Índice general

Part 1

1. Introduction

2. Neuron Dynamics

Spiking neural network's neuron model describe how the membrance potential of a neuron change over the time. In this chapter, we dive into the biological neuron's dynamics and exhibit several examples neuron models in the spiking neural network.

2.1 Biological Neuron

2.2 Neuron Model Abstract and Taxonomy

In spiking neural network, Equation 2.2.1 to Equation 2.2.4 provide a highly abstract description for modeling a biological neuron. In these equation, $\mathbf{x}_t$ is the state vector at time t, Δx_t is the state variation at time t, $\mathbf{u}_t$ is the input vector at time t, V is the membrane potential of the neuron, $\mathbf{y}_t$ is the neuron's output at time t, and V_{out} is the output voltage that will be sent to the synapses that depature from this neuron. $(\mathbf{x}_t)_i$ is the i-th element of the state vector.

$$\Delta \mathbf{x}_t = f(\mathbf{x}_t, \mathbf{u}_t) \tag{2.2.1}$$

$$\mathbf{y}_t = g(\mathbf{x}_t, \mathbf{u}_t) \tag{2.2.2}$$

$$s.t., \ (\exists i \in [0, |\mathbf{x}_t|])(\mathbf{x}_t)_i = V \tag{2.2.3}$$

$$(\exists i \in [0, |\mathbf{y}|])_i \mathbf{y} = V_{out} \tag{2.2.4}$$

These equations are in the form of a **state-space model (SMM)**.
Equation 2.2.3 indicates that, a neuron should maintain a membrane potential V, and Equation 2.2.4 indicates that, the neuron's output should contains a voltage V_{out}.

2.2.1 Current-based Neuron

In a current-based neuron model, the synaptic input is represented as an injected current directly added to the membrane potential equation. we have the input $\mathbf{u}_t$ contains the current I_t, and $\mathbf{x}_t$ contains the membrance potential V_t.

We will later to see that a current-based LIF neuron (Section 2.3.2) can hold a form of Equation 2.2.5.

$$\dot{V}_t = \frac{1}{\tau_m}(-(V_t - V_{rest}) + I_t) = f_{CurrentLIF_1}(V_t, \mathbf{u}_t = [I_t]) \tag{2.2.5}$$

2.2.2 Conductance-based Neuron

In a conductance-based neuron model, the synaptic input is modeled by changing the conductance of the membrane, which then affects the current flow.
In the state space representation, the input $\mathbf{u}_t$ is the synapse conductance. let $\mathbf{u}_t = g_{syn}(t)$, $\mathbf{x}_t = V_t$, we will see that in conductance-based LIF neuron model, it hold Equation 2.2.6

$$\dot{V}_t = \frac{1}{\tau_m}(-(V_t - V_{rest}) + g_syn(t)(E_{syn} - V_t)) = f_{ConductanceLIF_1}(V_t, \mathbf{u}_t = g_{syn}(t)) \tag{2.2.6}$$

2.3 Neuron Model Examples

2.3.1 Hodgkin-Huxley (HH) Model

The Hodgkin-Huxley (HH) model is a conductance-based model , which can be utilize to accurately reproduce the bio-neuron's dynamics. Its form is shown in Equation 2.3.1 to Equation 2.3.5. Combine all these equations, we get Equation 2.3.6.

$$I = C_m \frac{dV_m}{dt} + I_i \tag{2.3.1}$$
$$I_i = I_{Na} + I_K + I_l \tag{2.3.2}$$
$$I_{Na} = g_{Na}(V_m - V_{Na}) \tag{2.3.3}$$
$$I_K = g_K(V_m - V_K) \tag{2.3.4}$$
$$I_l = \bar{g}_l(V_m - V_l) \tag{2.3.5}$$

$$I_l = C_m \frac{dV_m}{dt} + g_{Na}(V_m - V_{Na}) + g_K(V_m - V_K) + \bar{g}_l(V_m - V_l) \tag{2.3.6}$$

Ion channel function $g_\cdot$ are function respect to time t and membrance potential V. Specifically, Equation 2.3.7 is held.

$$g_{Na} = \bar{g}_{Na} m^3 h \qquad g_K = \bar{g}_K n^4 \qquad g_l = \bar{g}_l \tag{2.3.7}$$

Combine Equation 2.3.1 to Equation 2.3.7, we get Equation 2.3.8

$$I_l = C_m \frac{dV_m}{dt} + \bar{g}_{Na} m^3 h(V_m - V_{Na}) + \bar{g}_K n^4 (V_m - V_K) + \bar{g}_l(V_m - V_l) \tag{2.3.8}$$

