation below. This neuron is a more realistic model of the membrane potential close to the threshold.[1]

$$\begin{aligned} &C_m \frac{dV}{dt} = -\frac{(V - E_m)(V_{threshold} - V)}{R_m(V_{threshold} - E_m)} + I \\ &\text{if} \quad V \geq V_{threshold}, \quad V \to E_m[1] \end{aligned}$$

Figure 3 shows the membrane potential of the QIF. We used the same input current and parameters as for the LIF neuron. For more details see reference [1].

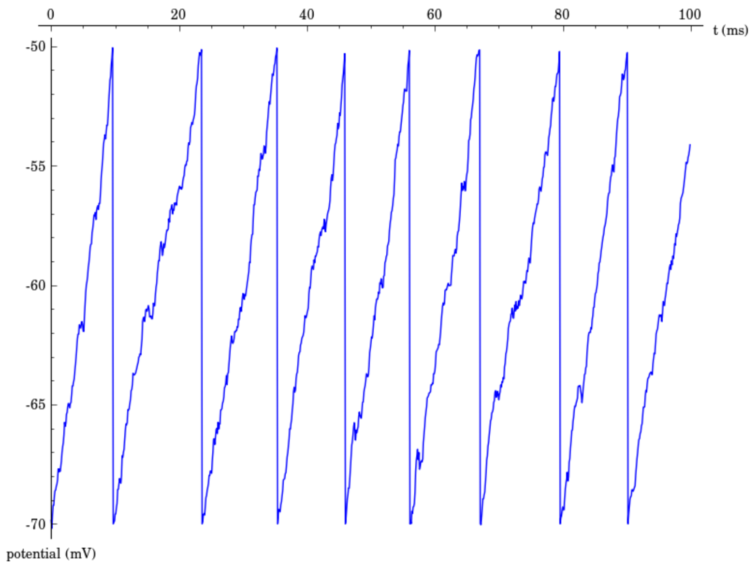


Figure 3: Membrane potential V for Quadratic Integrate and Fire Neuron with noisy input current I calculated by adding zero-mean white noise with standard deviation= 2 to constant mean current $= 2.2\mu A$. The constants used are: $C_m = 1\mu A$, $R_m = 10k\Omega$, $E_m = -70mV$, $V_{threshold} = -50mV$. [1]

3.3 Izhikevich Neuron

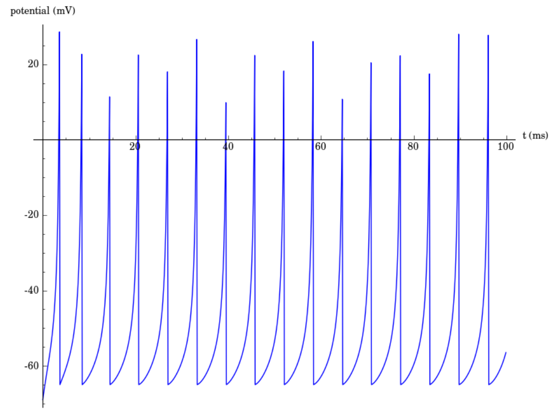
The Izhikevich model is a more realistic version of the quadratic integrate and fire model, consisting of two differential equations.

$$\frac{dV}{dt} = k(V - E_m)(V - V_{threshold}) - u + I$$

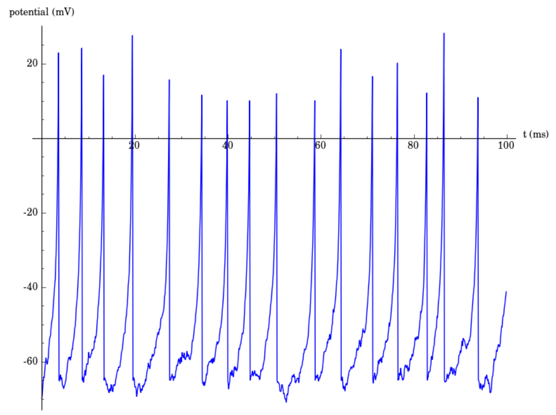
$$\frac{du}{dt} = a(b(V - E_m) - u)$$

$$\text{if} \quad V \geq 30\text{mV:} \quad \begin{cases} V \to c \\ u \to u + d \end{cases} \quad [1]$$

