ESANN'2000 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 26-28 April 2000, D-Facto public., ISBN 2-930307-00-5, pp. 419-424

SpikeProp: Backpropagation for Networks of Spiking Neurons

Sander M. Bohte[1], Joost N. Kok[2] and Han La Poutré[1]

[1]CWI, Kruislaan 413, 1090 GB Amsterdam, P.O. Box 94079, The Netherlands
[2]LIACS, Leiden University, 2300 RA Leiden, P.O. Box 9512, The Netherlands

Abstract. For a network of spiking neurons with reasonable post-synaptic potentials, we derive a supervised learning rule akin to traditional error-back-propagation, *SpikeProp* and show how to overcome the discontinuities introduced by thresholding. Using this learning algorithm, we demonstrate how networks of spiking neurons with biologically plausible time-constants can perform complex non-linear classification in fast temporal coding just as well as rate-coded networks. When comparing the (implicit) number of neurons required for the respective encodings, it is empirically demonstrated that temporal coding potentially requires significantly less neurons.

1 Introduction

Ever since the work of Rashevsky and others [5] in the early sixties, the real-valued output of a neuron is assumed to be its average firing-rate. However, increasingly attention has been turned to the possibility of information coding in the timing of individual action potentials (spikes).
For networks of spiking neurons with multiple delayed synapses, Natschläger & Ruf [4] have described a powerful unsupervised learning algorithm based on a temporal version of Hebbian learning. To study the computational power of such networks without the constraints associated with Hebbian learning, we derive an error-backpropagation (BP) algorithm for networks of spiking neurons analogous to the work by Rumelhart et al. [6]. To deal with the discontinuous nature of spiking neurons, we approximate the thresholding function, which is validated for small learning rates. The algorithm is capable of learning complex non-linear tasks in spiking neural networks in much the same fashion as traditional sigmoidal neural networks. This is demonstrated by the classical and extended XOR classification tasks, as well as for real-world datasets. Thus, networks of spiking neurons with biologically plausible time-constants can perform complex non-linear classification in a fast temporal encoding just as well as rate-coded networks. Importantly, this requires significantly less neurons in the extended XOR-problem as compared to fast rate-coding as in [3].

2 Error-Backpropagation in a Network of Spiking Neurons

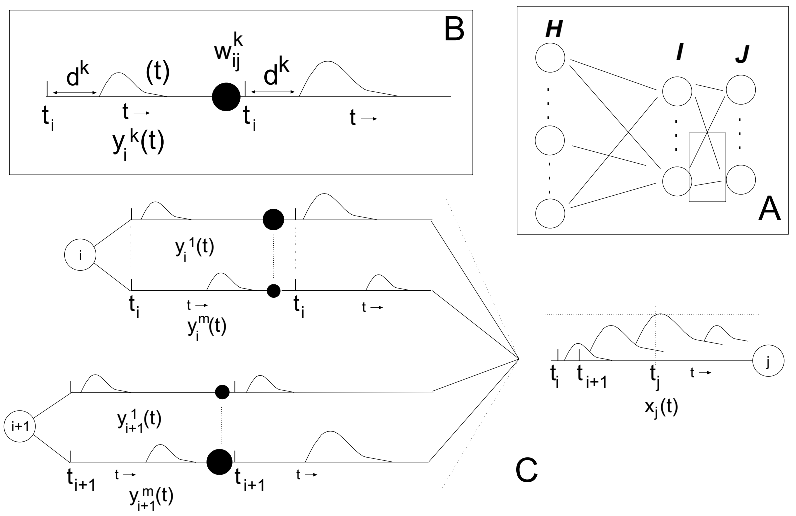

Figure 1 Network configuration and connectivity of a spiking neural network. A) A multi-layer feedforward network. In B) a single synaptic connection: the delayed pre-synaptic potential is multiplied by the synaptic efficacy w_{ij}^k to obtain the post-synaptic potential. C) Two multi-synapse connections: for every connection, the post-synaptic potentials from all delayed synaptic terminals are summed to obtain the membrane potential x_j. A spike at t_j is generated in neuron j when x_j exceeds threshold θ.