$\frac{d\cdot}{dt} = \alpha_\cdot(V_m)(1-\cdot) - \beta_\cdot(V_m)\cdot$ is held. Where $\cdot$ is a placeholder for m, n and h. As such, Equation 2.3.9 to Equation 2.3.11 are held.

$$\frac{dn}{dt} = \alpha_n(V_m)(1-n) - \beta_n(V_m)n \tag{2.3.9}$$
$$\frac{dm}{dt} = \alpha_m(V_m)(1-m) - \beta_m(V_m)m \tag{2.3.10}$$
$$\frac{dh}{dt} = \alpha_h(V_m)(1-h) - \beta_h(V_m)h \tag{2.3.11}$$
$$\tag{2.3.12}$$

From experiment, we have Equation 2.3.13 to Equation 2.3.18.

$$\alpha_n(V_m) = \frac{0{,}01(10-V)}{exp(\frac{10-V}{10})-1} \tag{2.3.13}$$

$$\alpha_m(V_m) = \frac{0{,}1(25-V)}{exp(\frac{25-V}{10})-1} \tag{2.3.14}$$

$$\alpha_h(V_m) = 0{,}07exp(-\frac{V}{20}) \tag{2.3.15}$$

$$\beta_n(V_m) = 0{,}125exp(-\frac{V}{80}) \tag{2.3.16}$$

$$\beta_m(V_m) = 4exp(-\frac{V}{18}) \tag{2.3.17}$$

$$\beta_h(V_m) = \frac{1}{exp(\frac{30-V}{10})+1} \tag{2.3.18}$$

Hodgkin-Huxley could be seen as a current-based neuron model, which may represent by space state model, with $\mathbf{x}_t = [V_t, m_t, h_t, n_t]$, and $\mathbf{u}_t = I_t$.

2.3.2 Leaky Integrate-and-fire Model

Leaky Integrate-and-fire model is a computational effective model, in which a threshold is set, when membrance potential cross the threshold, the neuro emit a spike. In implementation, we may set the voltage output at time t, V_{out} be 1 mV. The scale problem has the potential to be solved by automatically adjusting the synapses' weights in trainning. A LIF neuron i's output form a **spike train** $S_i(t) = \sum_m \delta(t-t_m)$.

$$\dot{V}_t = \frac{1}{\tau_m}(-(V_t - V_{rest}) + g_s yn(t)(E_{syn} - V_t)) \tag{2.3.19}$$