The first is describing the subthreshold potential and is almost identical to the quadratic integrate and fire dynamics, the only difference being a recovery variable u added to the right hand side of the equation. The second equation determines the dynamics of the recovery variable u. The recovery variable is added to describe the dynamics of ion channels which slows down the growth of the membrane potential right after a spike. k, a, b, c and d are constant parameters.[1, 5] For more information about these, see reference [5]. Figure 4 shows Izhikevich voltage curves with and without noise.



(a) Constant input current $I = 10$mA



(b) Noisy input current I calculated by adding zero-mean white noise with standard deviation= 8 to constant mean current $= 10\mu A$

Figure 4: Membrane potential V for Izhikevich Neuron. Constants used in the calculation: $k = 0.04$, $E_m = -70mV$, $V_{threshold}$, $a = 0.02$, $b = 0.2$, $c = -65mV$, $d = 2$. [5]

4 Methods

After generating voltage curves for the different neuron models we want to convert the voltage output into a binary string. This is done by dividing the time axis into same size sample time intervals. For each interval we check whether the membrane potential reaches its threshold or not. If it does, a "1" is added to the string, if not we add a "0". By choosing a sample time significantly less than, but with the same order of magnitude as the mean spike time interval, the binary string captures all spikes as well as the neuron's inter spike intervals of the neuron.

In order to get a better understanding of the spiking processes, we use two different methods to construct ϵ-machines from the binary strings of each neuron model. The first involves construction of parse trees, and the in the second we'll use Bayesian Structural Inference.[2]

A parse tree labeled with output symbols, 0/1, and node-to-node transition probabilities on each arc can easily be constructed from the binary strings. Now we want to define each node's causal state, by looking at future morphs. Future morphs are subtrees of a specified length branching out from the nodes. Two nodes are in the same causal state if they grow identical future morphs. Now the parse tree gives us causal states and the state-to-state transition probabilities and we can construct an ϵ-machine from the spike train.[8]

The parse tree method gives us a best estimate of a model limited by the choice of tree length and the available data. By instead using the Bayesian Strucutural Inference (BSI) method, we can consider a set of plausible model topologies, and calculate the probability of each topology given the model set and a data string. [2] First of all we want to choose a set of plausible models for the data. The binary string outputs from the neuron models considered here are basically telling you how many zeros you see before you see a one. This dynamic can be modeled by the renewal process. We will therefore use a library of renewal processes with different numbers of states as our set of candidate models. After specifying our model set we can use the techniques described in [2] to find a distribution of ϵ-machines.

To quantify the uncertainty and the structure of the neuron dynamics, we compute the entropy rate and the statistical complexity from the ϵ-machines.

As described in the results, one advantage of using the Bayesian Structural Inference to construct ϵ-machines, is that we can use all of the considered machine topologies to compute estimates of h_μ and C_μ and their confidence intervals.

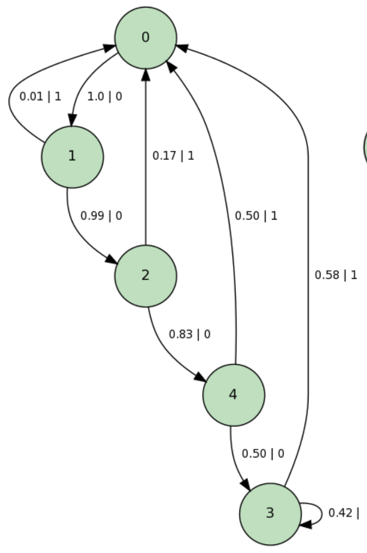
5 Results

We'll start by looking at the ϵ-machines constructed from the parse tree method. Tree length used is 16 for all the models, and the morph length is 3 for all the models except the noisy Izhikevich model, which has morph length 5. Note that the sample times used for constructing ϵ-machines for the different neuron models are not the same. Even though the length of the binary string was increased by a factor 10, and tree and morph lengths were varied, I did not succeed in finding ϵ-machines that were not biased coins or running into dangling state errors. The machines are therefore not directly comparable. See table 3 for comparable ϵ-machines for the different neuron models computed by using the BSI method.

