Temporal Coding in Spiking Neural Networks with Alpha Synaptic Function (Review)

Jethro Kuan

September 1, 2019

Outline

Agenda

What are Spiking Neural Networks?

- NN architectures that mimics biological neural networks:
  - SNN computes with asynchronous spikes that signal the occurrence of a characteristic event

Motivation:

- low power consumption
- analog computation
- fast inference
- event-driven processing
- online learning
- parallelism

Spike Response Model (SRM) [2]

Model for membrane potential:

$$u_i(t) = \eta(t - \hat{t}_i) + \sum_j \sum_f \epsilon_{ij}(t - t_j^{(f)}) + u_{rest} \tag{1}$$

- Spikes come from the dendrites (input neurons), and this voltage accumulates
- Voltage decays slowly to resting potential
- Upon exceeding threshold, spike, and enter refractory period

Spiking, illustrated

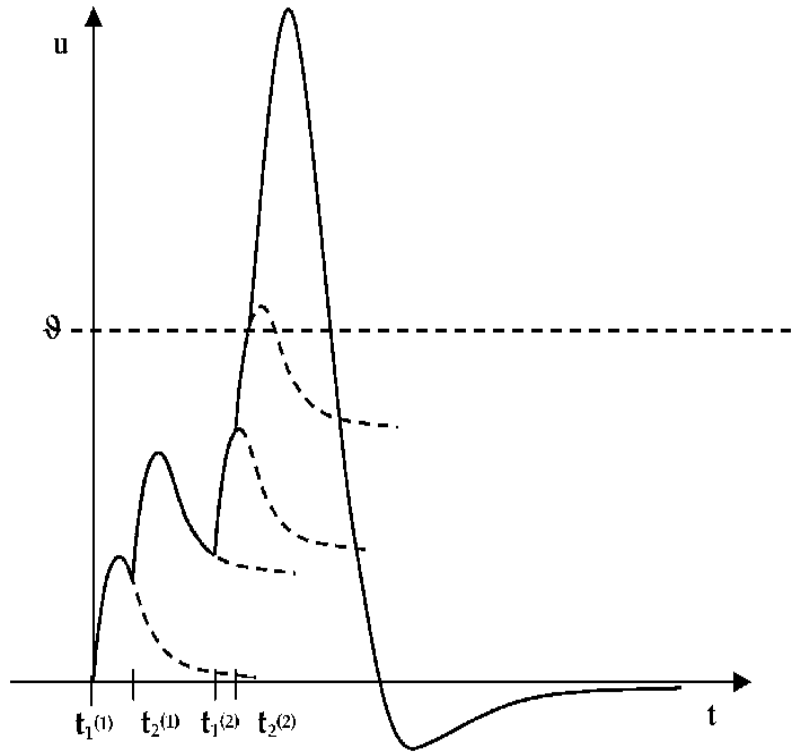

Figure: Membrane potential over time. Source: [3]

