SPIKING NEURAL NETWORKS

SPIKING NEURAL NETWORKS

PROEFSCHRIFT

ter verkrijging van
de graad van Doctor aan de Universiteit Leiden,
op gezag van de Rector Magnificus Dr. D.D. Breimer,
hoogleraar in de faculteit der Wiskunde en
Natuurwetenschappen en die der Geneeskunde,
volgens besluit van het College voor Promoties
te verdedigen op woensdag 5 Maart 2003
te klokke 14.15 uur

door

Sander Marcel Bohte
geboren te Hoorn (NH)
in 1974

Promotiecommisie

Promotores:	Prof. Dr. J.N. Kok *Universiteit Leiden*
	Prof. Dr. Ir. J.A. La Poutré *CWI / Technische Universiteit Eindhoven*
Referent:	Dr. S. Thorpe , D.Phil (Oxon) *CNRS Centre de Recherche Cerveau et Cognition* *Toulouse, France*
Overige leden:	Prof. Dr. W.R. van Zwet Prof. Dr. G. Rozenberg Prof. Dr. H.A.G. Wijshoff

The work in this thesis has been carried out under the auspices of the research school IPA (Institute for Programming research and Algorithmics).

Work carried out at the Centre for Mathematics and Computer Science (CWI), Amsterdam.

Spiking Neural Networks
Sander Marcel Bohte.
Thesis Universiteit Leiden. - With ref.

ISBN 90-6734-167-3
© 2003, Sander Bohte, all rights reserved.
Layout kindly provided by Dr. Dick de Ridder, Delft.

“Loyalty to petrified opinion never yet broke a chain or freed a human soul.”

– Mark Twain

CONTENTS

---

1. Introduction 1
  1.1 Artificial Neural Networks 1
  1.2 Computing with asynchronous spike-times 4

2. Unsupervised Clustering with Spiking Neurons by Sparse Temporal Coding and Multi-Layer RBF Networks 11
  2.1 Introduction 11
  2.2 Networks of delayed spiking neurons 13
  2.3 Encoding continuous input variables in spike-times 18
  2.4 Clustering with Receptive Fields 20
    2.4.1 Capacity 21
    2.4.2 Scale sensitivity 21
    2.4.3 Clustering of realistic data 23
  2.5 Hierarchical clustering in a multi-layer network 26
  2.6 Complex clusters 28
  2.7 Discussion and Conclusions 30

3. Error-Backpropagation in Temporally Encoded Networks of Spiking Neurons 33
  3.1 Introduction 34
  3.2 Error-backpropagation 35
  3.3 The XOR-problem 39
    3.3.1 Error gradient and learning rate 41
  3.4 Other Benchmark Problems 42
  3.5 Discussion 45
  3.6 Conclusion 49

4. A Framework for Position-invariant Detection of Feature-conjunctions 51
  4.1 Introduction 51
  4.2 Local Computation with Distributed Encodings 55

- 4.2.1 Architecture. 55
- 4.2.2 Neural data-structure. 58
- 4.2.3 Local Feature Detection. 58
- 4.2.4 Local Feature Binding. 58
- 4.2.5 Conjunction detection. 59
- 4.3 Implementation 61
- 4.4 Experiments 64
- 4.5 Discussion 68
- 4.6 Conclusions 71

5. Formal Specification of Invariant Feature-conjunction Detection 73
- 5.1 Introduction 73
- 5.2 Formal Description 74
- 5.3 Conclusion 78

6. The effects of pair-wise and higher order correlations on the firing rate of a post-synaptic neuron 79
- 6.1 Introduction 80
- 6.2 Mathematical Solution of the three-neuron problem 83
- 6.3 Calculating the Distribution with N Identical Neurons 84
- 6.4 An artificial neural network 91
  - 6.4.1 The neuron model 91
  - 6.4.2 Network Simulations 93
  - 6.4.3 Firing-rate of a post-synaptic neuron 97
  - 6.4.4 Estimation of the N-cluster distribution by entropy maximization 99
  - 6.4.5 The effects of varying bin-width on the maximal entropy distribution. 99
  - 6.4.6 The effects of network scaling 103
- 6.5 Discussion 105
- 6.6 Conclusion 108

7. The Biology of Spiking Neurons 109
- 7.1 Real Neurons Spike 109
- 7.2 Precision and Reliability of Real Spikes 111

Publications 117

Bibliography 119

Samenvatting 129

Curriculem Vitae 134

1

INTRODUCTION

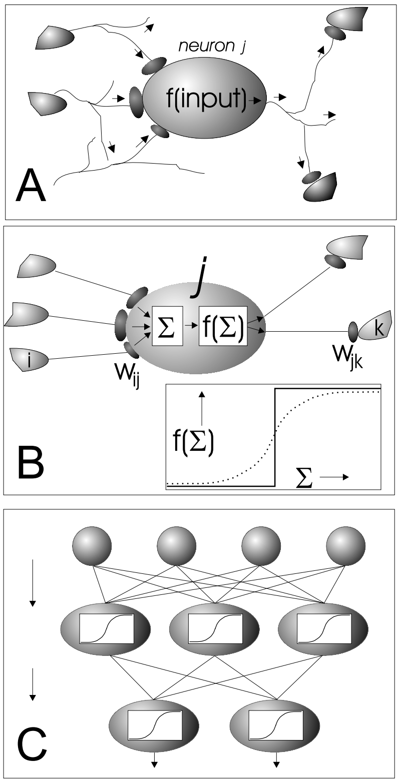
1.1 Artificial Neural Networks

Artificial neural networks attempt to understand the essential computations that take place in the dense networks of interconnected neurons making up the central nervous systems in living creatures (see also "On Networks of Artificial Neurons"). Originally, McCulloch and Pitts (1943) proposed a model based on simplified "binary" neurons, where a single neuron implements a simple thresholding function: a neuron's state is either "active" or "not active", and this is determined by calculating the weighted sum of the states of neurons it is connected to. For this purpose, connections between neurons are directed (*from* neuron i *to* neuron j), and have a weight (w_{ij}). If the weighted sum of the states of the neurons i connected to a neuron j exceeds some threshold, the state of neuron j is set to active, otherwise it is not.

Remarkably, networks of such simple, connected computational elements, can implement a range a mathematical functions relating input states to output states, and, with algorithms for setting the weights between neurons, these artificial neural networks can "learn" many such functions.

However, the limitations of these early artificial neural networks were amply recognized, i.e. see Minsky and Papert (1969). To alleviate these issues, the original binary thresholding computation in the neuron has often been replaced by the sigmoid: the sum of the weighted input into a neuron is mapped onto a real output value via a sigmoidal transformation-function, thus creating a graded response of a neuron to changes in its input. Abstracted in this transformation-function is the idea that real neurons communicate via *firing rates*: the rate at which a neuron generates action potentials (spikes). When receiving an increasing number of spikes, a neuron is naturally more likely to emit an increasing number of spikes itself.

On Networks of Artificial Neurons



The human brain consists of an intricate web of billions of interconnected cells called “neurons”. The study of neural networks in computer science aims to understand how such a large collection of connected elements can produce useful computations, such as vision and speech recognition.

A “real” neuron receives pulses from many other neurons. These pulses are processed in a manner that may result in the generation of pulses in the receiving neuron, which are then transmitted to other neurons (fig. A). The neuron thus “computes” by transforming input pulses into output pulses.

Artificial Neural Networks try to capture the essence of this computation: as depicted in figure B, the rate at which a neuron fires pulses is abstracted to a scalar “activity-value”, or *output*, assigned to the neuron. Directional connections determine which neurons are input to other neurons. Each connection has a weight, and the output of a particular neuron is a function of the sum of the weighted outputs of the neurons it receives input from. The applied function applied is called the *transfer-function*, $F(\Sigma)$. Binary “thresholding” neurons have as output a “1” or a “0”, depending on whether or not the summed input exceeds some threshold. Sigmoidal neurons apply a sigmoidal transfer-function, and have a real-valued output (inset fig. B, solid resp. dotted line). Neural networks are sets of connected artificial neurons. Its computational power is derived from clever choices for the values of the connection weights. Learning rules for neural networks prescribe how to adapt the weights to improve performance given some task. An example of a neural network is the *Multi-Layer Perceptron* (MLP, fig. C). Learning rules like *error-backpropagation* (Rumelhart et al., 1986) allow it to learn and perform many tasks associated with intelligent behavior, like learning, memory, pattern recognition, and classification (Ripley, 1996; Bishop, 1995).

