Pattern recognition using latency coding in multilayer spiking neural networks

Brian Gardner, André Grüning

Department of Computer Science, University of Surrey, UK

1. Objectives

- **To establish an efficient learning algorithm** as applied to multilayer spiking neural networks for pattern recognition.
- **To investigate temporal codes** as a means to rapidly identify patterns that are based on spike times.

2. Background

Few learning rules exist for spiking networks that are as technically efficient and versatile in their deployment as backpropagation is for rate-coded networks, and yet take full advantage of rapid temporal coding.

To address this, we propose a supervised method for training multilayer spiking networks to identify patterns based on a time-to-first spike decoding scheme. This generalises our previous formulation in [1] to real-world data sets.

3. Method

Approach. We start by considering a winner-take-all competitive learning scheme for a feed-forward spiking neural network, such that the output neuron with the earliest spike, and hence strongest activation, decides the input class.

Cost function. Specifically, each output neuron's activation is described by a softmax function of first-spikes, $\{\tau_i\}$, i.e.:

$$a_i^L = \frac{\exp\left(-\tau_i\right)}{\sum_{i'} \exp\left(-\tau_{i'}\right)}$$

for the $i\text{-}th$ neuron in the last layer, L. Hence, assuming a one-hot encoded target output vector, $\boldsymbol{y} = \{y_1, y_2, \ldots, y_c\}$ for c different classes, then a suitable choice of cost function for network optimisation is given by the cross-entropy:

$$C(\mathbf{y}, \mathbf{a}^L) = -\sum_i y_i \log a_i^L$$

4. Learning algorithm

Error signal. Neurons in the network are treated as leaky-integrate-and-fire (LIF) for analytical tractability. Thus, by linearising output neuron voltages close to their thresholds [2], gradient descent yields the following error signal:

$$\delta_i^L := y_i - a_y^L$$

Weight update formulae. The above error signal thus defines weight updates for the last and second-last network layers:

$$\Delta w_{ij}^L \propto \delta_i^L \sum_f \epsilon(\tau_i - t_j^f)$$

$$\Delta w_{ij}^{L-1} \propto \sum_{i'} w_{i'i}^L \delta_{i'}^L \sum_f \epsilon(\tau_{i'} - t_i^f) \sum_g \epsilon(t_i^f - t_j^g)$$

where ε is the shape of an evoked postsynaptic potential, and t^f corresponds to the $f\text{-}th$ firing time of a neuron.

5. Simulations

Learning task. The performance of the derived learning algorithm was tested through simulations of multilayer spiking networks trained on two benchmark classification datasets: Iris and Wisconsin. Iris consists of 150 samples split between three classes, and Wisconsin 699 samples split between two classes. The latter two classes of Iris are not linearly separable from each other.

Network setup. Networks contained a single hidden layer of 20 neurons, with the number of output neurons matched to the number of input classes. Input patterns were encoded as spikes using receptive fields [2], a method that transforms real-valued features into spike latencies. The number of input neurons encoding Iris and Wisconsin were 48 and 63, respectively. The setup is depicted in Fig. 1.

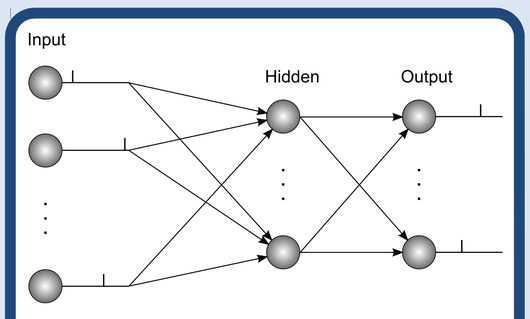

Fig. 1. Architecture of the feed-forward multilayer spiking neural network, tasked with classifying data based on output spike latencies.

Input neurons: encode each sample as spike times, via receptive fields, with latencies of up to 9 ms.

Hidden neurons: work to learn sample features.

Output neurons: classify samples according to which neuron responds with the first output spike.

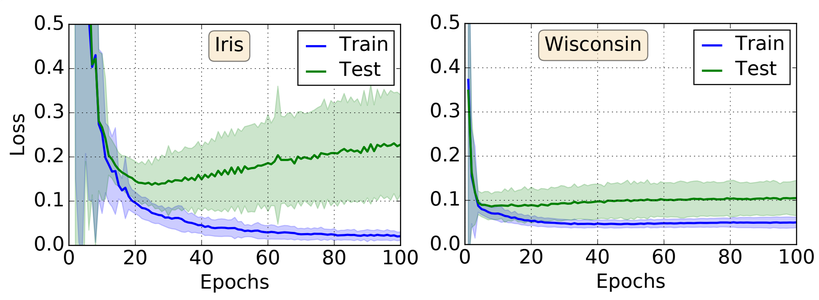

Fig. 2. Training and test loss with the number of training epochs, using three-fold cross-validation on Iris and Wisconsin.

(Left) The Iris dataset, for a 48x20x3 structured network.

(Right) The Wisconsin dataset, for a 63x20x2 structured network.

Results were averaged over 10 independent runs, where shaded regions indicate the standard deviation.

Network loss. The network's performance on Iris and Wisconsin was tested using three-fold cross validation, when trained for up to 100 epochs using mini-batch learning (Fig. 2). To maximise the network's test accuracy, the following techniques were used:

- L2 regularisation of weights to discourage extremal values.
- Synaptic scaling to sustain network activity in all layers.
- Early-stopping where the test loss has reached its minima.

