Error-Backpropagation in Temporally Encoded Networks of Spiking Neurons

Sander M. Bohte[1] Han A. La Poutré[1] Joost N. Kok[1,2]
S.M.Bohte@cwi.nl Han.La.Poutre@cwi.nl joost@liacs.nl
[1] *CWI*
P.O.Box 94079, 1090 GB Amsterdam, The Netherlands
[2] *LIACS, Leiden University*
P.O. Box 9512, 2300 RA Leiden, The Netherlands

ABSTRACT
For a network of spiking neurons that encodes information in the timing of individual spike-times, we derive a supervised learning rule, *SpikeProp*, akin to traditional error-backpropagation and show how to overcome the discontinuities introduced by thresholding. With this algorithm, we demonstrate how networks of spiking neurons with biologically reasonable action potentials can perform complex non-linear classification in fast temporal coding just as well as rate-coded networks. We perform experiments for the classical XOR-problem, when posed in a temporal setting, as well as for a number of other benchmark datasets. Comparing the (implicit) number of spiking neurons required for the encoding of the interpolated XOR problem, it is demonstrated that temporal coding requires significantly less neurons than instantaneous rate-coding.

2000 Mathematics Subject Classification: 82C32, 68T05, 68T10, 68T30, 92B20.
1998 ACM Computing Classification System: C.1.3, F.1.1, I.2.6, I.5.1.
Keywords and Phrases: spiking neurons; temporal coding; error-backpropagation
Note: Work carried out under theme SEN4 "Evolutionary Systems and Applied Algorithmics". This paper has been submitted for publication, a short version has been presented at the European Symposium on Artificial Neural Networks 2000 (ESANN'2000) in Bruge, Belgium.

1. Introduction

Due to its success in artificial neural networks, the sigmoidal neuron is reputed to be a successful model of biological neuronal behavior. By modeling the rate at which a single biological neuron discharges action potentials (spikes) as a monotonically increasing function of input-match, many useful applications of artificial neural networks have been build [22; 7; 37; 34] and substantial theoretical insights in the behavior of connectionist structures have been obtained [40; 27].

However, the spiking nature of biological neurons has recently led to explorations of the computational power associated with temporal information coding in single spikes [31; 21; 13; 26; 20; 17; 49]. In [32] it was proven that networks of spiking neurons can simulate arbitrary feedforward sigmoidal neural nets and can thus approximate any continuous function. In fact, it has been shown theoretically that spiking neural networks that convey information by individual spike times are computationally more powerful than neurons with sigmoidal activation functions [29].

As spikes can be described by 'event' coordinates (place,time) and the number of active (spiking) neurons is typically sparse, artificial spiking neural networks have been shown to allow for very efficient implementations of large neural networks [48; 33]. Single-spike-time computing has also been suggested as a new paradigm for VLSI neural network implementations [28] and would offer a drastic speed-up.

Network architectures based on spiking neurons that encode information in the individual spike times have yielded, amongst others, a self-organizing map akin to Kohonen's SOM [39] and a network

for unsupervised clustering [35]. The principle of coding input intensity by relative firing time has also been successfully applied to a network for character recognition [10]. For applications of temporally encoded spiking neural networks however, no practical supervised algorithm has been developed so far. Even in [10] the authors resort to traditional sigmoidal back-propagation to learn to discriminate the histograms of different spike-time responses.

To enable useful supervised learning with the temporal coding paradigm, we develop a learning algorithm for single spikes which leverages the aforementioned advantages of spiking neurons while allowing for at least equally powerful learning as in sigmoidal neural networks. We derive an error-backpropagation based supervised learning algorithm for networks of spiking neurons that transfer the information in the timing of a single spike. The method we use is analogous to the derivation by Rumelhart *et al.* [41]. To overcome the discontinuous nature of spiking neurons, we approximate the thresholding function. We show that the algorithm is capable of learning complex non-linear tasks in spiking neural networks with similar accuracy as traditional sigmoidal neural networks. This is demonstrated experimentally for the classical XOR classification task, as well as for a number of real-world datasets.

We believe that our results are also of interest to the broader connectionist community, as the possibility of coding information in spike times has been receiving considerable attention. In particular, we demonstrate empirically that networks of biologically reasonable spiking neurons can perform complex non-linear classification in a fast temporal encoding just as well as rate-coded networks. Although this paper primarily describes a learning rule applicable to a class of artificial neural networks, spiking neurons are significantly closer to biological neurons then sigmoidal ones. Importantly, to compute with a rate-code on a very short time-scale, it is generally believed that in biological neural systems the responses of a large number of spiking neurons are pooled to obtain an instantaneous measure of average firing-rate. When we compare such an instantaneous rate-code to temporal coding with single spikes, we find that significantly less spiking neurons are required, as will be shown for the XOR-problem.

Our results also support the prediction that the length of the rising segment of the post-synaptic potential needs to be longer than the relevant temporal structure in order to allow for reliable temporal computation [28]. For spiking neurons, the post-synaptic potential describes the dynamics of a spike impinging onto a neuron, and is typically described as the difference of two exponentially decaying functions [19]. The effective rise and decay time of such a function is modeled after the membrane-potential time constants of biological neurons. As noted, from a computational point of view, our findings support the theoretical predictions in [28]. From a biological perspective, these findings dispel the notion among neuroscientists that fine temporal processing in spiking neural networks is prohibited by the relatively long time constants of cortical neurons (as noted for example in [15]).

This paper is organized as follows: in section 2 the spiking neural network is formally defined. In section 3, we derive the error-backpropagation equations. In section 4 we demonstrate the algorithm on the classical example of XOR, and also study the learning behavior of the algorithm. Encoding real-valued input-dimensions into a temporal code by means of receptive fields, we show results for a number of other benchmark problems in section 5. These experimental results are discussed in section 6. In a separate section, we consider the relevance of our findings for biological systems. In particular, we examine the currently available biological evidence that seems to point towards (at least) partial use of the temporal dimension of spikes in biological neural systems.

2. A Network of Spiking Neurons

The network architecture used consists of a feedforward network of spiking neurons with multiple delayed synaptic terminals (figure 1, as described by Natschläger & Ruf [35]). The neurons in the network generate action potentials, or spikes, when the internal neuron state variable, designated 'membrane potential', crosses a threshold ϑ. The relationship between input spikes and the internal state variable is described by the Spike Response Model (SRM), as introduced by Gerstner [18]. Depending on the choice of suitable spike-response functions, one can adapt this model to reflect the

dynamics of a large variety of different spiking neurons.

Formally, a neuron j, having a set Γ_j of immediate predecessors ('pre-synaptic neurons'), receives a set of spikes with firing times t_i, $i \in \Gamma_j$. Any neuron generates at most one spike during the simulation interval, and fires when the internal state variable reaches a threshold ϑ. The dynamics of the internal state variable $x_j(t)$ are determined by the impinging spikes, whose impact is described by the spike-response function $\varepsilon(t)$ weighted by the synaptic efficacy ("weight") w_{ij}:

$$x_j(t) = \sum_{i \in \Gamma_j} w_{ij} \varepsilon(t - t_i) \tag{2.1}$$

The spike-response function in (2.1) effectively models the unweighted post-synaptic potential (PSP) of a single spike impinging on a neuron. The height of the PSP is modulated by the synaptic weight w_{ij} to obtain the actual post-synaptic potential. The spike-response function as used in our experiments is defined in (2.3).