With the introduction of sigmoidal artificial neurons, and learning rules for training networks consisting of multiple layers of neurons (Werbos, 1974; Rumelhart et al., 1986), some of the deficiencies of the earlier neural networks were overcome: the most prominent example was the ability to learn to compute the XOR function in an artificial neural network.

Since then, multi-layer networks of sigmoidal neurons have been shown to accommodate many useful computations, such as pattern classification, pattern recognition, and unsupervised clustering.

However, a problem referred to as "dynamic binding", has at best remained elusive to implement in neural networks. In 1949, Hebb already hypothesized that, to achieve sufficient flexibility and productivity in a neural network, it would be useful have the network dynamically "link" neurons that detect different properties of an object into assemblies. The purpose of an assembly would be to signal that its constituent neurons, each coding for different properties, are in fact part of the same object (Hebb, 1949). Objects composed of different "atomic" parts could thus be efficiently detected. An example would be to have a neuron that detects the color "red", and another neuron that detects a contour "apple-shaped". When linked together in an assembly, these neurons would indicate the presence of a "red apple". By having neurons that can each detect a particular "atomic" property, a linking mechanism allows the system to be productive, in the sense that by just having a limited set of detectors for atomic properties, any combination of these properties can be expressed: linking separate "red", "green", "yellow", "apple", "banana" and "pear" detectors allows the expression of nine differently colored objects.

In the presence of a single object composed of a number of properties, a simple "on-off" detector-signal for each property is sufficient to correctly signal the particular composition. However, in the presence of multiple objects this simple compositional signaling scheme is ambiguous (von der Malsburg, 1999), and more powerful means of "linking" atomic elements into composite structures (like "red apple") in neural networks have so far remained elusive at best (von der Malsburg, 1999), even though the usefulness of such schemes has been well recognized: e.g. for vision (Rosenblatt, 1961), speech recognition (von der Malsburg & Schneider, 1986), and the representation and manipulation of symbolic information (von der Malsburg, 1999). This led some even to arguing that the representation of compositional information is impossible in neural networks (Fodor & Pylyshyn, 1988).

The starting point of this thesis is the notion originally put forward by Von der Malsburg (1981), that a novel type of neural network, based on more detailed models of actual "real", *spiking* neurons, could help solve the binding-problem. Von der Malsburg proposed that in order to signal the binding of neurons coding for features that belong to the same object, these neurons would synchronize the times at which they emit spikes (the "synchrony-hypothesis"). Neurons coding for features belonging to different objects would then fire out of phase, allowing for multiple compositional objects to be represented simultaneously. The discovery of apparently assembly-dependent correlations between neurons by Gray et al. (1989) was interpreted as support for this idea, and much research into the temporal properties of spiking neurons has ensued since, both in neuroscience, as well as in computational modeling.

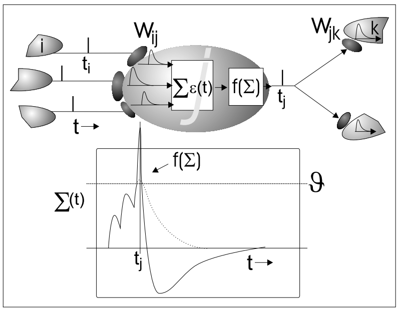
Although the synchrony-hypothesis has since come under increasing criticism (Shadlen & Movshon, 1999), the principal findings of research are indicating that the precision with which single spikes are emitted by biological neurons can be quite high, and it now seems very plausible that the timing of individual spikes conveys information to other neurons (see also chapter 7).

These findings have led to the proposal of more refined models of neural computation, such as the Asynchronous Spiking Neural Network by Maass (1996, 1997) (see also "On Artificial Spiking Neurons"). In this type of network, the precise process of the generation of a single action potential by a spiking neuron is modeled. Maass (1997) showed that when for such spiking neurons the input consists of asynchronously timed spikes (say a set of spikes $\{t_i, t_j, \ldots, t_k\}$, with t_i the time of a spike from input neuron i), the precise timing of the output spike can be interpreted as a computation on the input, just like for sigmoidal neurons. Theoretically, spiking neurons can perform very powerful computations with precisely timed spikes, as computational devises have been shown to be at least as computationally powerful as the sigmoidal neurons traditionally used in artificial neural networks (Maass, 1997).

1.2 Computing with asynchronous spike-times

The idea of neural computation with precisely timed spikes in networks of asynchronous spiking neurons is treated in detail in this thesis. We develop and extend algorithms that allow Asynchronous Spiking Neural Networks (ASNN's) to compute in ways traditionally associated with artificial neural networks, like pattern recognition and unsupervised clustering. Additionally, we investigate how spiking neurons could be used for solving the binding-problem: we propose a framework for dynamic feature binding based on the properties of *distributed coding* with populations of spiking neurons, and we investigate the most likely nature of the synchrony measured in biological systems.

On Artificial Spiking Neurons



As an artificial neuron models the relationship between the inputs and the output of a neuron, artificial spiking neurons describe the input in terms of single spikes, and how such input leads to the generation of output spikes. The transmission of a single spike from one neuron to another is mediated by *synapses* at the point where the two neurons interact. An input, or *presynaptic* spike arrives at the synapse, which in turn releases neurotransmitter which then influences the state, or *membrane potential* of the target, or *postsynaptic* neuron. When the value of this state crosses some threshold ϑ, the target neuron generates a spike, and the state is reset by a *refractory response*. The size of the impact of a presynaptic spike is determined by the type and efficacy (weight) of the synapse (see accompanying figure). In biology, neurons have one of two types of synapses: excitatory, where the synapses release neurotransmitter that increases the membrane potential of a target cell, and inhibitory synapses, that decrease this potential.

In Asynchronous Spiking Neural Networks, a model of this chain of events is taken as the transformation function of the neuron. In particular, given a set of precisely timed input spikes, say a set of input spikes $\{t_i^{(1)}, t_j^{(1)}, \ldots, t_k^{(1)}\}$, with $t_i^{(n)}$ the time of the nth spike from input neuron i, the time of the output spike is a function of these input spike-times. Changing the weights of the synapses alters the timing of the output spike for a given temporal input pattern. We can then interpret the timing of spikes in terms of neural computation (Thorpe & Gautrais, 1997; Maass & Bishop, 1999). In this thesis in particular, we interpret the value of input spikes relative to the first spike in a pattern, that is: early and late spike are associated with respectively a "high" and "low" value. Contrary to traditional neural networks, for spiking neurons not all input is equal: an "early" plus a "late" spike is not equal to two "medium" spikes: whether or not spikes arrive simultaneously can make a significant difference. This property of spiking neurons might allow more refined computations, as outlined in this introduction. Another important property of such spiking neurons is that as the input pattern is only defined in relative spike times, the computation in a target neuron can be considered scale-invariant: if say an input pattern is defined by spikes t_1 and t_2, a less salient version of this pattern would encoded by "later" input spikes $t_1 + \Delta$ and $t_2 + \Delta$. Since a receiving neuron is only triggered by the relative timings, it is effectively invariant to stimulus strength.

In **Chapter 2**, we present methods to enhance the precision, capacity and clustering capability of Asynchronous Spiking Neural Networks akin to (Natschläger & Ruf, 1998), thus overcoming limitations associated with the original network architecture. We encode continuous input variables each with a group of neurons (population coding) where the input neurons perform a transformation of the input value with Gaussian-type kernels. We consider high dimensional datasets, and for such datasets, we encode each input dimension separately. This yields an efficient, linear scaling of the required number of input neurons with increasing dimensionality of the input data.

With such encoding, we show that a feedforward, MLP-like spiking neural network is able to correctly cluster a number of datasets with relatively few spiking neurons, while enhancing cluster capacity and precision. The proposed encoding allows for the reliable and flexible detection of clusters in the data. By extending the network to multiple layers, we show how the network can correctly separate complex clusters by synchronizing in the hidden layers the neurons coding for parts of the same cluster. Together, these results demonstrate that asynchronous spiking neural networks can effectively perform unsupervised clustering of real-world data.

In **Chapter 3**, we derive a learning algorithm that changes the weights of an asynchronous spiking neural network by determining the exact error that the network makes on each example of a particular task (supervised learning). The algorithm is based on error-backpropagation (Werbos, 1974), and is derived analogously to that by Rumelhart et al. (1986) for sigmoidal neural networks. To overcome the discontinuous nature of spiking neurons, we approximate the point-process of spike-generation. We show that the algorithm is capable of learning complex non-linear tasks in asynchronous 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.