For a network of spiking neurons with multiple delayed synaptic terminals (as described in [4]), we derive error-backpropagation, analogous to the derivation by Rumelhart et al. [6]. As described in [2], leaky integrate-and-fire neurons are modeled where the incoming, weighted post-synaptic potentials are added up, and a spike is generated when the excitatory input exceeds the threshold (depicted in figure 1). Error-backpropagation equations are derived for a fully connected feedforward network with layers labeled H(input), I(hidden) and J(output), however the derivation works equally well for networks with more hidden layers. Each individual connection consists of a fixed number (m) of synaptic terminals, where each terminal serves as a sub-connection that is associated with a different delay. The delay d^k is defined by the difference between the post-synaptic firing time and the time the pre-synaptic potential starts rising (figure 1B). In the derivation, we treat each synaptic terminal as a separate connection with independent weights. A spike event is modeled as a pre-synaptic potential, which is weighted by the synaptic efficacy of the terminal to obtain the post-synaptic potential.
Formally: the pre-synaptic input of neuron $i \in I$ to neuron $j \in J$ is described as

ESANN'2000 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 26-28 April 2000, D-Facto public., ISBN 2-930307-00-5, pp. 419-424

the sum of synaptic contributions:

$$y_i(t) = \sum_k^m y_i^k(t) \tag{1}$$

with m the number of delays and $y_i^k(t)$ representing a delayed pre-synaptic potential (PSP) for each terminal.
The post-synaptic input x_j of neuron j receiving input from neurons i can then be described as the weighted sum of the pre-synaptic input:

$$x_j(t) = \sum_i \sum_k w_{ij}^k y_i^k(t), \tag{2}$$

where w_{ij}^k denotes the weight associated with synaptic terminal k. The firing time t_j of neuron j is determined as the first time when the post-synaptic input crosses the threshold ϑ: $x_j(t) \geq \vartheta$.
The target of the algorithm is to learn a set of target firing times, denoted $\{t_j^d\}$, at the output neurons for a given set of input patterns $\{P[t_1..t_i]\}$. We choose for the error function the least mean squares error function, but other choices like entropy are also possible. Given desired spike times $\{t_j^d\}$ and actual firing times $\{t_j\}$, this function is defined as:

$$E = \frac{1}{2}\sum_j (t_j - t_j^d)^2. \tag{3}$$

For error-backpropagation, we need to calculate:

$$\Delta w_{ij}^k = -\eta \frac{\partial E}{\partial w_{ij}^k} \tag{4}$$

with η the learning rate and w_{ij}^k the weight connecting neuron i to neuron j with delay d_k. The derivative in the right hand part of (4) can be expanded to:

$$\frac{\partial E}{\partial w_{ij}^k} = \frac{\partial E}{\partial t_j}\frac{\partial t_j}{\partial w_{ij}^k} = \frac{\partial E}{\partial t_j}\left.\frac{\partial t_j}{\partial x_j(t)}\right|_{t=t_j}\left.\frac{\partial x_j(t)}{\partial w_{ij}^k}\right|_{t=t_j}. \tag{5}$$

In the last t wo right-handed terms, we express t_j as a function of the thresholded post-synaptic input $x_j(t)$ around $t = t_j$. We assume here that for an ε region around $t = t_j$, x_j can be approximated by a linear function of t. For such a small region we can thus approximate the threshold function as $\delta t_j = -\delta x_j(t_j)/\alpha$, where α equals the local derivative of x_j with respect to t: $\left.\frac{\partial x_j(t)}{\partial t}\right|_{t=t_j}$. Thus, the second right hand term in (5) evaluates to:

$$\left.\frac{\partial t_j}{\partial x_j(t)}\right|_{t=t_j} = \frac{-1}{\alpha} = \frac{-1}{\left.\frac{\partial x_j(t)}{\partial t}\right|_{t=t_j}} = \frac{-1}{\sum_{i,l} w_{ij}^l \left.\frac{\partial y_i^l(t)}{\partial t}\right|_{t=t_j}}. \tag{6}$$