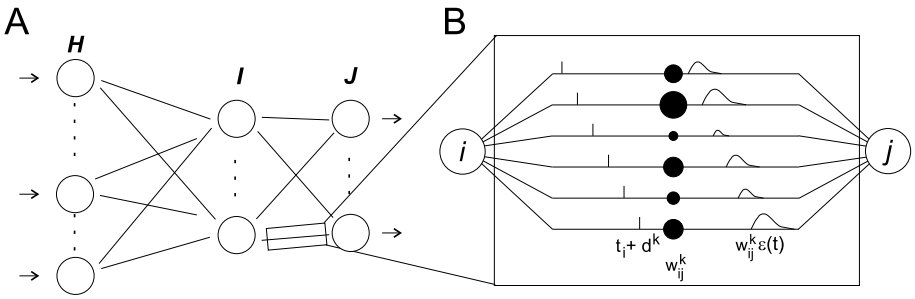

Figure 1: A) Feedforward spiking neural network B) Connection consisting of multiple delayed synaptic terminals.

In the network as introduced in [35], an individual connection consists of a fixed number of m synaptic terminals, where each terminal serves as a sub-connection that is associated with a different delay and weight (figure 1B). The delay d^k of a synaptic terminal k is defined by the difference between the firing time of the pre-synaptic neuron, and the time the post-synaptic potential starts rising (cf. figure 2A). For the sake of the calculations, we describe a pre-synaptic spike at a synaptic terminal k as a PSP of standard height with delay d^k. In effect, we thus obtain the unweighted contribution of a single synaptic terminal to the state variable:

$$y_i^k(t) = \varepsilon(t - t_i - d^k), \tag{2.2}$$

with $\varepsilon(t)$ a spike-response function shaping a PSP, with $\varepsilon(t) = 0$ for $t < 0$. The time t_i is the firing time of pre-synaptic neuron i, and d^k the delay associated with the synaptic terminal k. The spike-response function describing a standard PSP is of the form:

$$\varepsilon(t) = \frac{t}{\tau} e^{1 - \frac{t}{\tau}}, \tag{2.3}$$

modeling a simple α-function for $t > 0$ (else 0), and τ models the membrane potential decay time constant that determines the rise- and decay-time of the PSP.

Extending (2.1) to include multiple synapses per connection and inserting (2.2), the state variable x_j of neuron j receiving input from all neurons i can then be described as the weighted sum of the pre-synaptic contributions:

$$x_j(t) = \sum_{i \in \Gamma_j} \sum_{k=1}^{m} w_{ij}^k y_i^k(t), \tag{2.4}$$

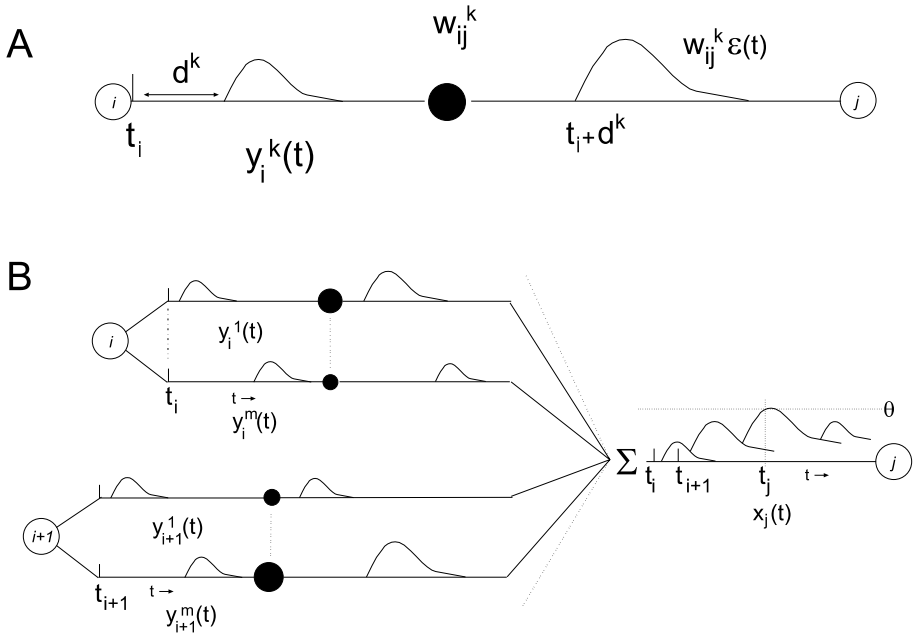

Figure 2: Connections in a feedforward spiking neural network: A) single synaptic terminal: the delayed pre-synaptic potential is weighted the synaptic efficacy to obtain the post-synaptic potential. B) Two multi-synapse connections. Weighted input is summed at the target neuron.

where w_{ij}^k denotes the weight associated with synaptic terminal k (cf. figure 2B). The firing time t_j of neuron j is determined as the first time when the state variable crosses the threshold ϑ: $x_j(t) \geq \vartheta$. Thus, the firing time t_j is a non-linear function of the state-variable x_j: $t_j = t_j(x_j)$. The threshold ϑ is constant and equal for all neurons in the network.

3. Error-backpropagation

We derive error-backpropagation, analogous to the derivation by Rumelhart *et al.* [41]. Equations are derived for a fully connected feedforward network with layers labeled H(input), I(hidden) and J(output), where the resulting algorithm applies equally well to networks with more hidden layers.

The target of the algorithm is to learn a set of target firing times, denoted $\{t_j^d\}$, at the output neurons $j \in J$ for a given set of input patterns $\{P[t_1..t_h]\}$, where $P[t_1..t_h]$ defines a single input pattern described by single spike-times for each neuron $h \in H$. We choose as 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 $\{t_j^a\}$, this error-function is defined by:

$$E = \frac{1}{2} \sum_{j \in J} (t_j^a - t_j^d)^2. \tag{3.1}$$

For error-backpropagation, we treat each synaptic terminal as a separate connection k with weight w_{ij}^k. Hence, for a backprop-rule, we need to calculate:

$$\Delta w_{ij}^k = -\eta \frac{\partial E}{\partial w_{ij}^k} \tag{3.2}$$

with η the learning rate and w_{ij}^k the weight of connection k from neuron i to neuron j. As t_j is a function of x_j, which depends on the weights w_{ij}^k, the derivative in the right hand part of (3.2) can be expanded to:

$$\frac{\partial E}{\partial w_{ij}^k} = \frac{\partial E}{\partial t_j}(t_j^a)\frac{\partial t_j}{\partial w_{ij}^k}(t_j^a) = \frac{\partial E}{\partial t_j}(t_j^a)\frac{\partial t_j}{\partial x_j(t)}(t_j^a)\frac{\partial x_j(t)}{\partial w_{ij}^k}(t_j^a). \tag{3.3}$$

In the last two right-handed factors, we express t_j as a function of the thresholded post-synaptic input $x_j(t)$ around $t = t_j^a$. We assume here that for a small enough region around $t = t_j^a$, the function x_j can be approximated by a linear function of t, as depicted in figure 3. For such a small region, we approximate the threshold function as $\delta t_j(x_j) = -\delta x_j(t_j)/\alpha$, with $\frac{\partial t_j}{\partial x_j(t)}$ the derivative of the inverse function of $x_j(t)$. The value α equals the local derivative of $x_j(t)$ with respect to t: $\alpha = \frac{\partial x_j(t)}{\partial t}(t_j^a)$.