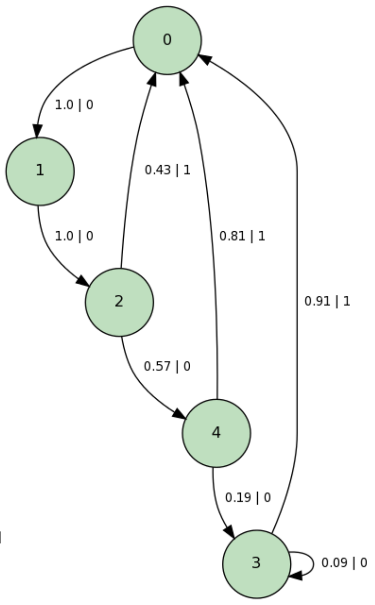
Figure 5a shows the ϵ-machine constructed from the noisy voltage data presented in figure 2 by using a sample time of 5ms. This was the smallest sample time I could use and get results consistent with the binary string using the parse tree method. The results for the QIF-model is shown in figure 5b. Here, the mean inter spike interval was smaller than for the LIF and the smallest sample time giving consistent results was 3ms.

For the Izhikevich neuron I've constructed two ϵ-machines to show the difference between constant and noisy input current. The results are shown in figure 6 and 7 respectively. I've used sample time 1ms for both of them. We can see that the added noise makes the mean inter spike interval larger. It reduces the number of states, however the entropy rate and statistical complexity are very similar.

Inter spike interval, sample time, number of states, entropy rate and statistical complexity for each neuron are summed up in table 1.



(a) LIF: The same, but longer voltage curve from figure 2 is converted into a binary string by using a sampling time 5ms. Parse tree length used was 16 and morph length was 3.



(b) QIF: Same idea as in figure 5a), however the voltage curve from figure 3 is used, and the sampling time was 3ms. Same morph and tree length.

Figure 5: ϵ-machines constructed by using the Parse Tree Method.