Chapter 4 is concerned with the design of a neural network architecture that is able to efficiently detect conjunctions of primitives (features) anywhere on a – large – input grid. This is a particular form of the binding-problem previously explained.

We propose a framework that can detect multiple conjunctions of features on an input-grid simultaneously, in an efficient, position-invariant manner. Our approach is based on the use of the properties of distributed representations in local nodes of the network: *local distributed representations*. Distributed representations refer to the idea that a collection of relatively a-specific features, when taken together, can be considered a unique identifier for an object that exhibits these features: features like *a bit red,* *somewhat green, spherical like, and shiny*, could fairly accurately identify an apple for instance.

Local distributed representations allow us to design *a-specific* detectors that locally detect the presence of a conjunction, like *a* color, and *a* shape. This local conjunction is then encoded in the distributed output of the local conjunction-detector. The outputs of all local detectors are aggregated in respective global, position-invariant detector, from which the *specific* feature-conjunctions are detected.

We show that this framework can be implemented in a feed-forward asynchronous spiking neural network, and that this network is capable of correctly detecting up to four simultaneously present feature-conjunctions.

Chapter 5 gives a formal definition of the framework introduced in Chapter 4, for neural nodes that process vector-like spiking activity. A set of n local spiking neurons each emitting a spike-train is defined as a tuple of spike-trains. Operators acting on this data-structure are defined for feature-detection, conjunction-detection, aggregation of multiple tuples to obtain respective position-invariant detector, and the feature-conjunction detector. The formal framework is illustrated in an example outlining the formal procedure for detecting a feature conjunction.

In **Chapter 6**, we examine the relationship between the expected spiking behavior of a group of spiking neurons, and a value that is typically measured in electro-physiological experiments: the pair-wise correlation.

It has been proposed that the precisely timed synchronous firing of neurons in the cortex could convey important information like whether different neurons are responding to the same object (von der Malsburg, 1981). Electro-physiological experiments have been argued to support this notion (Singer & Gray, 1995). These experiments recorded the correlation ρ between the firing-times of pairs of neurons. Such pair-wise correlations between neurons however do not uniquely determine the type of synchronized firing in the group of neurons as a whole.

We develop a framework in order to calculate the amount of synchronous firing that should exist in a neural network in which the (identical) neurons have known pair-wise correlations and higher order correlations are absent. We find that for the distribution with maximal entropy, events in which a large proportion of the neurons fire synchronously should exist, even for small ($\rho < 0.05$) values of the pair-wise correlation. We show that network simulations also exhibit these highly synchronous events in the case of weak pair-wise correlations.

In **Chapter 7**, we consider the biological background that serves as motivation for studying asynchronous spiking neural networks. In particular, we find that recent studies of neural information processing increasingly suggest that the precise timing of single spike is important in neural systems.

The fact that real neurons communicate via action-potentials – or spikes – is effectively undisputed. Whether or not these neurons – or neural systems in general – exhibit use the timing of single spikes is a fundamental question that is much debated, but has so far remained unresolved (Mainen & Sejnowski, 1995; Singer, 1999). Neuro-physiological experiments with neurons cultured in a dish have shown that the integration of impinging spikes in individual neurons is in principle reliable enough to support precise spike-time coding (Mainen & Sejnowski, 1995), and there are well known examples of specialized neural systems in animals for which the relevance of precise spike-times has been clearly demonstrated. Prominent examples are the electro-sensory system of the electric fish (Heiligenberg, 1991), the echolocation-system of bats (Kuwabara & Suga, 1993), and the auditory system of barn-owls (Carr & Konishi, 1990). Recent neurophysiological work has also uncovered that in the hippocampus of the brain, a precisely relationship can be found between the firing-rate of a neuron, and the timing of the first spike relative to the (slow) theta rhythm oscillations of the brain. The precision of these spikes is comparable to those we employ in our asynchronous spiking neural networks. It is proposed that in particular in the hippocampus this conversion of rate coding into temporal coding enables the compression of temporal sequences on a long ($>$1000ms) time scale into temporal spike-time patterns on the scales we consider (≈ 10ms).

As demonstrated by these examples, and others summarized in Chapter 7, there are most certainly cases where the precise timing of single spikes is important in the neural systems of animals. Precise spike-times seem to be found in particular when the relevant information in the animal's environment has a high temporal resolution (30-300ms). Notably, when neural systems that have to process information from such a fast environment are studied in an artificially slow environment, the temporal precision of single spikes is quickly lost, e.g. (de Ruyter van Steveninck, Borst, & Bialek, 2001).

From the collected evidence, the prediction seems to emerge that fast neural systems use fast neural coding, where the precise timing of single spikes is important. Tellingly, the human visual system has been shown to be capable of performing very fast classification (Thorpe, Fize, & Marlot, 1996), where a participating neuron can essentially fire at most one spike. On the time-scales involved, the relevant input of neurons further along the processing pathway thus consists of at most one spike per input neuron. The speed involved in decoding auditory information, and even the generation of speech also suggest that most crucial neural systems of the human brain operate *fast*.

These findings are only gradually altering the traditional belief that neuronal firing rate is the main means by which neurons communicate information. The success of traditional artificial neural networks that model firing-rates clearly contributes to this persistent notion. We demonstrate in this thesis that spiking neurons operating on precisely timed spikes can perform essentially the same types of computation as traditional artificial neural networks, and we also propose an architecture based on spiking neural networks that can perform computations that are particularly hard to implement in traditional, sigmoidal artificial neural networks which merely model neuronal firing-rates.

2

UNSUPERVISED CLUSTERING WITH SPIKING NEURONS BY SPARSE TEMPORAL CODING AND MULTI-LAYER RBF NETWORKS

ABSTRACT We demonstrate that spiking neural networks encoding information in the timing of single spikes are capable of computing and learning clusters from realistic data. We show how a spiking neural network based on spike-time coding and Hebbian learning can successfully perform unsupervised clustering on real-world data, and we demonstrate how temporal synchrony in a multi-layer network can induce hierarchical clustering. We develop a temporal encoding of continuously valued data to obtain adjustable clustering capacity and precision with an efficient use of neurons: input variables are encoded in a population code by neurons with graded and overlapping sensitivity profiles. We also discuss methods for enhancing scale-sensitivity of the network and show how the induced synchronization of neurons within early RBF layers allows for the subsequent detection of complex clusters.

2.1 Introduction

Hopfield (1995) presents a model of spiking neurons for discovering clusters in an input space akin to Radial Basis Functions. Extending on Hopfield's idea, Natschläger and Ruf (1998) propose a learning algorithm that performs unsupervised clustering in spiking neural networks using spike-times as input. This model encodes the input patterns in the delays across its synapses and is shown to reliably find centers of high-dimensional clusters. However, as we argue in detail in section 2.2, this method is limited in both cluster capacity as well as in precision.

We present methods to enhance the precision, capacity and clustering capability of a network of spiking neurons akin as in (Natschläger & Ruf, 1998), in a flexible and scalable manner, thus overcoming limitations associated with the network architecture. Inspired by the local receptive fields of biological neurons, we encode continuous input variables by a population code obtained from neurons with graded and overlapping sensitivity profiles. In addition, each input dimension of a high dimensional dataset is encoded separately, avoiding an exponential increase in the number of input neurons with increasing dimensionality of the input data. With such encoding, we show that the spiking neural network is able to correctly cluster a number of datasets at low expense in terms of neurons while enhancing cluster capacity and precision. The proposed encoding allows for the reliable detection of clusters over a considerable and flexible range of spatial scales, a feature that is especially desirable for unsupervised classification tasks as scale-information is a-priori unknown.

By extending the network to multiple layers, we show how the temporal aspect of spiking neurons can be further exploited to enable the correct classification of non-globular or interlocking clusters. In a multi-layer RBF network, it is demonstrated that the neurons in the first layer center on components of extended clusters. When all neurons in the first RBF layer are allowed to fire, the (near) synchrony of neurons coding for nearby components of the same cluster is then distinguishable by a subsequent RBF layer, resulting in a form of hierarchical clustering with decreasing granularity. Building on this idea, we show how the addition of lateral excitatory connections with a SOM-like learning rule enables the network to correctly separate complex clusters by synchronizing the neurons coding for parts of the same cluster. Adding lateral connections thus maintains the low neuron count achieved by coarse coding, while increasing the complexity of classifiable clusters.