Spike Trains

- Sequence of `(spike, timestamp)`

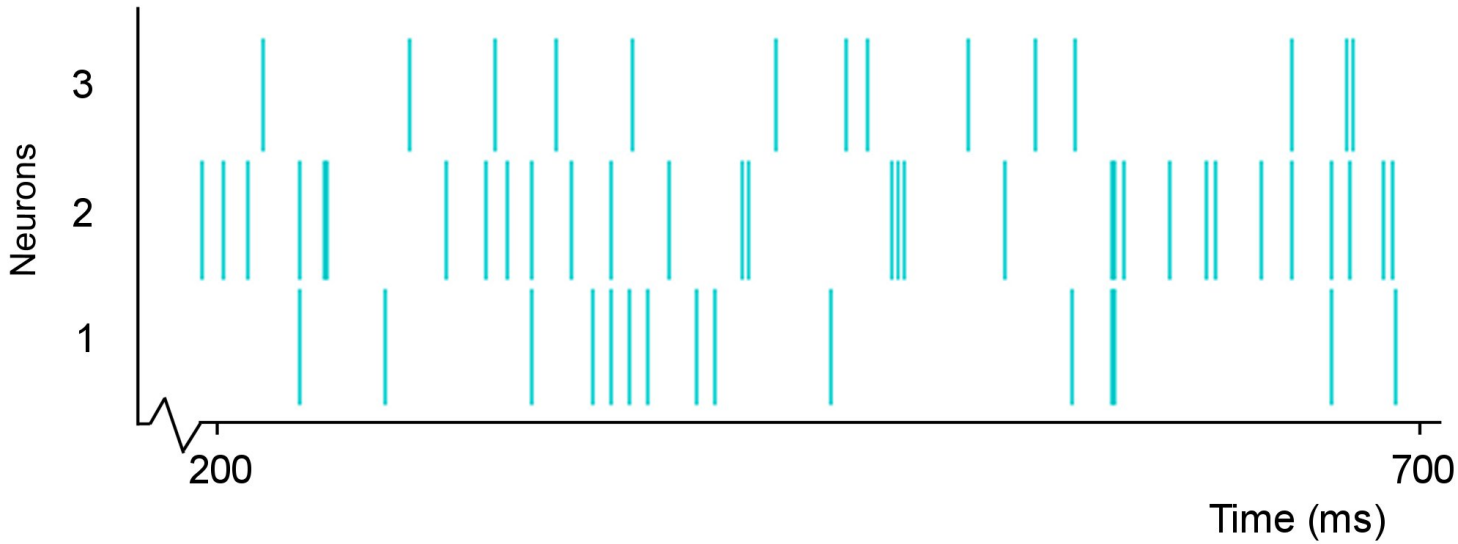

Figure: Spike Trains for 3 neurons

How Do Neurons Encode Information? I

Rate-coding

- Windowed average across:
  - single neuron
  - multiple runs
  - population of neurons

Problems:

- Windowed average $\rightarrow$ minimum latency (brain has to wait for average)
- Research shows brain can act on single spikes

Temporal-coding

- Time to first spike
- Phase
- Correlations and Synchrony

How Do Neurons Encode Information? II

Which coding scheme is better?

- Both codes are consistent: if the mean firing rate of a neuron is high, then the time to first spike is expected to occur early
- Rate codes discard temporal information
- For more information see [2]

What's the landscape for SNNs?

- Pretty bad.
- Most SNNs cannot be trained with gradient-based methods, because there are no gradients
- The current approach to training SNNs include:
  - Binarization of ANNs
  - Conversion from ANNs
  - Training of constrained networks
  - Supervised learning with spikes
  - Local learning rules at synapses
- Exception: probabilistic SNNs define outputs as jointly distributed random binary processes. The joint distributions are differentiable in the synaptic weights, and one can use principled learning criteria from ML and information theory

Agenda

Motivation

1. Atemporal networks (think LSTMs) don't have the benefits of encoding information directly in the temporal domain
   - 1.1 They remain sequential (require all previous layers of computation to produce answer)
   - 1.2 Information in the real world are typically temporal

Key Ideas

1. Temporal Coding: Information is encoded in the relative timing of neuron spikes. Using temporal coding allows shift of differentiable relationship into the temporal domain.
   1.1 Find differentiable relationship of the time of postsynaptic spike with respect to the weights and times of the presynaptic spikes.
2. Alpha synaptic transfer function: Use the SRM, but with the exponential decay of form te^{-t}.
3. Synchronization pulses: input-independent spikes, used to facilitate transformations of the class boundaries.

The Coding Scheme

More salient information about a feature is encode as an earlier spike in the corresponding input neuron (think time-to-first-spike). In a classification problem with m inputs and n possible classes:

input spike times of m input neurons

output index of output neuron that fires first (among the n output neurons)

Alpha Synaptic Function

Incoming exponential synaptic kernels are of the form $\epsilon(t) = \tau^{-1} e^{-\tau t}$ for some decay constant τ. Potential of membrane in response to the spike is then $u(t) = te^{-\tau t}$. It has a gradual rise, and slow decay.

