Improving eligibility propagation using Izhikevich neurons in a multilayer RSNN.

Presentation 3: Implementing TIMIT

Werner van der Veen
(w.k.van.der.veen.2@student.rug.nl)

November 10, 2020

Where things stand

- ☑ Simulate LIF, ALIF and Izhikevich neuron pairs in e-prop simulation, and observe STDP-like weight change.
- ☑ Make multilayered spiking recurrent neural network.
- ☑ Prepend the TIMIT dataset reader to the pipeline.
- ☐ Include validation sets.
- ☐ Implement Bellec's tricks. Should be able to reproduce thereafter:
  - ☐ L2 & firing rate regularization
  - ☐ Firing rate regularization
  - ☐ Gaussian distribution for broadcast weights
  - ☐ Adam optimizer
- ☐ I will enable the Izhikevich neurons and increase the number of layers (*).
- ☐ Implement long-term synaptic scaling in Izhikevich neurons.
- ☐ Implement metaplasticity.

Work done since previous meeting

- TIMIT preprocessing and data handling.
- Wrote the outer loops processing the epochs and batches.
  The whole system is now essentially a nested loop:
  (1) Epochs of batches;
  (2) Batches of series;
  (3) Series of time points;
  (4) Layers to process each time point.

TIMIT preprocessing

X: From `.wav` to $\mathbf{x}(t) \in \mathbb{R}^{39}$.

Y: From `.phn` to $\mathbf{y}(t) \in \mathbb{R}^{61}$.

(1) The `.wav` is sampled at $SR = 16\text{kHz}$. Every 10ms (160 samples), we take a sample frame of 25ms (400 samples). The goal is to obtain 39 Mel-frequency cepstral coefficients (MFCCs) for each such frame.

(2) Calculate the periodogram estimate of the power spectrum for each frame.

(3) Apply a Mel-scaled filterbank to the power spectra and sum the energy in each filter.

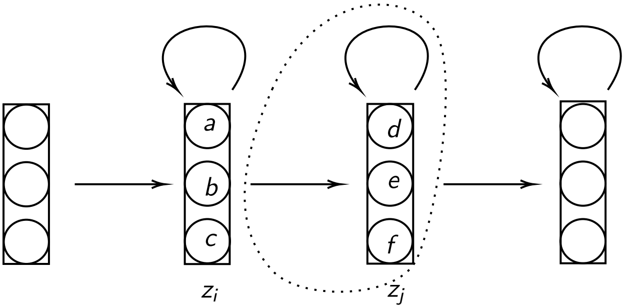
(4) Take the discrete cosine transform (DCT) of the logarithm of the energies.

(5) Only keep DCT coefficients 2–13.

(6) Compute the first and second derivatives of the coefficients.

(7) Parse the phonemes from the raw `.phn` and encode them in a one-hot vector that aligns with $\mathbf{x}(t)$.

Questions



$$W = \begin{array}{c} \\ d \\ e \\ f \end{array} \begin{array}{c} \begin{array}{cccccc} a & b & c & d & e & f \end{array} \\ \begin{pmatrix} W_{da} & W_{db} & W_{dc} & 0 & W_{de} & W_{df} \\ W_{ea} & W_{eb} & W_{ec} & W_{ed} & 0 & W_{ef} \\ W_{fa} & W_{fb} & W_{fc} & W_{fd} & W_{fe} & 0 \end{pmatrix} \end{array}$$

Questions

– Input: now converting signal to Bernoulli spikes, and then feeding them to layer 0.