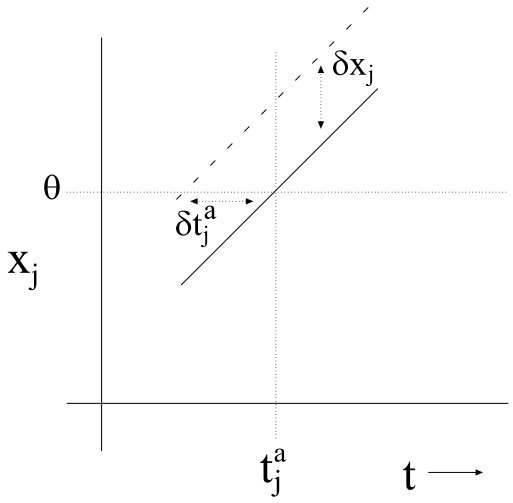

Figure 3: Relationship between δx_j and δt_j for an ϵ space around t_j.

The second right-hand factor in (3.3) then evaluates to:

$$\frac{\partial t_j}{\partial x_j(t)}(t_j^a) = \left.\frac{\partial t_j(x_j)}{\partial x_j(t)}\right|_{x_j=\vartheta} = \frac{-1}{\alpha} = \frac{-1}{\frac{\partial x_j(t)}{\partial t}(t_j^a)} = \frac{-1}{\sum_{i,l} w_{ij}^l \frac{\partial y_i^l(t)}{\partial t}(t_j^a)}. \tag{3.4}$$

In further calculations, we will write terms like $\frac{\partial x_j(t)}{\partial t}(t_j^a)$ as $\frac{\partial x_j(t_j^a)}{\partial t_j^a}$. We want to remark that this approximation implies that for large learning rates the algorithm might be less effective. We will consider this issue in the application of the algorithm in section 4.1.

The first factor in the right hand of (3.3), the derivative of E with respect to t_j, is simply:

$$\frac{\partial E(t_j^a)}{\partial t_j^a} = (t_j^a - t_j^d). \tag{3.5}$$

We also have:

$$\frac{\partial x_j(t_j^a)}{\partial w_{ij}^k} = \frac{\partial\{\sum_{n\in\Gamma_j}\sum_l w_{nj}^l y_n^l(t_j^a)\}}{\partial w_{ij}^k} = y_i^k(t_j^a). \tag{3.6}$$

When we combine these results, (3.2) evaluates to:

$$\Delta w_{ij}^k(t_j^a) = -\eta \frac{y_i^k(t_j^a) \cdot (t_j^d - t_j^a)}{\sum_{i \in \Gamma_j} \sum_l w_{ij}^l \frac{\partial y_i^l(t_j^a)}{\partial t_j^a}}. \tag{3.7}$$

For convenience, we define δ_j:

$$\delta_j \equiv \frac{\partial E}{\partial t_j^a} \frac{\partial t_j^a}{\partial x_j(t_j^a)} = \frac{(t_j^d - t_j^a)}{\sum_{i \in \Gamma_j} \sum_l w_{ij}^l \frac{\partial y_i^l(t_j^a)}{\partial t_j^a}}, \tag{3.8}$$

and (3.3) can now be expressed as:

$$\frac{\partial E}{\partial w_{ij}^k} = y_i^k(t_j^a) \frac{\partial E}{\partial t_j^a} \frac{\partial t_j^a}{\partial x_j(t_j^a)} = y_i^k(t_j^a) \delta_j, \tag{3.9}$$

yielding:

$$\Delta w_{ij}^k = -\eta y_i^k(t_j^a) \delta_j. \tag{3.10}$$

Equations (3.8) and (3.10) yield the basic weight adaptation function for neurons in the output layer.

We continue with the hidden layers: for error-backpropagation in other layers than the output layer, the generalized delta error in layer I is defined for $i \in I$ with actual firing times t_i^a:

$$\begin{aligned} \delta_i &\equiv \frac{\partial t_i^a}{\partial x_i(t_i^a)} \frac{\partial E}{\partial t_i^a} \\ &= \frac{\partial t_i^a}{\partial x_i(t_i^a)} \sum_{j \in \Gamma^i} \frac{\partial E}{\partial t_j^a} \frac{\partial t_j^a}{\partial x_j(t_j^a)} \frac{\partial x_j(t_j^a)}{\partial t_i^a} \\ &= \frac{\partial t_i^a}{\partial x_i(t_i^a)} \sum_{j \in \Gamma^i} \delta_j \frac{\partial x_j(t_j^a)}{\partial t_i^a}, \end{aligned} \tag{3.11}$$

where Γ^i denotes the set of immediate neural successors in layer J connected to neuron i.

As in [7], in (3.11) we expand the local error $\frac{\partial E(t_i^a)}{\partial t_i^a}$ in term of the weighted error contributed to the subsequent layer J. For the expansion, the same chain rule as in (3.3) is used under the same restrictions, albeit for $t = t_i$.

The term $\frac{\partial t_i^a}{\partial x_i(t_i^a)}$ has been calculated in (3.4), while for $\frac{\partial x_j(t_j^a)}{\partial t_i^a}$:

$$\begin{aligned} \frac{\partial x_j(t_j^a)}{\partial t_i^a} &= \frac{\partial \sum_{l \in I} \sum_k w_{lj}^k y_l^k(t_j^a)}{\partial t_i^a} \\ &= \sum_k w_{ij}^k \frac{\partial y_i^k(t_j^a)}{\partial t_i^a}. \end{aligned} \tag{3.12}$$

Hence,

$$\delta_i = \frac{\sum_{j \in \Gamma^i} \delta_j \{\sum_k w_{ij}^k \frac{\partial y_i^k(t_j^a)}{\partial t_i^a}\}}{\sum h \in \Gamma_i \sum_l w_{hi}^l \frac{\partial y_h^l(t_i^a)}{\partial t_i^a}}. \tag{3.13}$$

Thus, for a hidden layer and by (3.10),(3.11),(3.12) and (3.13), the weight adaptation rule compounds to:

$$\Delta w_{hi}^k = -\eta y_h^k(t_i^a)\delta_i = -\eta \frac{y_h^k(t_i^a)\sum_j\{\delta_j \sum_k w_{ij}^k \frac{\partial y_i^k(t_j^a)}{\partial t_i^a}\}}{\sum_{n\in\Gamma_i}\sum_l w_{ni}^l \frac{\partial y_n^l(t_i^a)}{\partial t_i^a}}. \tag{3.14}$$

Analogous to traditional error-backpropagation algorithms, the weight adaptation rule (3.14) above generalizes to a network with multiple hidden layers I numbered $J-1\ldots 2$ by calculating the delta-error at layer i from the delta-error in layer $i+1$, in effect back-propagating the error.