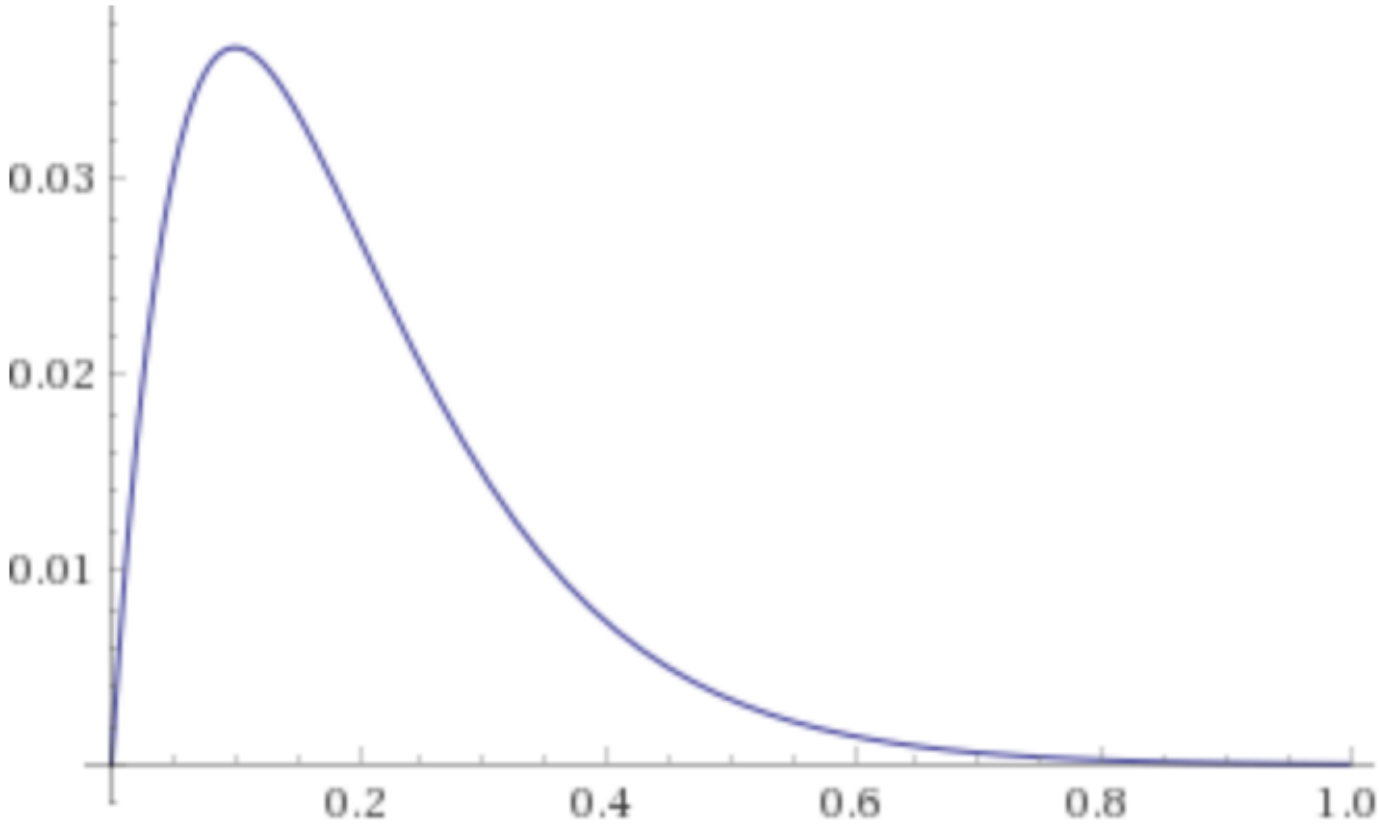

Figure: Plot of $y = xe^{-10x}, x \in [0, 1]$

Modelling Membrane Potential I

The membrane potential is a weighted sum of the presynaptic inputs:

$$V_{mem}(t) = \sum_i w_i(t - t_i)e^{\tau(t_i - t)} \tag{2}$$

We can compute the spike time t_{out} of a neuron by considering the minimal subset of presynaptic inputs $I_{t_{out}}$ with $t_i \leq t_{out}$ such that:

$$\sum_{i \in I_{t_{out}}} w_i \left(t_{out} - t_i\right) e^{\tau(t_i - t_{out})} = \theta \tag{3}$$

has 2 solutions: 1 on rising part of function and another on decaying part. The spike time is the earlier solution.