Summarizing, we show that temporal spike-time coding is a viable means for unsupervised computation in a network of spiking neurons, as the network is capable of clustering realistic and high-dimensional data. Adjustable precision and cluster capacity is achieved by employing a 1-dimensional array of graded overlapping receptive fields for the encoding of each input variable. By introducing a multi-layer extension of the architecture we also show that a spiking neural network can cluster complex, non-Gaussian clusters. Combined with our work on supervised learning in spiking neural networks (chapter 3), these results show that single spike-time coding is a viable means for neural information processing on real-world data within the novel paradigm of artificial spiking neural networks.

This chapter is organized as follows: we describe the spiking neural network and limitations in section 2.2. In section 2.3 we introduce a means of encoding input-data to overcome these limitations, and clustering examples using this encoding are given in section 2.4. In section 2.5 we show how the architecture can be extended to a multi-layer RBF network capable of hierarchical clustering, and in section 2.6 we show how the addition of lateral connections enables the network to classify more complex clusters via synchronization of neurons within an RBF layer. A discussion of the results and conclusions are given in section 2.7.

2.2 Networks of delayed spiking neurons

In this section, we describe the spiking neural network as introduced for unsupervised clustering in (Natschläger & Ruf, 1998), as well as the results and open questions associated with this type of network.

The network architecture consists of a feedforward network of spiking neurons with multiple delayed synaptic terminals (figure 2.1). As briefly explained in chapter 1, spiking neurons generate action potentials, or spikes, when the internal neuron state variable, called '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 (1995). 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 a simulation interval (the presentation of an input pattern), 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

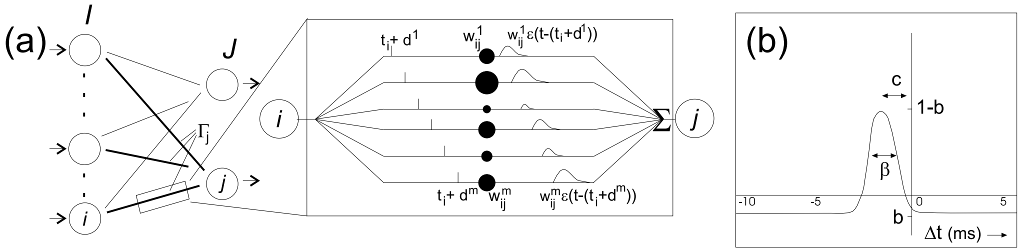


Figure 2.1: (a) Network connectivity and a single connection composed of multiple delayed synapses. Neurons in layer J receive connections from neurons Γ_j in layer I. Inset: a single connection between two neurons consists of m delayed synaptic terminals. A synaptic terminal k is associated with a weight w_{ij}^k, and delay d^k. A spike from neuron i thus generates m delayed spike-response functions ($\varepsilon(t-(t_i+d^k))$), the sum of which generates the membrane-potential in neuron j. (b) Graph of the learning function $L(\Delta t)$. The parameter Δt denotes the time-difference between the onset of a PSP at a synapse and the time of the spike generated in the target neuron.

spike-response function $\varepsilon(t)$ weighted by the synaptic efficacy ("weight") w_{ij}:

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

In this chapter, all weights are strictly positive (excitatory), in other chapters, we take some neurons to be excitatory, and some to be inhibitory, meaning that the connections projecting from these neurons all have strictly positive, respectively strictly negative weights.

The spike-response function $\varepsilon(t - t_i)$ in (2.1) models how the arrival of a single (unweighted) spike changes the membrane-potential of the target neuron as a function of time-since-impact. This function is referred to as the post-synaptic potential (PSP). The height of the PSP is modulated by the synaptic weight w_{ij} to obtain the effective post-synaptic potential at a neuron j due to a spike from neuron i. The spike-response function as used in our experiments is defined in (2.3).

In an extension on the basic spiking neuron model described above, 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 (after (Natschläger & Ruf, 1998), see also figure 2.1A, inset). 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 . We describe a pre-synaptic spike at a synaptic terminal k as a PSP of standard height with delay d^k. The unweighted contribution $y_i^k(t)$ of a single synaptic terminal to the state variable is then given by

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

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 (e.g. (Natschläger & Ruf, 1998)) for $t > 0$ (else 0), thus implementing a leaky-integrate-and-fire spiking neuron. The parameter 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:

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

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 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.

Input patterns can be encoded in the synaptic weights by local Hebbian delay-learning where, after learning, the firing time of an output neuron reflects the distance of the evaluated pattern to its learned input pattern thus realizing a kind of RBF neuron (Natschläger & Ruf, 1998). For unsupervised learning, a Winner-Take-All learning rule modifies the weights between the input neurons and the neuron first to fire in the output layer using a time-variant of Hebbian learning: if the start of a PSP at a synapse slightly precedes a spike in the output neuron, the weight of this synapse is increased, as it had significant influence on the spike-time via a relatively large contribution to the membrane potential. Earlier and later synapses are decreased in weight, reflecting their lesser impact on the output neuron's spike time. For a weight with delay d^k from neuron i to neuron j we use

$$\Delta w_{ij}^k = \eta L(\Delta t) = \eta(1-b)e^{-\frac{(\Delta t - c)^2}{\beta^2}} + b, \tag{2.5}$$

after (Natschläger & Ruf, 1998) (depicted in figure 2.1b), where the parameter b determines the effective size of the integral over the entire learning window (usually negative), β sets the width of the positive part of the learning window, and c determines the position of this peak. The value of Δt denotes the time difference between the onset of a PSP at a synaptic terminal and the time of the spike generated in the winning output neuron. The weight of a single terminal is limited by a minimum and maximum value, respectively 0 and w_{max}, where learning drives the individual weights of the synaptic terminals to one of the extremes.

If an input neuron were to precede the firing of the output-neuron by a fixed amount Δt_{ij}, the set of connecting delayed terminals that is positively reinforced is determined by the width of the positive part of the learning window: consider a single connection with n delays, $\{d_1 \ldots d_n\}$. For an input with a fixed Δt_{ij}, the width of the learning window, as determined by β, will increase the weights of a minimal number of m consecutive delayed synaptic weights: $\{d_j \ldots d_{j+m}\}$ (for some j). When learning with this time difference Δt_{ij} is repeated, ultimately all m weights are driven to w_{max}. The process of learning a cluster thus results in a minimal value for the effective weight (efficacy) between an input neuron that codes for part of a cluster, and the corresponding output neuron, both in length of time, as well as in size. Larger efficacies can be learned when a cluster extends over a larger temporal width (i.e., Δt_{ij} varies over some range $[t_s, t_{s+u}]$), and more weights are thus driven to the maximal value. If the temporal variation becomes too large, the average delayed weight adjustment due to (2.5) becomes negative, as the integral over the learning-window is then negative, and all weights converge to zero. This effect allows neurons in the network to ignore inputs that only contribute "noise" (see also (Natschläger & Ruf, 1998)). This dynamic recruitment of delayed terminals negates the need for overall weight normalization (see also the delay selection in (Gerstner et al., 1996)).

An input (data-point) to the network is coded by a pattern of firing times within a coding interval T_Δ and each input neuron is allowed to fire at most once during this interval. In our experiments, following (Natschläger

& Ruf, 1998), we set T_Δ to $[0 - 9]$ ms and the delays d^k to $\{1, \ldots, 15\}$ [ms] in 1 ms intervals ($m = 16$). For the experiments, the parameter values for the learning function $L(\Delta t)$ are set to: $b = -0.2$, $c = -2.85$, $\beta = 1.67$, and $\eta = 0.0025$. We use the α-function with τ = 3.0 ms. The parameter values are taken from (Natschläger & Ruf, 1998); deviations from these defaults in experiments are noted.

To clarify the rationale behind the selection of the respective parameters, we briefly discuss their effects. Contrary to the experiments in (Natschläger & Ruf, 1998), the majority of the input-neurons in our network does not fire: we found that a larger value of c was required to select a stable subset of synaptic terminals. Values between approximately 2 and 3(ms) yielded stable results. For smaller and in particular larger values, the selected delayed terminals tended to drift either to zero or out of range, effectively negating the connection. In the experiments, any value of β that selected a minimum of three consecutive delayed terminals typically yielded better results than other settings. Provided that the value c results in stable weight selection, the values of b and β determine the minimal extent of the clusters learned. Despite this minimal extend, we do not lose generality with these fixed parameters, provided that we use the input-encoding as outlined in Section 2.3.