The algorithm derived above, termed *SpikeProp*, is summarized in the following table:

SpikeProp Algorithm
Calculate δ_j for all outputs according to (3.8)
For each subsequent layer $I = J-1\ldots 2$ Calculate δ_i for all neurons in I according to (3.13)
For output layer J, adapt w_{ij}^k by $\Delta w_{ij}^k = -\eta y_i^k(t_j)\delta_j$ (3.10) For each subsequent layer $I = J-1\ldots 2$ Adapt w_{hi}^k by $\Delta w_{ij}^k = -\eta y_h^k(t_i)\delta_i$ (3.14)

4. The XOR-problem

In this section, we consider learning the exclusive-or (XOR) function as an elementary test of the algorithm derived in section 3. The XOR function is a classical example of a non-linear problem that requires hidden units to transform the input into the desired output.

To encode the XOR-function in spike-time patterns, we substitute a 0 with a "late" firing time and a 1 with an "early" firing time. With specific values 0 and 6 for the respective input times, we use the following temporally encoded XOR:

Input Patterns		Output Patterns
0 0	→	16
0 6	→	10
6 0	→	10
6 6	→	16

The numbers in the table represent spike times, which we will denote in milliseconds for clarity. We use a third (bias) input neuron in our network that always fired at $t = 0$ to designate the reference start time (otherwise the problem becomes trivial). We define the difference between the times equivalent with "0" and "1" as the coding interval ΔT, which in this example corresponds to 6ms.

For the network we used the feed-forward network described in section 2. The connections have a delay interval of 15 ms; hence the available synaptic delays are $1-16$ ms. The PSP is defined by an α function as in (2.3) with a decay time $\tau = 7$ ms. We want to remark that larger values up to at least 15 ms resulted in similar learning (see also section 4.1).

The network was composed of three input neurons (2 coding and 1 reference neuron), 5 hidden neurons (of which one inhibitory neuron generating only negative sign PSP's) and 1 output neuron. Only positive weights were allowed. The results with this configuration are as follows: the network reliably learned the XOR pattern within 250 cycles with $\eta = 0.01$, see also subsection 4.1. Note that in order to "learn" XOR, some 16 x 3 x 5 + 16 x 5 x 1 = 320 individual weights had to be adjusted.

While testing the algorithm, we found that it was necessary to explicitly incorporate inhibitory and excitatory neurons, with inhibitory and excitatory neurons defined by generating respectively negative and positive PSP's while maintaining positive weights. In fact, the *Spikeprop* algorithm would not converge if the connections were allowed to contain a mix of both positive and negative weights. We suspect that the cause of this problem lies in the fact that in the case of mixed weights, the effect

of a single connection onto the target neuron is no longer a monotonically increasing function (as it is for sufficiently large time-constants, see also the discussion in section 4.1). We remark that the introduction of inhibitory and excitatory neurons is not a limitation on the *Spikeprop* algorithm: by expanding the network in such a way that each excitatory neuron has an inhibitory counterpart, in effect a mixed sign connection is implemented. In the experiments though, the inclusion of one or two inhibitory neurons was sufficient to enable learning.

We also tested the network on an interpolated XOR function mapping $f(x_1, x_2) : [0, 1]^2 \rightarrow [0, 1]$, like in [30]. We encode this function temporally as $f(t_1, t_2) : [0, 6]^2 \rightarrow t_3 : [10, 16]$, with times t_1, t_2 and t_3 in milliseconds (figure 4A). Using 3 input, 5 hidden and 1 output neurons, 961 data-points from this function were presented to the network. The result of learning with *Spikeprop* by the network is shown in figure 4B. The network proved to be capable of learning the presented input with accuracy of the order of the internal integration time-step of the *Spikeprop* algorithm: 0.1 ms (for the target sum squared error of 50.0 the average error per instance was 0.2ms).

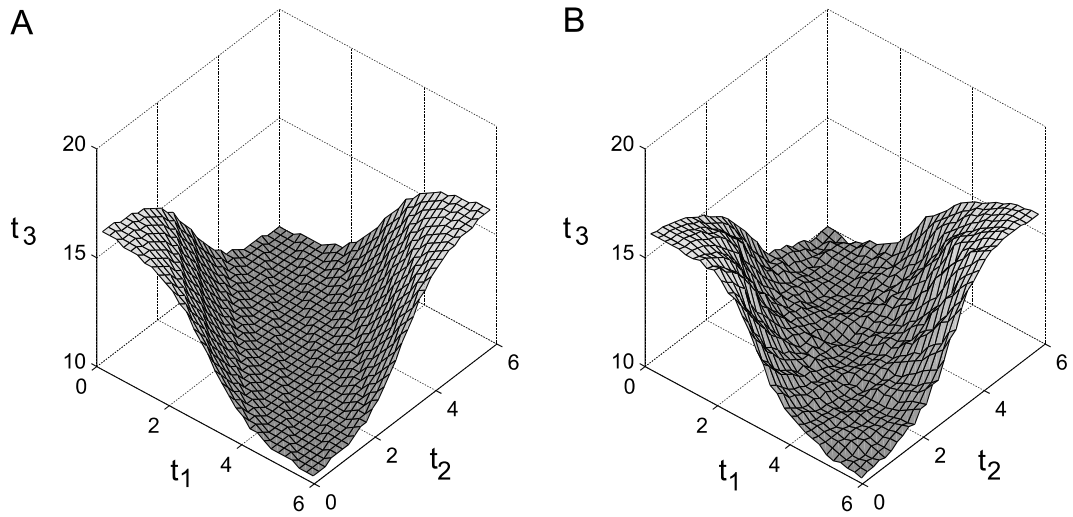

Figure 4: Interpolated XOR function $f(t_1, t_2) : [0, 6]^2 \rightarrow [10, 16]$. A) Target function. B) Network output after training. The network reached the sum squared error-criterion of 50.0 after learning 12996 randomly drawn examples from the 961 data-points.

In the interpolated XOR experiment with a spiking neural network, 9 spiking neurons are sufficient for encoding the function. This in contrast to the observation in [30] where it was demonstrated that in order to encode an interpolated XOR-function by *rate-code* within 10ms, a pool of some 200 spiking neurons was required. A rate-code can be obtained within 10ms from sparsely spiking neurons (say 40Hz) by counting the number of spikes within a pool of neurons during the 10ms time-window (also referred to as the instantaneous firing-rate).

4.1 Error gradient and learning rate

In this section, we consider the influence of the learning rate η and the time-constant τ on the learning capabilities of the *Spikeprop* algorithm in a network of spiking neurons.

As noted in section 3, the approximation of the dependence of the firing time t_j^a on the post-synaptic input x_j is only valid for a small region around t_j^a. We found indeed that for larger learning rates the probability of convergence decreased, although for learning rates up to 0.01, larger learning rates were associated with faster learning times. This can be seen in figure 5, where the average number of learning iterations required for the XOR function are plotted for a number of time-constants τ.