Note that this approximation implies that we can use only small learning rates (see also the XOR-example in section 3). In further calculations, we will write terms like $\left.\frac{\partial x_j(t)}{\partial t}\right|_{t=t_j}$ as $\frac{\partial x_j(t_j)}{\partial t_j}$.
The other terms in (5) can easily be calculated, and (4) evaluates to:

$$\Delta w_{ij}^k(t_j) \quad = \quad -\eta \frac{y_i^k(t_j) \cdot (t_j^d - t_j)}{\sum_{i,l} w_{ij}^l \frac{\partial y_i^l(t_j)}{\partial t_j}}. \tag{7}$$

For convenience, we define δ_j:

$$\delta_j \equiv \frac{\partial t_j}{\partial x_j(t_j)} \frac{\partial E}{\partial t_j} = \frac{(t_j^d - t_j)}{\sum_{i,l} w_{ij}^l \frac{\partial y_i^l(t_j)}{\partial t_j}}. \tag{8}$$

We now continue with the other layers: for error-backpropagation in other layers than the output layer, the generalized delta error in layer I is defined by:

$$\begin{aligned} \delta_i &\equiv \frac{\partial t_i}{\partial x_i(t_i)} \frac{\partial E}{\partial t_i} = \frac{\partial t_i}{\partial x_i(t_i)} \sum_j \frac{\partial E}{\partial t_j} \frac{\partial t_j}{\partial x_j(t_j)} \frac{\partial x_j(t_j)}{\partial t_i} \\ &= \frac{\partial t_i}{\partial x_i(t_i)} \sum_j \delta_j \frac{\partial x_j(t_j)}{\partial t_i} \end{aligned} \tag{9}$$

Here, we apply the same approximated chain rule as in (5), albeit for $t = t_i$. Simple calculations yield:

$$\delta_i = \frac{\sum_j \delta_j \{\sum_k w_{ij}^k \frac{\partial y_i^k(t_j)}{\partial t_i}\}}{\sum_{h,l} w_{hi}^l \frac{\partial y_h^l(t_i)}{\partial t_i}}. \tag{10}$$

Thus, for a hidden layer, the weight adaptation reads:

$$\Delta w_{hi}^k = -\eta \frac{y_h^k(t_i) \sum_j \{\delta_j \sum_k w_{ij}^k \frac{\partial y_i^k(t_j)}{\partial t_i}\}}{\sum_{n,l} w_{ni}^l \frac{\partial y_n^l(t_i)}{\partial t_i}} \tag{11}$$

In this derivation, a critical approximation of the post-synaptic potential $x_j(t)$ around t_j is made. This obviously only holds for small perturbations, hence only for small learning rates.

3 XOR

The classical example of a non-linear problem requiring hidden units to perform the transformation required is the exclusive-or (XOR) problem. Associating a "0" with firing time "late" and a 1 with firing time "early", we propose the following version of this problem for time-coded networks:

Input Patterns		Output Patterns
0 0	$\to$	16
0 6	$\to$	10
6 0	$\to$	10
6 6	$\to$	16

The numbers represent spike times in milliseconds; a third input neuron is added, which always fired at $t = 0$ to designate the reference start time (otherwise the problem becomes trivial).
For the network we used the feed-forward network described, with connections with a delay interval of 15 ms, such that the available synaptic delays are $1 - 16$ ms. The PSP is modeled by an α function with a membrane-potential decay time $\tau = 5$ms, but larger values up to at least 15 ms also worked (well in line with estimates of the effective decay time in cortical neurons). The network was composed of three input neurons (2 coding and 1 reference neuron), 4 hidden neurons (of which one inhibitory) and 1 output neuron. Only positive weights were allowed. In this configuration, the network reliably learned the XOR pattern within 500 cycles with $\eta = 0.001$.
After [3], we also tested the network on an interpolated XOR function. Using 3 input, 6 hidden and 1 output neurons, the function shown in figure 2A was presented to the network. As shown in figure 2B, the network proved capable of learning the presented input with accuracy of the order of the internal time-step of the algorithm (0.2 ms). We observe that for a time-coded network, 10 neurons sufficed for encoding the function, in contrast to the 200 spiking neurons used in [3] to emulate a fast rate-coding.
As noted in section 2, the approximation of the dependence of the firing time t_j on the post-synaptic input x_j is only valid for a small ε space around t_j. We emperically confirmed this as we found that larger learning rates were associated with longer learning times due to increased fluctuations in [illegible]or.