2.3.3 Izhikevich Model

2.3.4 FitzHugh-Nagumo Model

2.3.5 Morris-Lecar Model

2.3.6 Hindmarsh-Rose Model

2.3.7 Cable theory

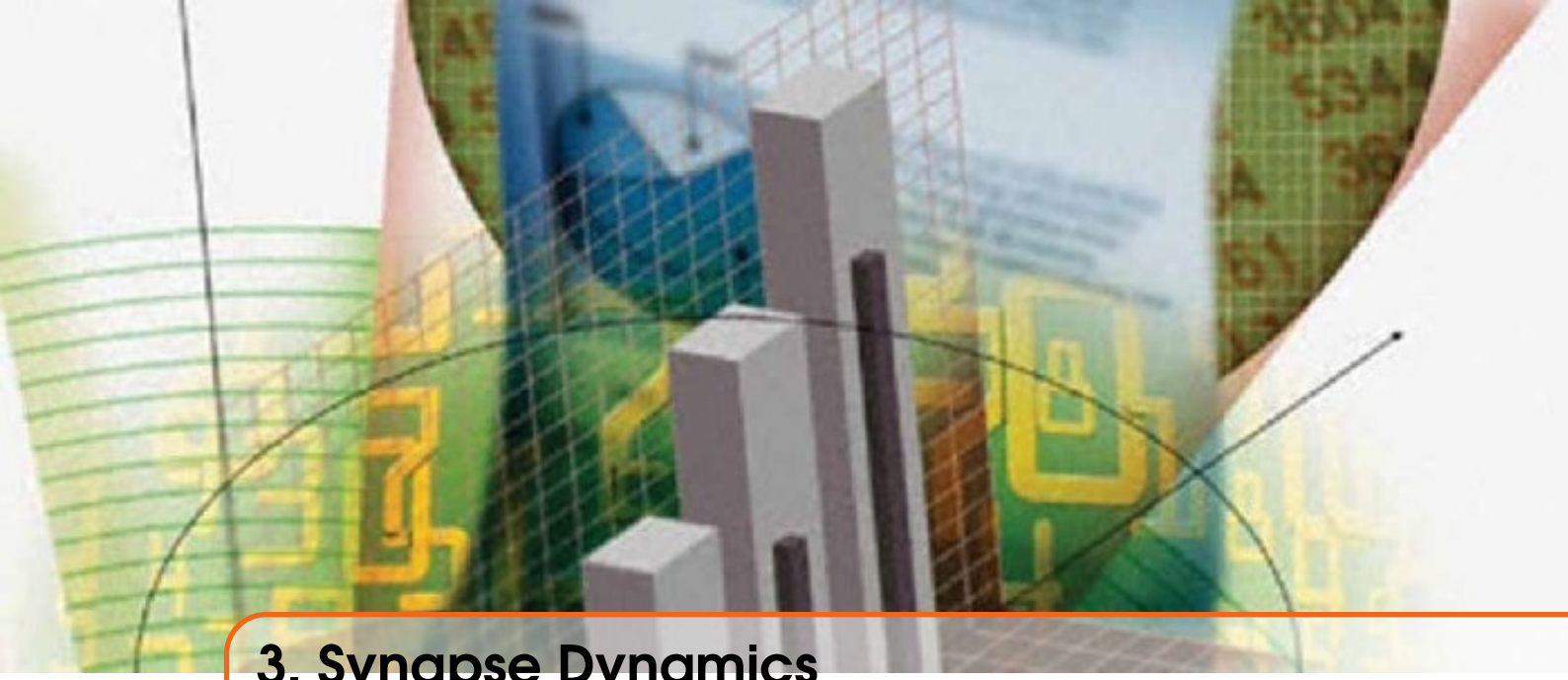
2.3.8 Perfect Integrate-and-fire

2.3.9 Adaptive Integrate-and-fire

2.3.10 Fring Rate Model

2.3.11 Discussion

Spike Representation You may note that, although different neuron models have different dynamics and spike representations, they can still communicate with each other through synapses. Some neuron models require current input, while others do not. Nonetheless, spikes can transmit over synapses and cause current variations across them by utilizing a common output element V_{out}.
One challenge that may arise is the normalization of spike representations. To address this issue, one approach is to standardize the spike events by converting all spike representations into binary form. Another approach can involve adjusting the synaptic weights. Although we may face scaling issues with different types of spike representations, these can be mitigated during training by appropriately adjusting the synapse weights.



3. Synapse Dynamics

4. Trainning Algorithms

4.1 Unsupervior Learning

4.1.1 Spike-timing-dependent plasticity (STDP)

4.1.2 Growing Spiking Neural Networks

4.1.3 Artola, Bröcher, Singer (ABS) rule

4.1.4 Bienenstock, Cooper, Munro (BCM) rule

4.1.5 Relationship between BCM and STDP rules

4.2 Supervised Learning

4.2.1 STDP-based Methods

Supervised STDP (SSTDP)

Spike-Timing-Dependent Plasticity (STDP) with Supervision

4.2.2 Spike-Timing Dependent Backpropagation (STDBP)

4.2.3 Liquid State Machine (LSM) and Readout Training

4.2.4 SpikeProp

Extension (McKennoch et al., 2006; Booij and tat Nguyen, 2005; Shrestha and Song, 2015; de Montigny and Mâsse, 2016; Banerjee, 2016; Shrestha and Song, 2017).
spike timing based methods is that they cannot learn starting from a quiescent state of no spiking.
Bohte (2011)
Huh and Sejnowski (2017)

4.2.5 ReSuMe

Related Work (Sporea and Grüning, 2013) Pfister et al. (2006) Gardner et al. (2015) Fremaux et al. (2010)

4.2.6 SuperSpike

SuperSpike [**super-spike**] is a supervised learning algorithm dedicated to deterministic Leaky Intergrate-and-Fire neuron model. While the backpropagation algorithm used in traditional neural

network cannot directly be used in the training of spiking neural network, the author provide a surrogate gradient-based method to tackling with the problems facing in solving the $S_i/\partial w_{ij}$, where S_i is the i-th neuron's spike train, and w_{ij} is the connection weight from neuron j to neuron i.

Specifically, $S_i(t) = \sum_k \delta(t - t_k)$, where t_k is the k-th spike emission time, and $\delta(\cdot)$ is the dirac delta function.

Approaches approximate the partial derivative of the hidden unit output by $f(S_{pre}, g(V_{post}))$.