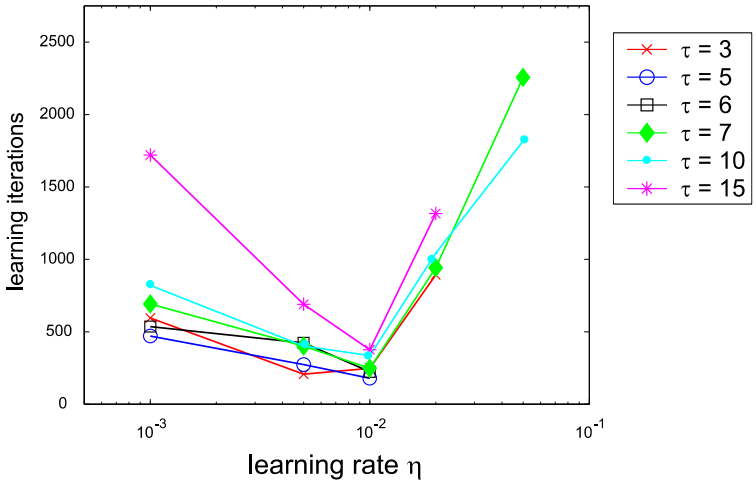

Figure 5: Learning XOR: Average number of required learning iterations to reach the sum squared error target (SSE) of 1.0. The average was calculated for those runs that converged.

In figure 6, the reliability of learning for different values of the time-constant τ is plotted. The plot shows that for optimal convergence, the most reliable results are obtained for values of the time-constant that are (somewhat) larger than the time interval ΔT in which the XOR-problem is encoded (here: $\Delta T = 6$).

The convergence graphs for different values of the coding interval ΔT are plotted in figure 7A-B and show the same pattern. These results confirm the results as obtained by Maass in [28], where it was shown theoretically that the time-constant τ needs to be larger than the relevant coding interval ΔT. This observation with regard to parameter choice was also made for the benchmark results presented in section 5: a substantial speedup and somewhat better results were obtained if the time constant τ was slightly larger than the coding interval.

5. Other Benchmark Problems

In this section, we perform experiments with the *SpikeProp* algorithm on a number of standard benchmark problems, notably the Iris-dataset, the Wisconsin breast-cancer dataset and the Statlog Landsat dataset.

First however, we introduce a formal method for encoding input-variables into temporal spike-time patterns by population coding. Although our simple and elegant, we are not aware of any previous encoding methods for transforming real-world data into spike-time patterns and describe the method in detail.

Encoding continuous input-variables in spike-times

When using larger datasets, a critical factor becomes the encoding of the input: as the coding interval ΔT is restricted to a fixed interval of the order of τ, the full range of the input can be encoded by either using increasingly small temporal differences or alternatively by distributing the input over multiple neurons. In our network simulation, the network state is iterated with a fixed time-step. Hence an increased temporal resolution for some input-values imposes a computational penalty on the entire network. Keeping to the biologically inspired approach, we also remark that cortical spike-times are believed to be inherently noisy as they exhibit a 'jitter' in their spike times of the order of 1-2ms [31]. Hence an encoding that requires single spikes to have a higher temporal precision is implausible.

These arguments favor population coding, and we employed an encoding based on arrays of *receptive fields*. This enables the representation of continuously valued input-variables by a population of

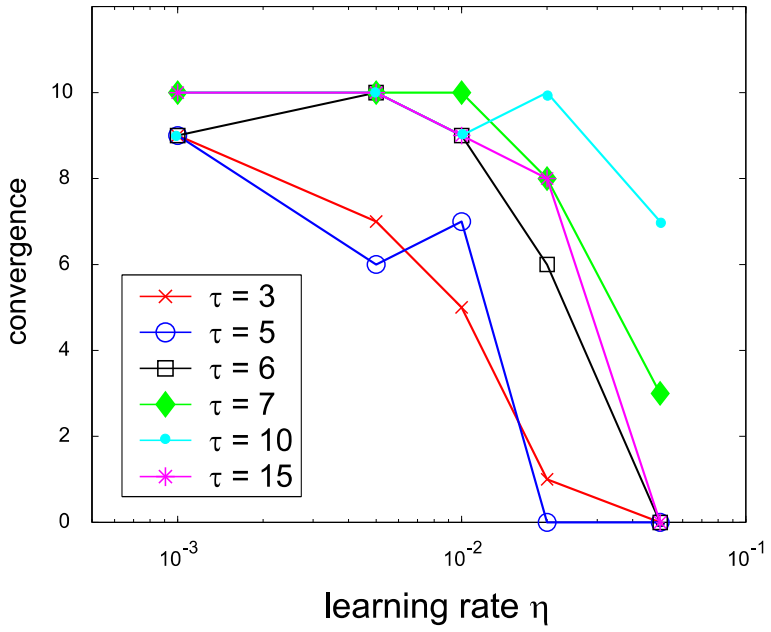

Figure 6: Learning XOR: Number of runs out of 10 that converged.

neurons with graded and overlapping sensitivity profiles, such as gaussian activation functions (the receptive fields, RF's). This encoding constitutes a biologically plausible and well studied method for representing real valued parameters [17; 4; 51; 52; 45; 44]. To encode values into a temporal pattern, it is sufficient to associate highly stimulated neurons with early firing times and less stimulated neurons with later (or non) firing times, the extension to temporal coding is straightforward

We independently encode the respective input variables: each input-dimension is encoded by an array of 1-dimensional receptive fields. Improved representation accuracy for a particular variable can then be obtained by sharpening the receptive fields and increasing the number of neurons [52]. Such an encoding has been shown to be statistically bias-free [4] and in the context of spike-time patterns it has been applied successfully to unsupervised learning problems [8].

In our experiments, we determined the input ranges of the data, and encoded each input variable with neurons covering the whole data-range. For a range $[I^n_{min} \ldots I^n_{max}]$ of a variable n, m neurons were used with gaussian receptive fields. For a neuron i its center was set to $I^n_{min} + i \cdot \frac{\{I^n_{max} - I^n_{min}\}}{m-2}$ and width $\sigma = \beta \frac{\{I^n_{max} - I^n_{min}\}}{m-2}$. For β, values between 1.0 and 2.0 were tried, and for the reported experiments a value of 1.5 was used as it produced the best results. For each input pattern, the response values of the neurons encoding the respective variables were calculated, yielding n x m values between 0 and 1. These values were then converted to delay times, associating $t = 0$ with a 1, and increasingly later times up to $t = 10$ with lower responses. The resulting spike-time were rounded to the nearest internal time-step, and neurons with responses larger than $t = 9$ were coded to not fire, as they are considered to be insufficiently excited. The encoding of a single input-value a by receptive field population coding is depicted in figure 8.