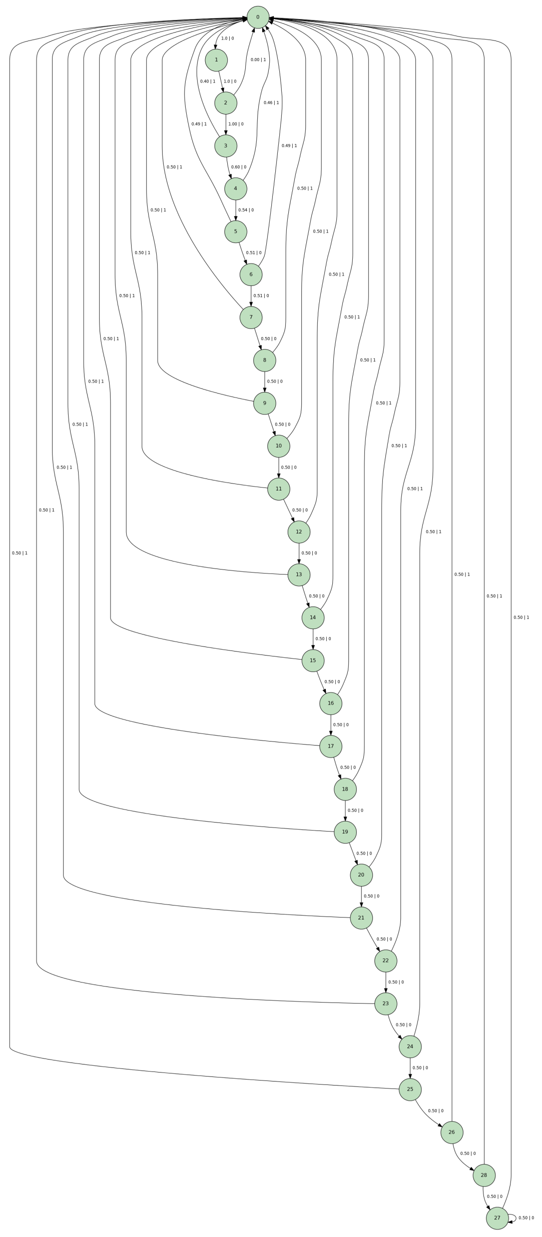


Figure 6: ϵ-machines for the Izhikevich model with constant input current constructed by using the Parse Tree method. Binary input strings are constructed from extended versions of the voltage curves in figure 4a). A sampling time $1ms$ and a tree length 16 and morph length = 3 are used.

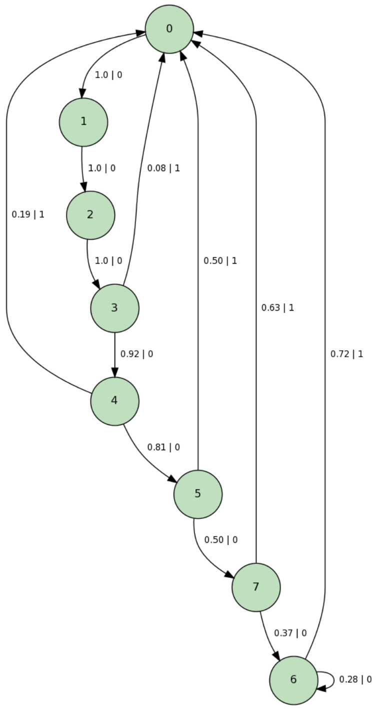
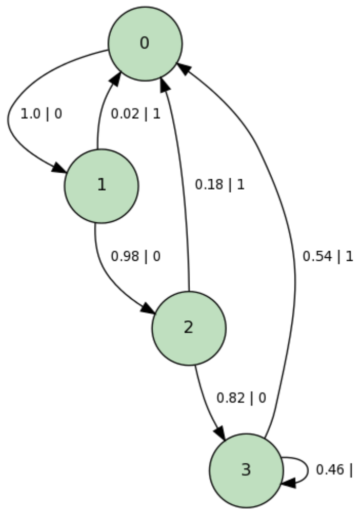


Figure 7: ϵ-machines for the Izhikevich model with white noise added to input current, constructed by using the parse tree method. Binary input strings are constructed from extended versions of the voltage curves in figure 4b). A sampling time 1ms, tree length 16 and morph length = 5 are used.

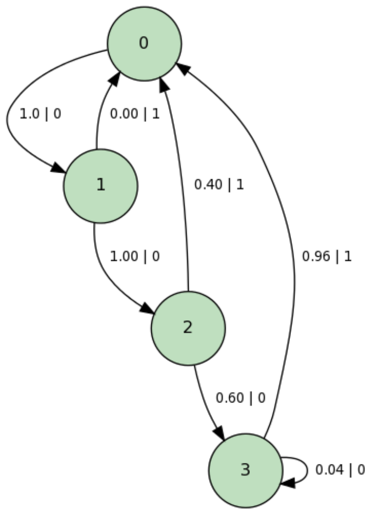
	ISI [ms]	sample time [ms]	number of states	h_μ [bits]	C_μ [bits]
LIF	25	5	5	0.49437	2.30937
QIF	11.11	3	5	0.38801	2.10647
Izhikevich without noise	6.25	1	29	0.42874	2.80775
Izhikevich with noise	5.88	1	8	0.37056	2.86471

Table 1: ϵ-machines measures for the LIF neuron, the QIF neuron and the Izhikevich neurons with and without noise are constructed using the parse tree method.