Let $\hat{S}_i$ be the target spike train of neuron i. The cost model for optimization that make $\hat{S}_i$ approach the real S_i hold the form: $L = \frac{1}{2}\int_{-\infty}^{t} ds[(\alpha * \hat{S}_i - \alpha * S_i)(s)]^2$.

α is a normalized smooth temporal convolution kernel. The original SuperSPike use *double exponential causal kernel*.

$$\partial L/\partial w_{ij} = -\int_{-\infty}^{t} ds[(\alpha * \hat{S}_i - \alpha * S_i)(s)](\alpha * \frac{\partial S_i}{\partial w_{ij}})(s)$$

.

Some existing methods for tackling the term $\frac{\partial S_i}{\partial w_{ij}}$: (1) making derivation directly to the membrance voltage, (2) introducing noisy which render the likelihood of $\langle S_i \rangle$ a smooth function of the membrance potential.

The superspike convert calculation of $\frac{\partial S_i}{\partial w_{ij}} \to \sigma'(U_i)\frac{\partial U_i}{\partial w_{ij}}$. In which U_i is the membrance voltage. Original superspike choose $\sigma(U)$ be the negative side of a fast sigmoid. This function is objective to increase steeply and peak at the spiking threshold. Other monotonic functions may also work.

For current-based LIF models the membrane potential $U_i(t)$ can be written in integral form as a spike response model (SRM0 (Gerstner et al., 2014)): $U_i(t) = \sum_j w_{ij}(\varepsilon * S_j(t)) + (\eta * S_i(t))$

In which, ε corresponds to the postsynaptic potential (PSP) shape, η captures spike dynamics and reset.

The existence of term $(\eta * S_i(t))$ make us difficult to perform derivation. Now, $U_i(t) \approx (\varepsilon * S_j(t))$.

The gradient calculation for a weight become:

$$\frac{\partial w_{ij}}{\partial t} = r\int_{-\infty}^{t} dse_i(s)\alpha * (\sigma'(U_i(s))(\varepsilon * S_j)(s))$$

.

r is the learning rate, $e_i(s) \equiv \alpha * (\hat{S}_i - S_i)$, $\lambda_{ij} = \alpha * (\sigma'(U_i(s))(\varepsilon * S_j)(s))$ is the eligibility trace. The form above is also known as *non-vanishing surrogate gradient*.

The neuron model utilized by the SuperSpike is

$$\tau^{mem}\frac{dU_i}{dt} = (U^{rest} - U_i) + I_i^{syn}(t)$$

. And synapse evolution model is

$$\frac{d}{dt}I_i^{syn}(t) = -\frac{I_i^{syn}(t)}{\tau^{syn}} + \sum_{j\in pre} w_{ij}S_j(t)$$

.

5. Network Architecture Investigation

5.1 Liquid Neural Network

State Space Models (SSM)
Linear State Space Models
Structured State Space Model (S4) [lnn-s4]
Liquid Time Constant (LTC) Model [ltc]
Linear Liquid Time Constant Model
High-order Polynomial Projection Operator (HiPPO)
Normal Plus Low-Rank (NPLR)
Scaled Legendre Measure (LegS)
HiPPO Matrix
black-box Cauchy Kernel
Coupled Bilinear
Dynamics Causal Model
Liquid structure state space model [**lnn-4s**] introduce the liquid time constant into the structure state model. Specifically it hold the form in Equation 5.1.1.

$$\frac{d\mathbf{x}(t)}{dt} = -[\mathbf{A} + \mathbf{B} \otimes f(\mathbf{x}(t), \mathbf{u}(t), t, \theta)] \otimes \mathbf{x}(t) + \mathbf{B} \otimes f(\mathbf{x}(t), \mathbf{u}(t), t, \theta) \tag{5.1.1}$$

A linear version could be expressed by the form of Equation 5.1.2.

$$\dot{\mathbf{x}}(t) = [\mathbf{A} + \mathbb{I}_N \mathbf{B} u(t)]\mathbf{x}(t) + \mathbf{B}u(t),\ y(t) = \mathbf{C}\mathbf{x}(t) \tag{5.1.2}$$

The output y_k for input u_0 to u_k can be represented by $y = \bar{K} * u + \bar{K}_{liquid} * u_{correlation}$. In which the $*$ is the convolution operator. This equation divide the calculation of y_k to the calculation of the kernel $\bar{K}$, and the convolution between the kernal and input, and the convolution between the kernal and input correlation.

Índice alfabético

P

R