Previous Results and Open Questions. In Natschläger and Ruf (1998) it was shown that artificially constructed clusters of inputs firing within the encoding interval are correctly clustered in an unsupervised manner, but the type of clusters they consider limits applicability. For N input neurons, a cluster C in (Natschläger & Ruf, 1998) is of dimensionality $M \leq N$, with M-dimensional location $\{s_1, \ldots s_M\}$, s_i being the spike-time of input neuron i. For such a setup it was found that the RBF neurons converged reliably to the centers of the clusters, also in the presence of noise and randomly spiking neurons.

In practice, problems arise when applying this scheme to more realistic data. A first issue concerns the *coding* of input: following the aforementioned method, we were not able to successfully cluster data containing significantly more clusters than input-dimensions, especially in the case of low dimensionality. This problem is associated with the minimum width β of the learning function $L(\Delta t)$, leading to a fixed minimal spatial extent of a learned cluster, potentially (much) larger than the actual cluster size. In fact, for 2-dimensional input containing more than two clusters, the above algorithm failed in our experiments for a wide range of parameters. Furthermore, the finite width of the learning rule effectively inhibits the detection of multiple nearby clusters of smaller size relative to the width of the learning function, requiring advance knowledge of the effective cluster-scale. Hence, to achieve practical applicability, it is necessary to develop an *encoding* that is scalable in terms of cluster capacity and precision and that is also efficient in terms of the number of input-neurons required. In the following section, we present improvements to the architecture that address these issues.

2.3 Encoding continuous input variables in spike-times

To improve the encoding precision and clustering capacity, we introduce a method for encoding input-data by population coding. The aim of this encoding is to increase the temporal distance between the temporal input-patterns associated with respective (input) data-points. Since we use delayed terminals with a resolution of 1 ms, the discriminatory power of the unsupervised learning rule is naturally limited to approximately this resolution. The encoding increases the temporal distance between points, and thus the separability of clusters. Although our encoding is simple and elegant, we are not aware of any previous encoding methods for transforming continuous data into spike-time patterns and therefore, we describe the method in detail.

As a means of population coding, we use multiple local receptive fields to distribute an input variable over multiple input neurons. Such a population code where input variables are encoded with graded and overlapping activation functions is a well-studied method for representing real-valued parameters (Eurich & Wilke, 2000; Baldi & Heiligenberg, 1988; Snippe & Koenderink, 1992; Zhang et al., 1998; Zhang & Sejnowski, 1999; Pouget et al., 1999). In these studies, the activation function of an input-neuron is modeled as a local receptive field that determines the firing rate. A translation of this paradigm into relative firing-times is straightforward: an optimally stimulated neuron fires at $t = 0$, whereas a value up to say $t = 9$ is assigned to less optimally stimulated neurons (depicted in figure 2.2).

For actually encoding high-dimensional data in the manner described above, a choice has to be made with respect to the dimensionality of the receptive-fields of the neurons. We observe that the least expensive encoding in terms of neurons is to independently encode each of the respective input variables: each input-dimension is encoded by an array of 1-dimensional receptive fields. Improved representation accuracy for a par-

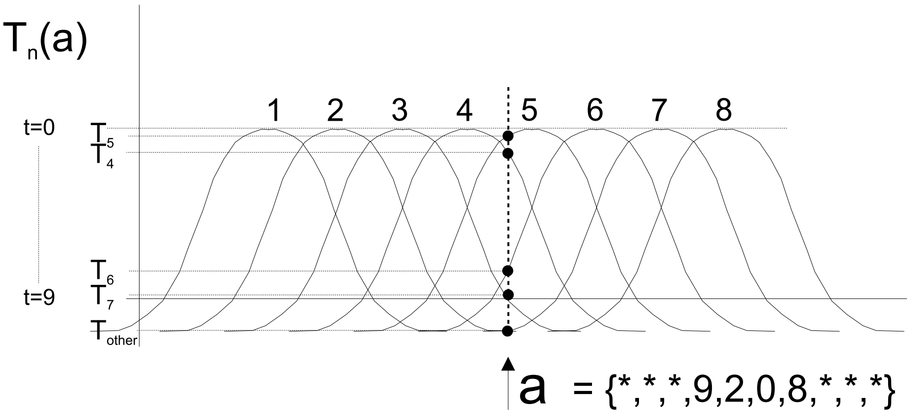


Figure 2.2: Encoding with overlapping Gaussian receptive fields. An input value a is translated into firing times for the input-neurons encoding this input-variable. The highest stimulated neuron (5), fires at a time close to 0, whereas less stimulated neurons, as for instance neuron 7, fire at increasingly later times.

ticular variable can then be obtained by sharpening the receptive fields and increasing the number of neurons (Zhang & Sejnowski, 1999). Such coarse coding has been shown to be statistically bias-free (Baldi & Heiligenberg, 1988) and in the context of spike-time patterns we have applied it successfully to supervised pattern classification in spiking neural networks (Bohte, Kok, & La Poutré, 2002b).

In our experiments, we determined the input ranges of the data, and encoded each input variable with neurons covering the whole data-range. For an input variable n with minimum value I^n_{min} and maximum value I^n_{max}, m neurons were used with Gaussian receptive fields. For the i^{th} neuron coding for variable n, the center of the Gaussian was set to $I^n_{min} + \frac{2i-3}{2} \cdot \frac{\{I^n_{max} - I^n_{min}\}}{m-2}$ $(m > 2)$, positioning one input neuron outside the data-range at both ends. The width was set to $\sigma = \frac{1}{\gamma} \frac{\{I^n_{max} - I^n_{min}\}}{m-2}$ (with $m > 2$). For γ, a range of values was tried, and, unless stated otherwise, for the 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 $N \times m(n)$ values between 0 and 1 (N: dimension of data, $m(n)$: number of neurons used to encode dimension n). 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-times were rounded to the nearest internal time-step, and neurons with responses larger than $t = 9$ were coded to not fire, as they were considered to be insufficiently excited. The encoding of a single input-value a by receptive field population coding is depicted in figure 2.2.

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, enabling us to process only a list of "active" connections, instead of all connections (event-based network simulation, Delorme, Gautrais, VanRullen, and Thorpe (1999)). Also, by encoding each variable independently, we achieve a coarse coding where each variable can be encoded by an optimal number of neurons while maintaining an efficient use of neurons.

2.4 Clustering with Receptive Fields

We investigate the clustering capabilities of spiking neural networks where the input is encoded with receptive fields. With such encoding, each data-point is translated into a multi-dimensional vector of spike-times (spike-time vector). Clustering relies on a single output neuron firing earlier than the other output neurons for data-points from a single cluster. The optimal activation of such an output neuron is achieved when the spikes of input neurons arrive at the output neuron simultaneously. This is what the Hebbian learning-rule (2.5) accomplishes, provided that the input lies within a particular cluster. If the distance between clusters is sufficient, the winner-takes-all competition between output neurons tunes these output neurons to the spike-time vectors associated with the centers of the respective clusters. The activation of a neuron for a given pattern then depends on the distance between the optimal and actual spike-time vector, resulting in increasingly later firing times (or none) with increasing distance from the cluster-center. We will use this diverging temporal firing pattern later for subsequent multi-layer clustering.

The encoding described in section 2.3 enhances capacity and precision as compared to the original architecture in (Natschläger & Ruf, 1998). In this section, we show this for a number of artificial and real-world datasets, both for low- as well as for high-dimensional input. In section 2.4.1, we show examples of improved capacity and precision, in section 2.4.2 a method for enhanced scale-sensitivity is shown, and in section 2.4.3 a examples of real-world clustering tasks are given.

2.4.1 Capacity In this section, we report on experiments that show how the outlined encoding allows for increased capacity, e.g. by encoding variables with more neurons, many different clusters can be separated.

In experiments, we cluster input consisting of two separately encoded variables, and found that a network with 24 input neurons (each variable encoded by 12 neurons) was easily capable of correctly classifying 17 evenly distributed clusters, demonstrating a significant increase in the clustering capacity (figure 2.3a). After presenting 750 randomly chosen data-points, all 1275 cluster points were correctly classified. In figure 2.3b the correct clustering of less regular input is shown. In general, we found that for such single layer RBF networks, capacity was only constrained by cluster separability. By decreasing the width of the receptive fields while increasing the number of neurons, increasingly narrowly separated clusters could be distinguished (just as predicted by theoretical work on the properties of receptive field encodings, e.g. (Zhang & Sejnowski, 1999)).