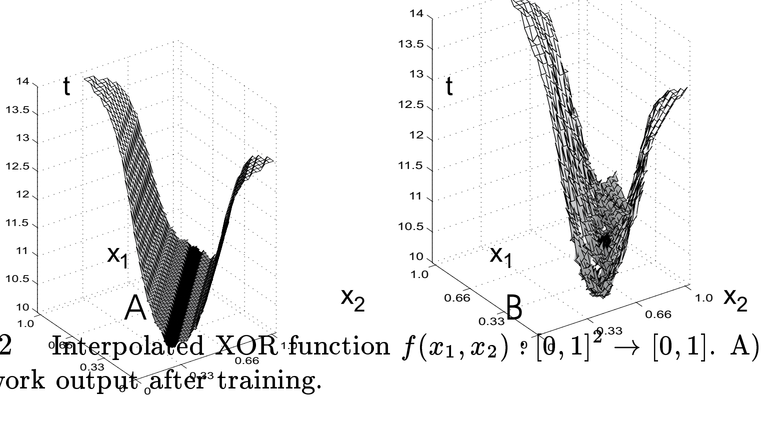

Figure 2 Interpolated XOR function $f(x_1, x_2) : [0,1]^2 \to [0,1]$. A) Target function. B) Network output after training.

Other data-sets. By encoding input variables over a number of input-neurons with graded overlapping receptive fields, we tested the algorithm on a number of benchmark problems: the Iris, the Wisconsin Breast Cancer and the Statlog Landsat dataset. For these problems, accuracies comparable to sigmoidal neural networks were obtained (data not shown, see [1]).

4 Discussion

In this paper, we derived a learning rule for feedforward spiking neural networks by back-propagating the temporal error at the output. By linearizing the relationship between the post-synaptic input and the resultant spiking time, we were able to circumvent the discontinuity associated with thresholding. The result is a learning rule that works well for smaller learning rates, which implies that we have shown in a direct way that networks of spiking neurons can carry out complex, non-linear tasks in a temporal code. As the experiments indicate, the SpikeProp algorithm is able to perform correct classification on non-linearly separable datasets with accuracy comparable to traditional sigmoidal networks, albeit with potential room for improvement.
Given the explicit use of the time-domain for calculations, we believe that the network is intrinsically more suited for learning and evaluating temporal patterns, as the network is virtually time-invariant in the absence of reference spikes. Applications of this t ypeare the subject of future research.
At this point there is no conclusive biological evidence that spike-based temporal coding is actively employed in the cortex. The results presented here suggest that in principle networks of spiking neurons with biologically plausible (e.g. relatively long) time-constants can perform complex non-linear classification in temporal code. In the case of fast processing of information, it has also been shown to be an order of magnitude more efficient in terms of neurons required for the extended XOR problem. Thus, there could be an advantage for the brain to use such temporal coding.

References

[1] S.M. Bohte, J. N. Kok, and H. La Poutr´e. Spike-prop: error-backprogation in multi-layer networks of spiking neurons. *CWI Te chnical Report*, in preparation, www.cwi.nl/~sbohte.

[2] W. Maass. Lower bounds for the computational power of networks of spiking neurons. *Neural Computation*, 8(1):1–40, 1996.

[3] W. Maass. Paradigms for computing with spiking neurons. In L. van Hemmen, editor, *Models of Neural Networks, vol 4*. Springer Berlin, 1999.

[4] T. Natschl¨ agerand B. Ruf. Spatial and temporal pattern analysis via spiking neurons. *Network: Computation in Neural Systems*, 9(3):319–332, 1998.

[5] N. Rashevsky. *Mathematical Biophysics*, volume II. New York: Dover, 1960.

[6] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.