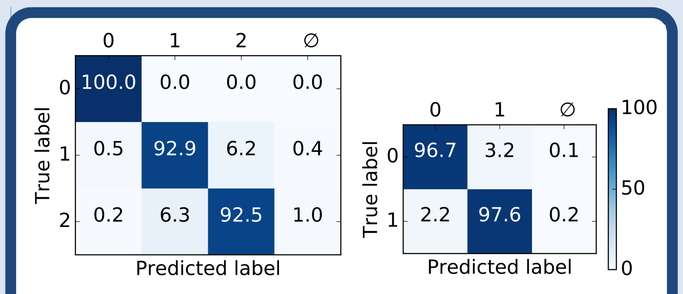

Fig. 3. Confusion matrices for the two datasets, as test accuracy.

(Left) The Iris dataset, for a network trained for 30 epochs.

(Right) The Wisconsin dataset, for a network trained for 20 epochs.

The symbol ∅ indicates no classification made by the network.

Network accuracy. The network's accuracy on the two datasets was obtained by applying early-stopping during training; The test loss was minimised after 30 and 20 epochs on Iris and Wisconsin, respectively. The test accuracies (%) obtained in this way are shown in Fig. 3. To summarise:

- The final Iris training and test accuracies were 98 ± 1 % and 95 ± 3 %, respectively.
- The final Wisconsin training and test accuracies were 98.0 ± 0.5 % and 97 ± 1 %, respectively.

Predictions on Iris and Wisconsin. The activity of the network when forming predictions on test samples belonging to different classes was visualised as input / hidden spike rasters and output voltage traces, for up to 40 ms. Figures 4 and 5 depict the network states when processing samples belonging to the Iris and Wisconsin datasets, respectively. Note that although each output neuron tends to fire more than once it is only the first output spike which drives weight updates in the network, as well as deciding the input class.

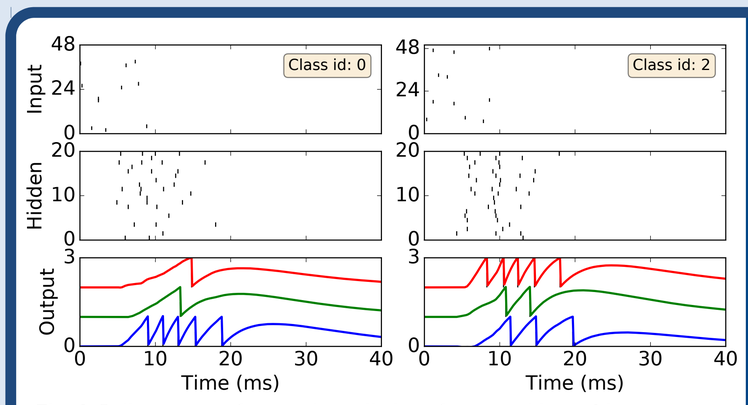

Fig. 4. Spike rasters of network predictions for two arbitrary Iris test samples, belonging to classes 0 and 2.

Top and middle rows show spike rasters of input and hidden activity, respectively. The bottom row displays the voltage trace of each output neuron.

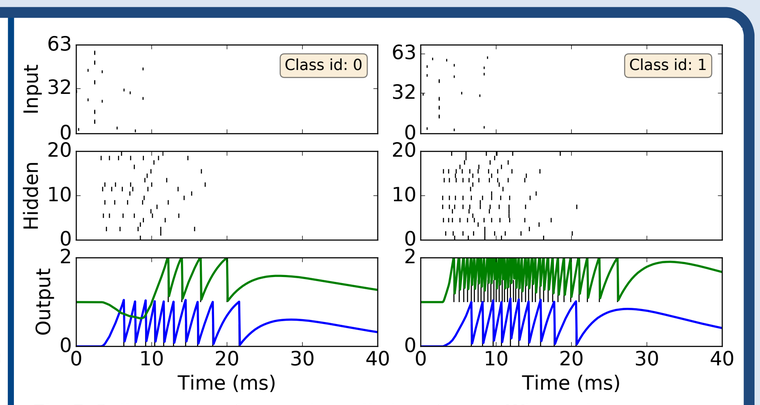

Fig. 5. Spike rasters of network predictions for two Wisconsin test samples, belonging to classes 0 and 1.

Similar figure layout as in Fig. 4. The high firing rate of the neuron responding to the second sample results from its strong input drive to fire first.

References

[1] Gardner, B., Sporea, I., & Grüning, A. (2015). Learning spatiotemporally encoded pattern transformations in structured spiking neural networks. Neural Computation, 27(12), 2548-2586.

[2] Bohte, S. M., Kok, J. N., & La Poutre, H. (2002). Error-backpropagation in temporally encoded networks of spiking neurons. Neurocomputing, 48(1-4), 17-37.

[3] Zenke, F., & Ganguli, S. (2018). SuperSpike: Supervised Learning in Multilayer Spiking Neural Networks. Neural Computation, 30(6), 1514-1541.

Poster online link:

Acknowledgement

This project has received funding from the European Union's Horizon 2020 Framework Programme for Research and Innovation under the Specific Grant Agreement No. 785907 (Human Brain Project SGA2).

Human Brain Project

Co-funded by the European Union

Conclusions

- These results provide a proof-of-concept for a time-to-first-spike decoding scheme in multilayer networks, as applied to example real-world datasets.
- This approach is capable of processing data on a very fast time scale: typically within just 10 ms.
- Our method complements previous studies on learning sequences of precisely-timed output spikes in multilayer spiking neural networks [1, 3].
- As part of the steps involved in deriving this learning algorithm, hidden neurons were required to spike stochastically; with respect to future work it would be interesting to explore the impact of variable spiking on network regularisation.