Solving for the Equation

Let $A_I = \sum_{i \in I} w_i e^{\tau t_i}$, and $B_I = \sum_{i \in I} w_i e^{\tau t_i} t_i$, we can compute:

$$t_{out} = \frac{B_I}{A_I} - \frac{1}{\tau} W \left(-\tau \frac{\theta}{A_I} e^{\tau \frac{B_I}{A_I}} \right) \tag{4}$$

where W is the Lambert W function.

The Loss Function I

The loss minimizes the spike time of the target neuron, and maximizes the spike time of non-target neurons (cross-entropy!)
Softmax on the negative values of the spike times o_i (which are always positive):

$$p_j = \frac{e^{-o_j}}{\sum_{i=1}^{n} e^{-o_i}} \tag{5}$$

The cross entropy loss $L(y_i, p_i) = -\sum_{i=1}^{n} y_i \ln p_i$ is used.
Changing the weights of the network alters the spike times. We can compute the exact derivative of the post synaptic spike time wrt any presynaptic spike time t_j and its weight w_j as:

$$\frac{\partial t_{out}}{\partial t_j} = \frac{w_j e^{t_j} \left(t_j - \frac{B_l}{A_l} + W_l + 1 \right)}{A_l (1 + W_l)} \tag{6}$$

The Loss Function II

$$\frac{\partial t_{out}}{\partial w_j} = \frac{e^{t_j}\left(t_j - \frac{B_I}{A_I} + W_I + 1\right)}{A_I(1 + W_I)} \tag{7}$$

where

$$W_I = W\left(-\frac{\theta}{A_I}e^{\frac{B_I}{A_I}}\right) \tag{8}$$

Synchronization Pulses

These act as a temporal form of bias, adjusting class boundaries in the temporal domain. Per network, or per layer biases are added. Spike times for each pulse are learned with the rest of the parameters of the network.

Hyperparameters

Table 1: Hyperparameters of the model. The first column shows the default parameters chosen to solve Boolean logic problems. The second column shows the search range used in the hyperparameter search. Asterisks (*) mark ranges that were probed according to a logarithmic scale; all others were probed linearly. The last column shows the value chosen from these ranges to solve MNIST.

Parameter	**Default value (Boolean tasks)**	**Search range**	**Chosen value (MNIST)**
`batch_size`	1	$[1, 1000]^*$	5
`clip_derivative`	100.0	$[1, 1000]$	539.7
`batch_constant` (τ)	1.0	$[0.1, 2]$	0.181769
`fire_threshold` (θ)	1.0	$[0.1, 1.5]$	1.16732
`learning_rate`	0.001	$[10^{-5}, 1.0]^*$	$10^{-4} \times 2.01864$
`learning_rate_pulses`	0.001	$[10^{-5}, 1.0]^*$	$10^{-2} \times 5.95375$
`n_hidden`	1×2	$[0, 4] \times [2, 1000]^*$	1×340
`n_pulses`	1	$[0, 10]$	10
`nonpulse_weight_mean_multiplier`	0.0	$[-10, 10]$	-0.275419
`penalty_no_spike`	1.0	$[0, 100]$	48.3748
`pulse_weight_mean_multiplier`	0.0	$[-10, 10]$	7.83912

* - logarithmic search space

Agenda

Boolean Logic Problems

Inputs encoded as individual spike times of two input neurons. All spikes occur between 0 and 1. True and False values are drawn from distributions $[0.0, 0.45]$ and $[0.55, 1.0]$ respectively.
Trained for maximum of 100 epochs, 1000 training examples.
Tested on 150 randomly generated test examples. 100% accuracy on all problems.

Non-convolutional MNIST I

784 neurons of the input layer corresponding to pixels of the image. Darker pixels encoded as earlier spike times. Output of network is the index of the earliest neuron to spike.
Trained with evolutionary-neural hybrid agents. Best networks achieved 99.96% and 97.96% accuracy on train and test sets.
The network learns two operating modes: slow-regime and fast-regime. Operating in the slow regime has higher accuracy, but takes more time. Fast regime makes quick decisions, with the first spike in the output layer occurring before the mean spike in the hidden layer.