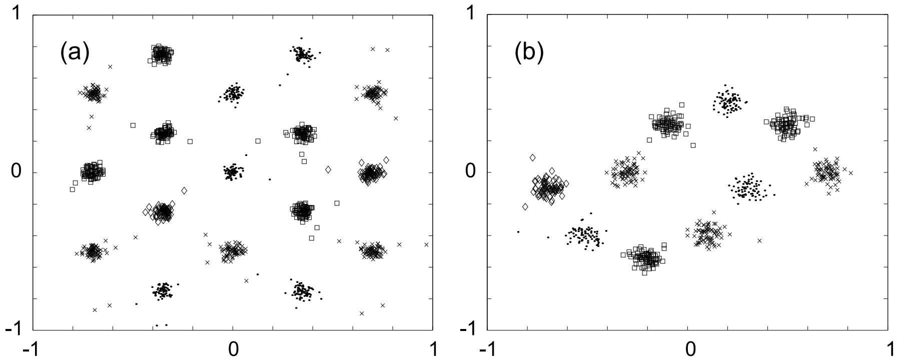


Figure 2.3: (a) Some 17 clusters in 2-d space, represented by two one-dimensional input variables, each variable encoded by 12 neurons (5 broadly tuned, 7 sharply tuned).(b) Classification of 10 irregularly spaced clusters. For reference, the different classes as visually extractable were all correctly clustered, as indicated by the symbol/graylevel coding.

2.4.2 Scale sensitivity Encoding input variables with local receptive fields incorporates an inherent choice of spatial scale sensitivity by fixing the width of the Gaussian; using a mix of neurons with varying receptive field widths proved to significantly enhance the range of detectable detail. In experiments, we found that the use of a mixture of receptive field sizes increased the range of spatial scales by more than an order of magnitude on a number of artificially generated datasets, and in general the clustering reliability improved.

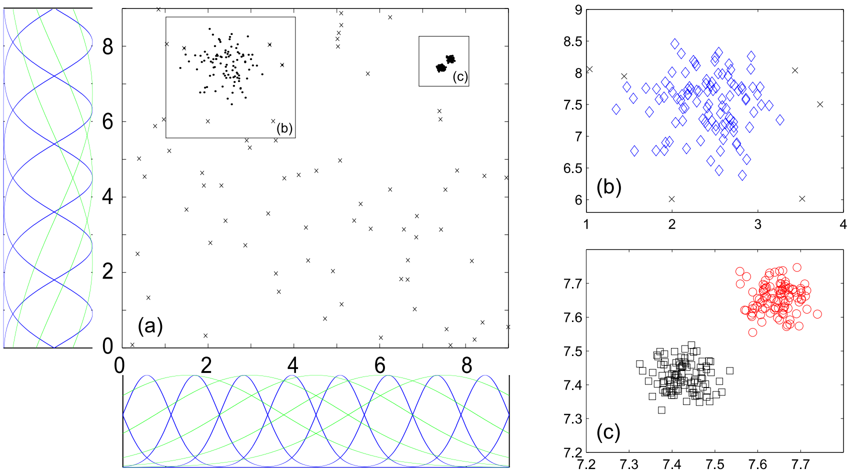


Figure 2.4: (a) Three clusters (upper left and upper right) of different scale with noise (crosses). (b,c) Insets: actual classification. Respective classes are marked with diamonds, squares, and circles. Noise outside the boxes or points marked by x's did not elicit a spike and were thus not attributed to a class. Side panels: graded receptive fields used.

The multi-scale encoding was implemented by assigning multiple sets of neurons to encode a single input dimension n. For different scales, say I and J, each scale was encoded with increasingly less neurons, scale I encoded with n_i neurons and scale J with n_j neurons, with $n_i < n_j$. As a set of neurons is evenly spread out over the data range, the width of the receptive field scales inversely proportional to the number of neurons, achieving multi-scale sensitivity as illustrated in the clustering example in figure 2.4. Data consisted of one large (upper left) and two small (upper right) Gaussian clusters. The input variables were encoded with 15 neurons for the variable along the x-axis, and 10 input neurons for the y-variable. These neurons were given a mixture of receptive field widths, 3 broad and 7 tight Gaussians for the y-variable, and 5 broad and 10 tight Gaussians for the x-variable (depicted in the side panels). The width σ_t of the tight Gaussians was set to $\frac{1}{\gamma}(I_{max} - I_{min})/(m-2)$, with $\gamma = 1.5$. The width σ_b of the broad Gaussians was set to $\frac{1}{\gamma}(I_{max} - I_{min})/(m+1)$, with $\gamma = 0.5$. This results in widths of respectively $\sigma_b = 4.5$ and $\sigma_t = 1.2$ (y-axis), and $\sigma_b = 3$ and $\sigma_t = 0.5$ (x-axis). The tight Gaussians were distributed along the respective axes as outlined in section 2.3, the broad Gaussians were all evenly placed with their centers *inside* the respective data-ranges, with center i placed at $I^n_{min} + i \cdot \frac{\{I^n_{max} - I^n_{min}\}}{m+1}$. Note that the width of the small clusters is still substantially smaller than the receptive field sizes of the tight Gaussians. As the spike-time vectors for a particular data-point are derived from the activation values of the population of neurons, the spike-time vectors corresponding to the respective cluster centers are still sufficiently distant to make discrimination possible.

The learning-rule successfully centered the output-neurons on the clusters, even though the large cluster is almost an order of magnitude larger than the two small clusters combined. When using a uniform receptive field size, the same size network often failed for this example, placing two neurons on the large cluster and one in between the two small clusters. Similar configurations with other datasets showed the same behavior, demonstrating increased spatial scale sensitivity when encoding the input with multiple sets of receptive field sizes.

In an unsupervised setting, scale is typically not, or not well, known (e.g. (Guedalia, London, & Werman, 1999)). Encoding the input with a mixture of receptive widths thus adds multi-scale sensitivity while maintaining the network architecture and learning rule.

2.4.3 Clustering of realistic data Good results were also obtained when classifying more realistic and higher dimensional data. As an example of relatively simple but realistic data, we clustered Fisher's 4-dimensional Iris data-set. The input was encoded in 4×8 input neurons, classification yielded 92.6 ± 0.9 % correct classification (over 10 runs, with 1 failing clustering removed and with parameter settings as outlined in section 2.2). Alternative clustering methods, like k-means[1] and a Self-Organizing Map (SOM)[2], yielded somewhat worse results, see table 2.1. Since our SOM and k-Means methods can probably be improved upon, this result indicates that the clustering capability of the RBF network is at least competitive with similar methods.

To assess the feasibility of using the RBF network for clustering in high-dimensional data, a number of artificial data-sets (10-D+) were generated

---

[1] from SOMToolbox at `www.cis.hut.fi/projects/somtoolbox/`.

[2] from Matlab R12.

Iris clustering	
method	error training-set
Spiking RBF	92.6% ± 0.9%
k-Means	88.6% ± 0.1%
SOM	85.33% ± 0.1%

Table 2.1: Unsupervised clustering of Fisher's Iris-dataset. The k-Means method was set to $k = 3$, SOM was run with 3 output neurons.

(not shown). In all experiments, the spiking RBF network correctly classified these datasets.

To show the viability of clustering with spiking neural networks on a more "real-life" unsupervised clustering task, we trained the network to classify a set of remote sensing data. This task is a more realistic example of unsupervised clustering in the sense that the data consists of a large number of data-points, has non-Gaussian classes, and probably contains considerable noise. The distribution of the data-points over the classes is also ill-balanced: some classes have many data-points and others only a few (e.g. grasslands vs. houses). As an example, we took a $103 \times 99 = 10197$ data-points of the full 3-band RGB image shown in figure 2.5a, and compared the classification obtained by the RBF network of spiking neurons to that of a SOM-network, both for the detection of 17 classes. As a benchmark, we use the results obtained by the UNSUP clustering algorithm for remote sensing (Kemenade, La Poutré, & Mokken, 1999) on the entire image (figure 2.5b). Figure 2.5c shown the classification of the area with a SOM-network, and figure 2.5d shows the classification by the spiking neural network. Note that both methods required approximately the same amount of computer runtime. When comparing figures 2.5c and 2.5d to the UNSUP classification, visual inspection shows that the respective classifications do not differ much, although some clusters detected by the RBF network are due to multiple neurons centered on the same class: both RBF and SOM classifications seem reasonable. Although the correctness of remote sensing classifications is notoriously difficult to determine due to lack of ground evidence (labeled data), the results show that our RBF network is robust with respect to ill-balanced, non-Gaussian and noisy real-world data.