Output classification was encoded according to a winner-take-all paradigm where the neuron coding for the respective class was designated an early firing time, and all others a considerably later one, thus setting the effective coding interval ΔT. A classification was deemed correct if the neuron that fired earliest corresponded to the neuron required to fire first. To obtain a winner in the case where multiple neurons fired at the same time step, a first-order approximation to the real-value firing time

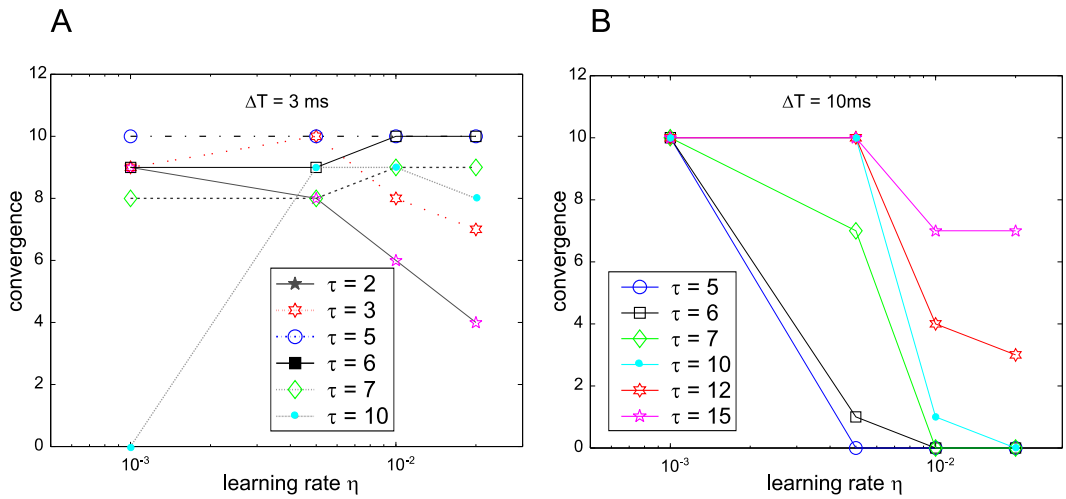

Figure 7: Number of runs out of 10 that converged for different values of τ. A) For a coding interval $t = 0 \ldots 3$, $[0, 3]^2 \rightarrow [7, 10]$. Target was an SSE of 0.7. B) For a coding interval $t = 0 \ldots 10$, $[0, 10]^2 \rightarrow [15, 25]$. Target was an SSE of 3.0.

was performed based on the current and previous membrane-potentials.

The temporal encoding of input-variables thus obtained has two important properties: by assigning firing times to only a small set of significantly activated neurons we achieve a sparse coding, allowing for an efficient event-based network simulation as in [14]. Also, by encoding each variable independently, we achieve a coarse coding where each variable can be encoded by an optimal number of neurons.

We tested our framework on several benchmark problems. The temporal encoding introduced above is used for the conversion of the datasets to translate them into temporal patterns of discrete spikes.

Iris dataset

The Iris data-set is considered to be a reasonably simple classification problem. It contains three classes of which two are not linearly separable. As such, it provides a basic test of applicability of our framework. The dataset contains 150 cases, where each case has 4 input-variables. Each input variable was encoded by 12 neurons with gaussian receptive fields. The data was divided in two sets and classified using two-fold cross-validation. The results are presented in table 5. We also obtained results on the same dataset from a sigmoidal neural network as implemented in Matlab v5.3. As the input is preprocessed in the same way, both error back-propagation methods can be compared directly. The results are highly comparable, as they are also for the other benchmark datasets.

Wisconsin Breast Cancer data-set

The breast cancer diagnosis problem is described in [50]. The data is from the University of Wisconsin Hospitals and contains 699 case entries, divided over benign and malignant cases. Each case has nine measurements, and each measurement is assigned an integer between 1 and 10, with larger numbers indicating a greater likelihood of malignancy. A small number of cases (16) contain missing data. In these cases, the neurons coding for the missing variable did not fire at all. For our experiment, we encoded each measurement with 7 equally spaced neurons covering the input range. The dataset was divided in two equal parts, and we used two-fold cross-validation. The results are summarized in table 5. The results as we obtained them are comparable with values reported in literature for BP learning on this dataset: for instance, in a benchmark study by Roy, Govil and Miranda [38] on an earlier version of this dataset containing 608 cases an error-rate of 2.96% was obtained on the test-set, using a multi-layer-perceptron with standard error-backpropagation learning.

Landsat data-set

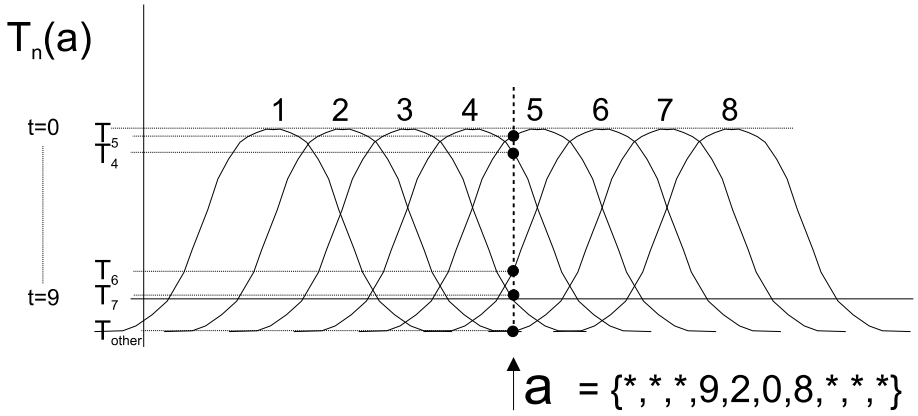

Figure 8: Input value a encoded by 8 broadly tuned activation functions. The resulting encoding is marked by the black dots for a number of input neurons (T_i).

To test the algorithm's performance on a larger data-set, we investigated the Landsat dataset as described in the StatLog survey of machine learning algorithms [34]. This dataset consists of a training set of 4435 cases and a test set of 2000 cases and contains 6 ground cover-types (classes). For classification of a single pixel, each case contains the values of a 3x3-pixel patch, with each pixel described by 4 spectral bands, totaling 36 inputs per case. For the classification of the central pixel, we averaged the respective spectral bands in the 3x3 patch and then encoded each separate band with 25 neurons. Results are shown in table 5. The results obtained by the Statlog survey are summarized in table 2.

Again, the results with *SpikeProp* are similar to the Matlab BP results, though even for twice the number of iterations, the Matlab BP results are somewhat worse. Compared to the best machine learning methods as reported in [34], the *SpikeProp* results are nearly identical to those reported for traditional BP, even though we reduced the input-space 9-fold by taking as input the average of 3x3 pixel environment. Note though that in [34] the Multi-Layer-Perceptron BP results lag other ML methods for this particular classification problem. After extended learning however, the traditional BP method as implemented in Matlab produces results that are significantly better than the reported values in [34] for MLP-BP learning. Such extended learning was not feasible for the SpikeProp algorithm due to time-constraints, although the results showed increasing classification accuracy with extended learning.

From the results shown, we conclude that the application of the *SpikeProp* algorithm on temporally encoded versions of benchmark problems yielded very similar results as compared to those obtained from traditional BP on sigmoidal neural networks. Typically, less learning epochs were required with *SpikeProp*, however, in single epoch 16 times more weights were modified offsetting this advantage. The results reported for the default Matlab BP routine were comparable or better as compared to those reported in literature, validating the generality of this conclusion.

6. Discussion

Computational Implications

The results obtained for the XOR-problem show that our *SpikeProp* algorithm works reliably when used with smaller learning rates and time constants that are larger than the coding interval. The results hence demonstrate that in order to reliably learn patterns of temporal width ΔT, the time-constant used to define the PSP is critical: as predicted on theoretical grounds in [28], this time-constant has to be larger than the pattern that is being learnt.