Non-convolutional MNIST II

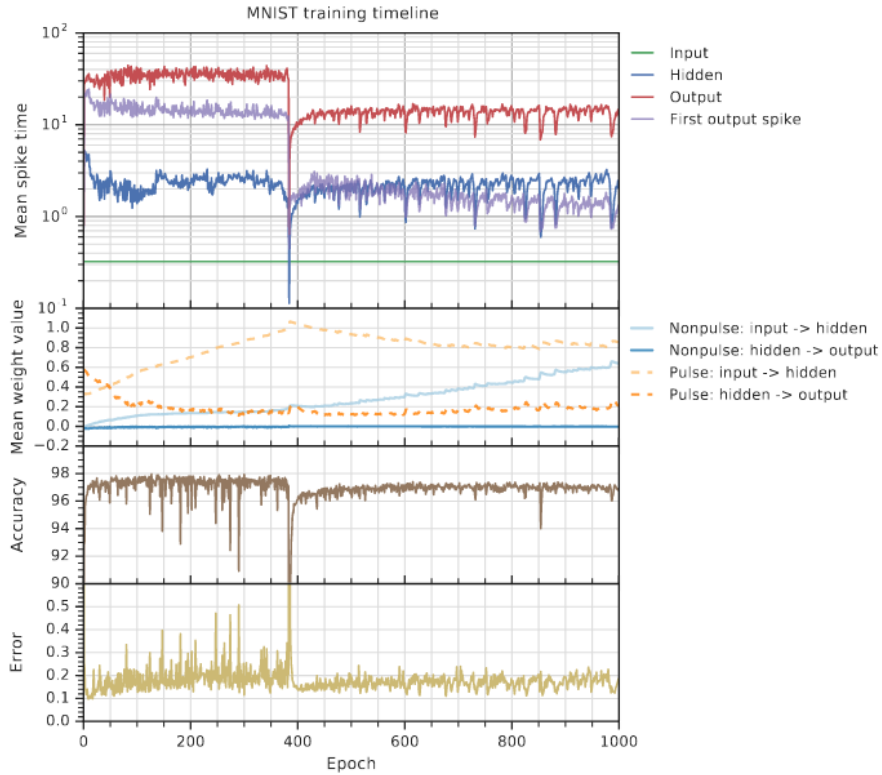

Figure 5: Learning dynamics during the training of a spiking network with the best set of hyperparameters. The plots show a change in regime at epoch 384 from slow to fast classification. All statistics are shown for the test set.

Agenda

Training the Model

```
Ignoring n_inputs flag for MNIST problem. Using 784 inputs.
Network architecture: [784, 340, 10]
Sync pulses: [0.0909091, 0.181818, 0.272727, 0.363636, 0.454545,
Saving the model that performs best on validation set.
Loading MNIST data...
Done loading MNIST data.
Loading MNIST data...
Done loading MNIST data.
Using ThreadPool with 16 threads.

run 0        epoch 0        train_error 0.97        train_acc.%

run 0        epoch 1        train_error 0.38        train_acc.%

run 0        epoch 2        train_error 0.29        train_acc.%
```

Testing the Models

```
[nix-shell:~/projects/ihmehimmeli/build]$ tempcoding/tempcoding_
W2019-09-01T21:48:01.040269665+08:00 /home/jethro/projects/ihmeh
```

Agenda

My Thoughts

Little information is lost with the temporal encoding scheme, so I'd expect the spiking neural network to perform well. Especially so, since there are gradients and gradient-based methods have already proven to be reliable.
Will augmenting gradients for a spiking neural network be useful in this scenario? Can we meta-learn for algorithms like STDP or equilibrium propagation instead?

Agenda

References I


Iulia M. Comsa, Krzysztof Potempa, Luca Versari, Thomas Fischbacher, Andrea Gesmundo, and Jyrki Alakuijala.
Temporal coding in spiking neural networks with alpha synaptic function.
CoRR, 2019.


Wulfram Gerstner and Werner M Kistler.
Spiking neuron models: Single neurons, populations, plasticity.
Cambridge university press, 2002.


Murilo Saraiva de Queiroz, Roberto Coelho de Berrêdo, and Antônio de Pádua Braga.
Reinforcement learning of a simple control task using the spike response model.
Neurocomputing, 70(1-3):14–20, 2006.