Summary. The experiments show that capacity and precision in spiking RBF networks can be enhanced such that they can be used in practice. The simulation of spiking neurons in our implementation is quite compu-

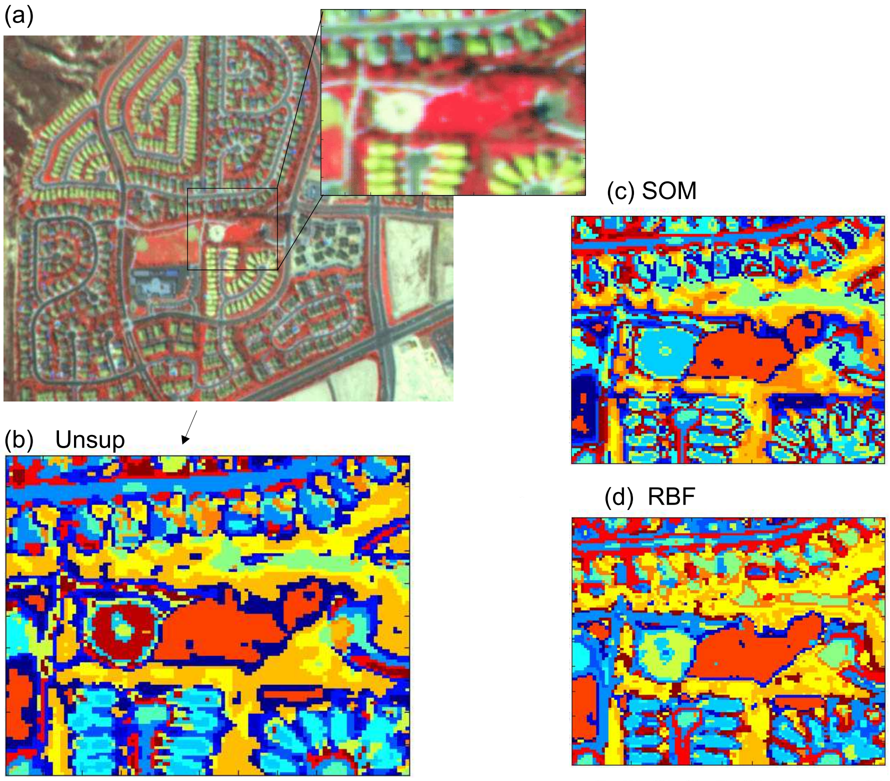


Figure 2.5: (a) The full image. Inset: image cutout actually clustered. (b) Classification of the cutout as obtained by clustering the entire image with UNSUP. (c) Classification of the cutout as obtained by clustering with SOM algorithm. (d) Spiking neural network RBF classification of the cutout image after learning from 70,000 randomly drawn data-points from the 103x99 image.

tationally intensive as compared to the optimized clustering by UNSUP (minutes vs. seconds), but takes approximately the same amount of time as SOM methods, and is only somewhat slower than k-Means (though run in Matlab). Possible speedups could be accomplished by using more computationally efficient spiking neural network models, for instance by taking a spike as an "event" and interpolating all deterministic effects between these events, e.g. the time-evolution of the membrane-potential under a set of preceding PSP's (Mattia & Giudice, 2000).

2.5 Hierarchical clustering in a multi-layer network

With a few modifications to the original network, we can create a multilayer network of spiking neurons that is capable of hierarchical clustering based on temporal cluster distance. Cluster boundaries in real data are often subjective, and hierarchical classification is a natural approach to this ambiguity, e.g. (Koenderink, 1984). By classifying data with increasing or decreasing granularity based on a cluster-distance measure, multiple "views" of a dataset can be obtained. To enable hierarchical clustering in spiking RBF neurons, we observe that the differential firing times of output neurons are a monotonic decreasing function of spatial separation, e.g. the further a data-point lies from the center of a neuron, the later the neuron fires. This could serve as a cluster-distance measure.

To achieve such hierarchical clustering, we created a multi-layer network of spiking neurons. Given a suitable choice of neurons within the layers, respective layers yield the classification of a data-point at a decreasing level of granularity as compared to the classification in a previous layer. The combined classification of all layers then effectively achieves hierarchical classification. To enable hierarchical classification with decreasing granularity, the size of the neural population decreases for subsequent layers, and all n neurons within a layer are allowed to fire such that the next layer with m neurons can extract up to m clusters from "input" n neurons firing, with $m < n$. The clustering mechanism is maintained by only modifying the weights for the winning neuron within a layer.

To implement hierarchical clustering in such a fashion, we added a second RBF layer to the network as described above, and successfully trained this network on a multitude of hierarchically structured datasets. An example is shown in figure 2.6: the data contained two clusters each consisting of two components. The winner-take-all classification found in the first RBF layer is shown in figure 2.6a, and correctly identifies the components of the two clusters. For a configuration as in figure 2.6a, the receptive field of any RBF neuron extends over the accompanying component. In this case, the evaluation of a single data point elicits a response from both neurons in the cluster, albeit one somewhat later than the other. The neurons centered on the other cluster are insufficiently stimulated to elicit a response. This disparate response is sufficient for the second layer to concatenate the neurons in the first layer into two clusters (figure 2.6b). Thus, as we extend the network with subsequent RBF layers comprising of fewer neurons, in effect we achieve hierarchical clustering with decreasing granularity: nearby components are compounded in the next layer based on relative spatial proximity as expressed in their temporal distance.

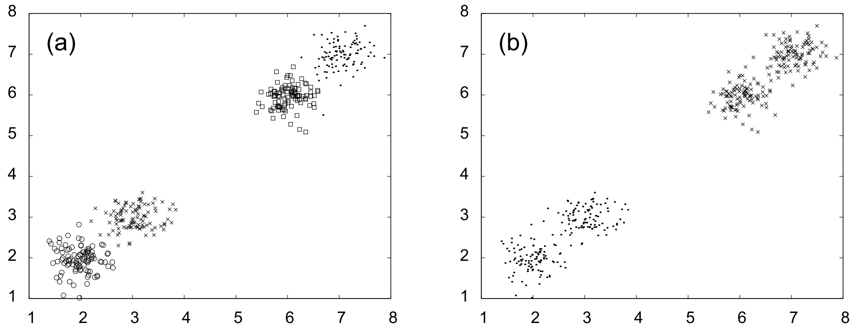


Figure 2.6: Hierarchical clustering in a 2 layer RBF network. (a) Clustering in the first layer consisting of 4 RBF neurons. Each data-point is labeled with a marker designating the winning neuron (squares, circles, crosses, and dots). (b) Clustering in the second layer, consisting of 2 RBF neurons. Again each data-point is labeled with a marker signifying the winning neuron (crosses and dots).

In unsupervised learning, the determination of the number of classes present in the dataset is a well-known problem in competitive winner-take-all networks, as it effectively is determined a-priori by the number of output neurons, e.g. (Zurada, 1992). In the hierarchical clustering example, we tuned the number of neurons to the number of components and clusters. In an RBF layer with more units than clusters or components, typically multiple output-neurons will become centered on the same cluster (experiments not shown), especially when clusters consisted of multiple components. Correct classification is only reliably achieved when the number of RBF neurons matches the number of clusters, see also (Natschläger & Ruf, 1998). However, in the case of more neurons than components/clusters the same hierarchical clustering principle holds, as multiple neurons centered on the same component are identifiable by their strong synchrony. Hence the relative synchronization of nearby neurons is an important clue when reading the classification from a layer, as well as an effective means of coding for further (hierarchical) neuronal processing. Note that the problem is rather one of extraction than of neuronal information processing, as multiple synchronized neurons are effectively indiscriminable downstream and can hence be considered to be one neuron.

2.6 Complex clusters

In this section, we show how temporal synchrony can be further exploited for separating interlocking clusters by binding multiple correlated RBF neurons via the addition of reciprocal excitatory lateral connections to the first RBF-layer, thus enhancing the network clustering capabilities.

Cluster boundaries in real data are often subjective. Hierarchical clustering is only part of the solution, as some measure for grouping components into subsequent clusters has to be implemented. For complex clusters, separate parts of the same cluster can easily be spatially separated to the point where the neuronal receptive fields no longer overlap: a neuron coding for one part will no longer respond when a data-point belonging to another part of the same cluster is presented. Another issue relates to the measure for concatenating components into clusters: only those components that have a certain density of data points "in between" should be concatenated, as implemented for instance in the UNSUP clustering algorithm (Kemenade et al., 1999). The situation is depicted in figure 2.7. The analogous problem exists when discriminating different clusters that are nearby. In both cases, when such clusters are covered by multiple neurons that are concatenated in a next layer, they might suffer from the fact that some of these neurons belonging to different clusters are in fact closer together than to other neurons in the same cluster (and thus fire closer together).