Using the exact same data strings as for the parse tree method, the ϵ-machines in figure 8 and 9 are sample machines of the most probable models found using Bayesian Structural Inference. The probabilities of these models are included in the figure captions. The prior library consists of renewal processes with 1 to 30 states. The data shown in table 2 is including estimates of h_μ and C_μ and their 95% confidence intervals which are calculated from all 30 model topologies.

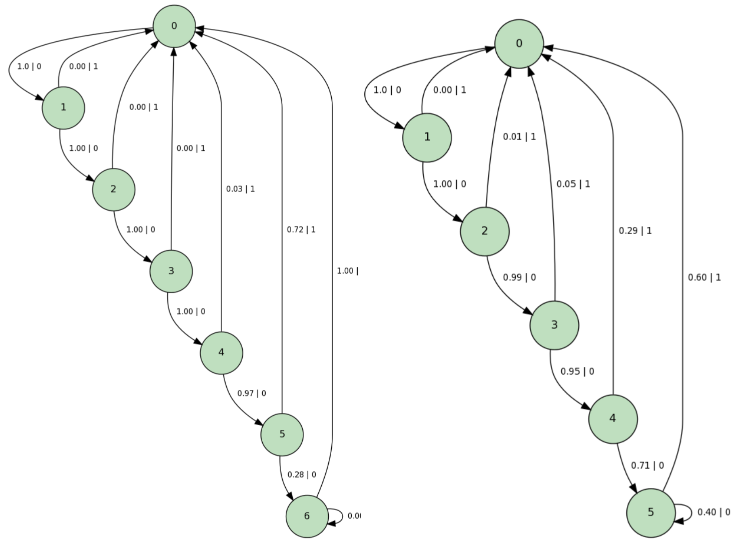


(a) LIF: ϵ-machine topology with the highest probability: 82.13%



(b) QIF: ϵ-machine topology with the highest probability: 99.55%

Figure 8: Bayesian Structural Inference computed ϵ-machines for the Linear Integrate and Fire and the Quadratic Integrate and Fire neurons. The samples shown here are drawn from the distribution of transition probabilities for the most probable topology given by the Bayesian Inference.



(a) Izhikevich neuron with constant input current. Probability of model topology: 95.7%

(b) Izhikevich neuron with noisy input current. Probability of model topology: 52.40%

Figure 9: Bayesian Structural Inference computed ϵ-machines for Izhikevich neurons with constant and noisy current input. The samples shown here are drawn from the distribution of transition probabilities for the most probable topology given by the Bayesian Inference.

	number of states	$E(h_\mu)$ [bits]	CI_{h_μ} [bits]	$E(C_\mu)$ [ms]	CI_{C_μ}
LIF	4	0.48014	(0.43881,0.52428)	2.03668	(1.95935,2.30853)
QIF	4	0.37226	(0.32474,0.43032)	1.97422	(1.95354,1.99050)
Izhikevich without noise	7	0.17601	(0.14641,0.21052)	2.72521	(2.58127,2.74930)
Izhikevich with noise	7	0.38790	(0.36964,0.40677)	2.77328	(2.75936,2.78553)

Table 2: ϵ-machine measures for the LIF neuron, the QIF neuron and the Izhikevich neurons with and without noise are constructed using the BSI method. Note that different sample times are used for the different neurons. For the LIF neuron sample time I used 5ms, 3ms for the QIF neuron and 1ms for both versions of the Izhikevich neuron.

When looking at table 1 and table 2, one can see some of the pros and cons for the parse tree method and the BSI method. The parse tree method does not give us the confidence intervals that we get in table 2. The h_μ's and C_μ's in table 1 are constructed from only one ϵ-machine per neuron model, estimated by the parse tree. Because of the randomness from trial to trial when constructing the voltage curve and the binary strings, the ϵ-machines also vary from trial to trial. However the Parse Tree Method does not require a candidate library and is therefore easier to use. If the candidate library for the BSI really captures all possible models for the data, using the library is a big advantage. Whether our renewal process library represents all relevant topologies is hard to say. However in my work with the parse tree method, only renewal process ϵ-machines has appeared so far, which is indicates that our library is appropriate.