Iris dataset						
algorithm	inputs	hidden	outputs	iterations	training-set	error testset
SpikeProp	50	10	3	1,000	97.4% ± 0.1	96.1% ± 0.1
Matlab BP	50	10	3	3,750	99.0% ± 0.1	95.7% ± 0.1
Wisconsin Breast Cancer dataset						
algorithm	inputs	hidden	outputs	iterations	training-set	error testset
SpikeProp	64	15	2	1500	97.6% ± 0.2	97.0% ± 0.6
Matlab BP	64	15	2	3,500	97.7% ± 0.3	96.7% ± 0.6
Statlog Landsat set						
algorithm	inputs	hidden	outputs	iterations	training-set	test-set
SpikeProp	101	25	6	60,000	87.0% ± 0.5	85.3% ± 0.3
Matlab BP	101	25	6	110,078	83.6% ± 1.3	82.0% ± 1.5
Matlab BP	101	25	6	1,108,750	91.2% ± 0.5	88.0% ± 0.1

Table 1: The Iris data set was split into two sets of 75 items each for cross-validation. The Matlab BP routine was run for 50 epochs, or 1500 iterations. The Wisconsin Breast Cancer dataset was split in two for cross-validation. The Matlab BP routine was run for 10 epochs, or 3500 iterations. For the Statlog Landsat, the Matlab BP routine was run for 25 epochs, or 100,00+ iterations as well as for 250 epochs, or 1,000,000+ iterations. All results are averaged from 10 runs on each cross-validation set, with the exception of the Landsat database where no cross-validation was used (as in the original Statlog experiment). The experiments were run with a coding interval of 4ms and a time constant τ of 7ms in order to reduce learning time. The learning rate η was set to 0.0075. Error-backpropagation in a 3-layer MLP was simulated with the Matlab v5.3 "TRAINLM" routine.

Statlog data		
method	error training-set	error test-set
BackProp	88.8%	86.1%
RBF	88.9%	87.9%
k-NN	91.11%	90.06%

Table 2: Benchmark results on Landsat database from literature. The k-Nearest Neighbor algorithm is mentioned as it provided the best classification of all methods examined in the Statlog survey.

For machine learning purposes, the event nature of spikes allows for very efficient implementations of spiking neural networks (e.g. [14; 33]), especially in the case of sparse activity as achieved by the channel encoding (section 5). Our current implementation of the algorithm is computationally somewhat intensive as we simulate the temporal evolution of the system with a large number of (small) time-steps. Switching to an event based system however should enhance system performance.

The number of learning iterations used in our simulation was comparable to those required in BP learning in Multi-Layer-Perceptrons (MLP) of the same pre-processed datasets. It has to be noted that we also did not exhaustively consider this issue in terms of parameter tuning. The time for a single learning iteration was clearly longer, as in the simulations each connection consisted of 16 delayed terminals each with their own weight. Hence the number of weights that needed to be tuned in any of the experiment was quite large. Qualitatively we remark that convergence in our network seemed to be more reliable as compared to BP: in experiments with spiking neural networks on the real-world datasets, we experienced no convergence failures of the *SpikeProp* algorithm, whereas the comparative BP-MLP experiments occasionally did (<10%). Research along the lines of optimal parameters would be a logical step for future investigations.

Given the explicit use of the time domain for calculations, we believe that a network of spiking neurons is intrinsically more suited for learning and evaluating temporal patterns than sigmoidal networks, as the spiking neural network is virtually time-invariant in the absence of reference spikes. Applications of this type will be the subject of future research; a first application of the *SpikeProp* algorithm using this feature has been implemented in a handwritten-character recognition task. In this task, the spiking neural network was able to get better recognition rates for a subset of characters as compared to simple statistical methods [36].

Biological Implications

Within the broader connectionist community, the real-valued output of a neuron is generally assumed to be its average firing-rate. In spite of the success of this paradigm in neural network modeling and the substantial electrophysiological support, there has been an increasing number of reports where the actual timing of action potentials (spikes) carries significant information (e.g., in the bat [25], the barn owl [12] and the electric fish [23]). Indeed, a respectable amount of neurophysiological evidence suggests that such neuronal activation with millisecond precision could be linked with dynamic neurocognitive information processing [1; 3; 24; 11; 16].

Besides the reports of remarkably precise firing times of individual neurons, the actual number of spikes available for neuronal processing has been calculated to be rather small in some cases: psychophysical experiments by Thorpe, Fize and Marlot [47] suggest that complex neuronal information processing can be achieved in under 10ms per synaptic stage, or less than one spike per neuron.

Given these considerations, attention has been given to the possibility of (additional) information coding in the timing of individual action potentials [6; 46; 1]. This could more easily achieve fast information transfer [48; 32] and would be cheaper in terms of neuron count. Biological evidence is emerging that suggests that delay-based temporal coding is actively employed in the cortex, e.g. [2; 26; 5; 9].

Against the coding of information with a temporal code on very short time-scales, it has been argued that this is unlikely due to the relatively long time constants of cortical neurons. The theoretical work by Maass [28] already refuted this notion (as also argued in [15]), but the results presented here clearly demonstrate that networks of spiking neurons with biologically plausible (relatively long) time constants are capable of performing complex non-linear classification for spike times encompassing a temporal coding intervals up to this time-constant. Indeed, the XOR results suggest that short time constants impede the integration of information over time-periods longer than this value.

In the case of fast processing of information, we have demonstrated that delay-based temporal coding can be an order of magnitude more efficient than rate coding in terms of spiking neurons required, as demonstrated in the interpolated XOR example. Due to the all-or-none nature of the action potentials generated by individual neurons, a rate-code on a time scale of 10ms is only available

to downstream neurons by deriving the instantaneous firing rate from a population of neurons activated by the same stimulus. This is problematic to achieve, as it has been noted that a reliable estimation of the instantaneous firing rate on a time-scale of 10 ms requires on the order of 100 pooled neurons [30; 42], whereas it has also been reported that pooling the activity of more than 100 neurons does not increase accuracy due to correlated input noise [43]. Especially in cases where quick responses are vital and neurons come at a premium, the neural economy of temporal coding would constitute an important advantage.

7. Conclusion

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 and for time-constants of the post-synaptic potential larger than the maximal temporal coding range. This latter result is in agreement with the theoretical predictions.

The algorithm also demonstrates 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.