Since the BSI method does not run into inconsistent machines for small sample times, we can use this method to construct comparable ϵ-machines for the dynamics of the three different neuron models. I.e. ϵ-machines all constructed from binary strings where sample time 1ms is used. The results are shown in table 3.

	number of states	$E(h_\mu)$ [bits]	CI_{h_μ} [bits]	$E(C_\mu)$ [ms]	CI_{C_μ}
LIF	16	0.47972	(0.43530,0.52111)	2.03445	(1.96125,2.30682)
QIF	10	0.37219	(0.32454,0.42904)	1.97415	(1.95048,1.99089)
Izhikevich with noise	7	0.38790	(0.36964,0.40677)	2.77328	(2.75936,2.78553)

Table 3: ϵ-machine measures for the LIF neuron, the QIF neuron and the Izhikevich neurons with noise constructed by using the BSI method. Here the sample time used is $1ms$ for all three neuron models.

When comparing the LIF, the QIF and the Izhikevich neuron models in table 3, it appears that Izhikevich is the least random model, and that it also has the most structure with a statistical complexity $C_\mu \approx 2.77$. For comparison a biased coin has 0 structural complexity. The Almost IID process (see reference 0), which is a coupled set of biased coins has a statistical complexity $C_\mu \approx log_2|S|$, where $|S|$ denotes the number of states.[9] A 7-state Almost IID process has a $C_\mu \approx 2.80$ which is larger than but close to the statistical complexity of the Izhikevich model machine with sample time = 1ms.

The QIF neuron model seems to have the least randomness. Note that the input currents (and the amount of noise) for the Izhikevich model and the LIF/QIF models are not the same.

6 Conclusion

From the tables and figures in the results, it is clear that both the LIF, the QIF and the Izhikevich neuron models are structured processes that can be described by ϵ-machines. Among the three the Izhikevich neuron model seems to be the most structured.

Since all these models are well suited for constructing networks of neurons, an interesting next step would be to couple these neuron models together and look at how a neuron in would respond to inputs from a network of neurons.

References

[1] STERRATT, D., GRAHAM B., GILLIES A. AND WILLSHAW D. *Principles of Computational Modelling in Neuroscience.* Cambridge University Press, Cambridge, p.205-215, 2011

[2] STRELIOFF, C.C. AND CRUTCHFIELD, J.P. *Bayesian Structural Inference for Hidden Processes.* Santa Fe Institute Working Paper 13-09-027 arXiv:1309.1392 [stat.ML], 2014

[3] CRUTCHFIELD, J.P. *Between Order and Chaos.* Nature Physics, Insight—Review Article, DOI: 10.1038/NPHYS2190, 2011

[4] SHALIZI, C.R. AND CRUTCHFIELD, J.P. *Computational Mechanics: Pattern and Prediction, Structure and Simplicity.* Journal of Statistical Physics 104, p.819-881, 2001

[5] IZHIKEVICH, E.M., *Simple Model of Spiking Neurons* IEEE TRANSACTIONS ON NEURAL NETWORKS, VOL. 14, NO. 6, 2003

[6] IZHIKEVICH, E.M. AND EDELMAN, G.M. *Large-scale model of mammalian thalamocortical systems* Proceedings of the national academy of sciences 105 (9), p.3593-3598, 2008

[7] IZHIKEVICH, E.M., *Which Model to Use for Cortical Spiking Neurons?* IEEE TRANSACTIONS ON NEURAL NETWORKS, VOL. 15, NO. 5, 2004

[8] CRUTCHFIELD, J.P. *Lecture Note 22: The Learning Channel* from PHY256B: Natural Computation and Self Organization, 2014

[9] CRUTCHFIELD, J.P. *Lecture Note 25: Measures of Structural Complexity* from PHY256B: Natural Computation and Self Organization, 2014