References

1. M. Abeles, H. Bergman, E. Margalit, and E. Vaadia. Spatiotemporal firing patterns in the frontal cortex of behaving monkeys. *J. Neurophysiol.*, 70:1629–1658, 1993.
2. E. Ahissar, R. Sosnik, and S. Haidarliu. Transformation from temporal to rate coding in a somatosensory thalamocortical pathway. *Nature*, 406:302–306, July 2000.
3. J-M. Alonso, W.M. Usrey, and R.C. Reid. Precisely correlated firing in cells of the lateral geniculate nucleus. *Nature*, 383:815–819, 1996.
4. P. Baldi and W. Heiligenberg. How sensory maps could enhance resolution through ordered arrangements of broadly tuned receivers. *Biol. Cybern.*, 59:313–318, 1988.
5. Buo-qiang Bi and Mu-ming Poo. Distributed synaptic modification in neural networks induced by patterned stimulation. *Nature*, 401:792–796, 1999.
6. W. Bialek, F. Rieke, R.R. de Ruyter van Steveninck, and D. Warland. Reading a neural code. *Science*, 252:1854–1857, 1991.
7. Ch. M. Bishop. *Neural Networks for Pattern Recognition*. Clarendon Press, Oxford, 1995.
8. S.M. Bohte, J.N. Kok, and H. La Poutré. Unsupervised classification in a layered network of spiking neurons. In *Proceedings of IJCNN'2000*, volume IV, pages 279–285, 2000.
9. M. Brecht, W. Singer, and A.K. Engel. Role of temporal codes for sensorimotor integration in the superior collicus. In *Proceedings of ENA 98*, 1998.
10. Dean V. Buonomano and Michael Merzenich. A neural network model of temporal code generation and position-invariant pattern recognition. *Neural Computation*, 11(1):103–116, 1999.
11. G.T. Buracas, A. Zador, M.R. DeWeese, and T.D. Albright. Efficient discrimination of temporal patterns by motion-sensitive neurons in primate visual cortex. *Neuron*, 20:959–969, 1998.
12. C.E. Carr and M. Konishi. A circuit for detection of interaural time differences in the brain stem of the barn owl. *J. Neurosci.*, 10:3227–3246, 1990.
13. G. Deco and B. Schurmann. The coding of information by spiking neurons: an analytical study. *Network: comput. neural syst.*, 9:303–317, 1998.
14. A. Delorme, J. Gautrais, R. VanRullen, and S.J. Thorpe. Spikenet: A simulator for modeling large networks of integrate and fire neurons. *Neurocomputing*, pages 989–996, 1999.
15. M. Diesmann, M.-O. Gewaltig, and A. Aertsen. Stable propagation of synchronous spiking in

cortical neural networks. *Nature*, 402:529–33, 1999.

16. A.K. Engel, P. König, A.K. Kreiter, T.B. Schillen, and W. Singer. Temporal coding in the visual cortex: new vistas on integration in the nervous system. *Trends Neurosci.*, 15:218–226, 1992.
17. C. W. Eurich and S. D. Wilke. Multi-dimensional encoding strategy of spiking neurons. *Neural Computation*, in press, 2000.
18. W. Gerstner. Time structure of the activity in neural network models. *Phys Rev E*, 51:738–758, 1995.
19. W. Gerstner. Spiking neurons. In W. Maass and C. Bishop, editors, *Pulsed Neural Networks*. MIT Press, Cambridge, 1999.
20. W Gerstner, R Kempter, JL van Hemmen, and H Wagner. A neuronal learning rule for sub-millisecond temporal coding. *Nature*, 383:76–78, 1996.
21. W. Gerstner, R. Ritz, and L. van Hemmen. Why spikes? hebbian learning and retrieval of time-resolved excitation patterns. *Biological Cybernetics*, 69:503–515, 1993.
22. S. Haykin. *Neural Networks, A Comprehensive Foundations*. Maxwell Macmillan, 1994.
23. W. Heiligenberg. *Neural Nets in Electric Fish*. MIT Press, 1991.
24. J. Heller, J.A. Hertz, T.W. Kjaer, and B.J. Richmond. Information flow and temporal coding in primate pattern vision. *J. Comp. Neurosci.*, in press:1–17, 2000.
25. N. Kuwabara and N. Suga. Delay lines and amplitude selectivity are created in subthalamic auditory nuclei: the branchium of the inferior colliculus of the mustached bat. *Journal of Neurophysiology*, 69:1713–1724, 1993.
26. Gilles Laurent. A systems perspective on early olfactory coding. *Science*, 286:723–728, 1999.
27. D.S. Levine. *Introduction to Neural & Cognitive Modeling*. Lawrence Erlbaum Associates London, 1991.
28. W. Maass. Lower bounds for the computational power of networks of spiking neurons. *Neural Computation*, 8(1):1–40, 1996.
29. W. Maass. Noisy spiking neurons with temporal coding have more computational power than sigmoidal neurons. In Michael C. Mozer, Michael I. Jordan, and Thomas Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9, page 211. The MIT Press, 1997.
30. W. Maass. Paradigms for computing with spiking neurons. In L. van Hemmen, editor, *Models of Neural Networks*, volume 4. Springer Berlin, 1999.
31. W. Maass and C. M. Bishop. *Pulsed Neural Networks*, volume 1. MIT Press, Cambridge, MA, 1999.
32. Wolfgang Maass. Fast sigmoidal networks via spiking neurons. *Neural Computation*, 9(2):279–304, 1997.
33. Maurizio Mattia and Paolo Del Giudice. Efficient event-driven simulation of large networks of spiking neurons and dynamical synapses. *Neural Computation*, 12(10):–, 2000.
34. D. Michie, D.J. Spiegelhalter, and C.C. Taylor, editors. *Machine learning, neural and statistical classification*. Ellis Horwood, West Sussex, 1994.
35. T. Natschläger and B. Ruf. Spatial and temporal pattern analysis via spiking neurons. *Network: Computation in Neural Systems*, 9(3):319–332, 1998.
36. G. Pieri. Improvement in handwritten character recogntion: Mixtures of experts and temporal pattern recogntion in spiking neural networks. Msc, Free University of Amsterdam, 2000.
37. B.D Ripley. *Pattern Recognition and Neural Networks*. Clarendon Press, Oxford, 1996.
38. A. Roy, S. Govil, and R. Miranda. An algorithm to generate radial basis function(rbf)-like nets

for classification problems. *Neural Networks*, 8(2):179–201, 1995.

39. B. Ruf and M. Schmitt. Self-organization of spiking neurons using action potential timing. *IEEE-Neural Network*, 9(3):575–578, May 1998.
40. David E. Rumelhart, J. L. McClelland, and the PDP Research Group. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1. MIT Press, Cambridge, MA, 1986.
41. D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.
42. M.N. Shadlen and W.T. Newsome. Noise, neural codes and cortical organization. *Curr. Opin. Neurobiol.*, 4:569–579, 1994.
43. M.N. Shadlen and W.T. Newsome. The variable discharge of cortical neurons: implications for connectivity, computation and information coding. *J. Neuronsci*, 18:3870–3896, 1998.
44. Herman P. Snippe. Parameter extraction from population codes: A critical assessment. *Neural Computation*, 8(3):511–529, 1996.
45. H.P. Snippe and J.J. Koenderink. Information in channel-coded systems: correlated receivers. *Biological Cybernetics*, 67(2):183–190, 1992.
46. W.R. Softky. Simple codes versus efficient codes. *Curr. Opin. Neurobiol.*, 5:239–247, 1995.
47. S. Thorpe, F. Fize, and C. Marlot. Speed of processing in the human visual system. *Nature*, 381:520–522, 1996.
48. S.J. Thorpe and J. Gautrais. Rapid visual processing using spike asynchrony. In Michael C. Mozer, Michael I. Jordan, and Thomas Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9, page 901. The MIT Press, 1997.
49. Ch. von der Malsburg and W. Schneider. A neural cocktail-party processor. *Biological Cybernetics*, 54:29–40, 1986.
50. W.H. Wolberg. Cancer dataset obtained from williams h. wolberg from the university of wisconsin hospitals, madison, 1991.
51. K.-C. Zhang, I. Ginzburg, B.L. McNaughton, and T.J. Sejnowski. Interpreting neuronal population activity by reconstruction: Unified framework with application to hippocampal place cells. *Journal of Neurophysiology*, 79:1017–1044, 1998.
52. Kechen Zhang and Terrence J. Sejnowski. Neuronal tuning: To sharpen or broaden? *Neural Computation*, 11(1):75–84, 1999.