We present a SOM-like addition to the network to overcome this problem: by adding excitatory lateral connections to an RBF-layer and using a competitive SOM-like rule for modifying connections, nearby neurons become tightly coupled and are in effect bound together as they synchronize their firing times. As only the weights between temporally proximate neurons are augmented, ultimately neurons centered on the same cluster are synchronized due to the data points that lie "in between" neighboring neurons. These points elicit approximately the same time-response from the nearest neurons, strengthening their mutual connections. This process does not take place for neurons coding for different clusters, due to the relative lack of "in between" points (figure 2.7). As a set of neurons synchronize their respective firing-times when a data-point lying within a cluster-structure is presented to the network, the temporal response from the first RBF layer enables a correct classification in the second layer.

We implemented such lateral connections in a multi-layer network and successfully classified a number of artificial datasets consisting of interlocking clusters. The lateral connections were modeled as the feedforward

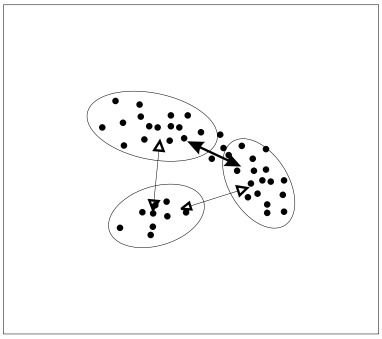


Figure 2.7: Weights connecting different part of a single cluster. Given two clusters of data-points (solid circles) classified by three RBF neurons (elliptic receptive fields), data-points between two RBF neurons strengthen the mutual lateral connections (solid arrows), whereas the connections to the equidistant third RBF neuron are not due to the lack of points "in between".

connections, albeit with only one delay $d^1 = 1$ ms. The lateral connections from the winning neuron are adapted using a "difference of Gaussians (DOG)" or "Mexican hat" learning function:

$$L(\Delta t) = e^{-\Delta t^2/b^2}\{(1-c)e^{-\Delta t^2/\beta^2} + c\}, \tag{2.6}$$

with $b = 4.5$, $c = -0.2$, $\beta = 0.8$. The "Mexican hat" learning functions defines the temporal difference for which connections are strengthened or weakened, where β determines the temporal width of the positive part of the learning function, and b determines the width of the weight depressing trough. During learning, the maximal allowed strength of the lateral connections is slowly increased from 0 to a value sufficiently strong to force connected neurons to fire. Experiments with these connections incorporated in the multi-layer network yielded the correct classification of complex, interlocking clusters. An example is shown in figure 2.8.

Summarizing, the addition of lateral excitatory connections with competitive SOM-learning synchronizes spatially correlated neurons within an RBF layer. This temporal property then enables the correct clustering of complex non-linear clusters in a multi-layer network, without requiring additional neurons.

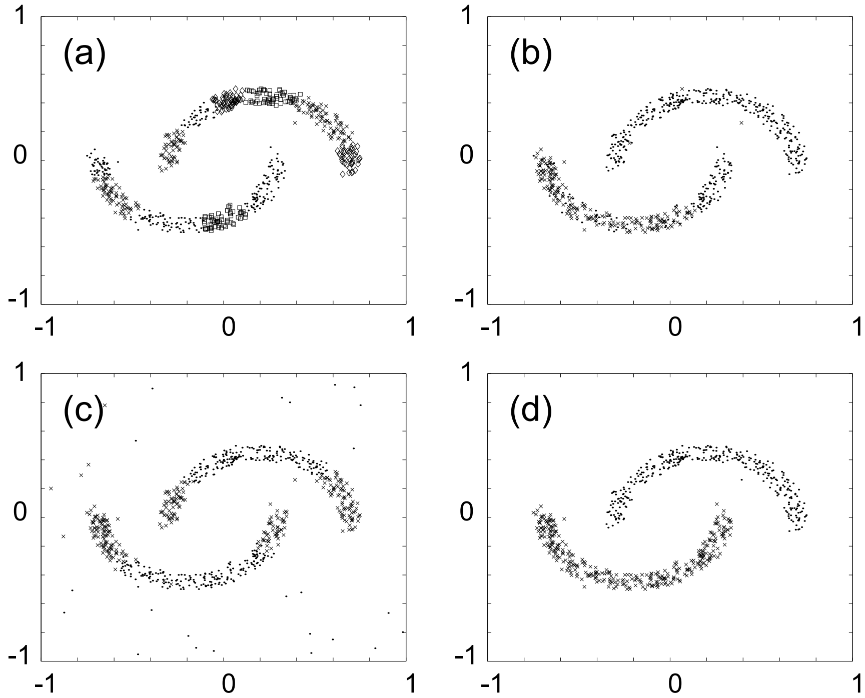


Figure 2.8: Clustering of two interlocking clusters in a multi-layer RBF network. (a) classification in the first layer: 11 outputs, the two clusters are spread over respectively 5 (upper cluster) and 6 neurons (lower cluster). The respective classifications are denoted by different markers and gray levels. (b) Incorrect clustering in the second layer with two RBF neurons and input from (a), without lateral connections. (c) Incorrect classification as obtained in a single-layer network. (d) Correct classification (100%) in the second layer, with lateral connections. Each input variable was encoded by 12 input neurons (3 broadly and 9 sharply tuned).

2.7 Discussion and Conclusions

We have shown that temporal spike-time coding in a network of spiking neurons is a viable paradigm for unsupervised neural computation, as the network is capable of clustering realistic and high-dimensional data. We investigated clustering for continuously valued input and found that our coarse coding scheme of the input data was effective and efficient in terms of required neurons. In a test on "real-life" data, our coarse coding approach proved to be effective on the unsupervised remote-sensing classification problem. Working from our findings, Goren (Goren, 2001) obtained similar results for slightly different problems.

To detect non-globular or complex interlocking clusters we introduced an extension of the network to allow for multiple RBF-layers, enabling hierarchical clustering. When we added excitatory lateral connections we showed that a competitive SOM-like lateral learning rule enhances the weights between neurons that code for nearby, uninterrupted cluster-parts. This learning leads to synchronization of neurons coding for the same cluster and was shown to enable the correct classification of larger cluster-like structures in a subsequent layer. Hence the combination of multi-layer RBF and competitive SOM-like lateral learning adds considerably to the clustering capabilities, while the number of neurons required remains relatively small. Also, we demonstrated how a local Hebbian learning-rule can both induce and exploit synchronous neurons resulting in enhanced unsupervised clustering capabilities, much as theorized in neurobiology.

The intuitive approach to within-layer synchronization as an aide for clustering is inspired by efforts to implement image-segmentation in neural networks via dynamic synchronization of spiking neurons that code for those parts of an image that are part of the same object, e.g. (König & Schillen, 1991; Chen & Wang, 1999; Campbell, Wang, & Jayapraksh, 1999). Clustering entails the classification of a data-point in terms of other data-points with "similar" properties in some, potentially high-dimensional, input-space, and is not necessarily concerned with the spatial organization of the data (e.g. the UNSUP remote sensing method used for figure 2.5). As such, clustering is essentially a different problem. For clustering, it is important that the number of neurons involved scales moderately with increasing dimensionality of the data, whereas image-segmentation is inherently two or three dimensional and is not, or less, subject to this restriction. However, our results lend further support for the use of precise spike timing as a means of neural computation and provide common ground in terms of the coding paradigm for these different problems.

3

ERROR-BACKPROPAGATION IN TEMPORALLY ENCODED NETWORKS OF SPIKING NEURONS

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. 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, the trained networks demonstrate that temporal coding is an effective code for fast neural information processing, and as such requires less neurons than instantaneous rate-coding. Furthermore, we find that reliable temporal computation in the spiking networks was only accomplished when using spike-response functions with a time constant longer than the coding interval, as has been predicted by theoretical considerations.

3.1 Introduction

In chapter 2, we demonstrated successful unsupervised classification with asynchronous spiking neural networks. To enable useful supervised learning with the temporal coding paradigm, we develop a learning algorithm for single spikes that keeps the 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. (1986). 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. In this chapter, we present, to the best of our knowledge, the first spiking neural network that is trainable in a supervised manner and as such demonstrates the effectiveness and efficiency of a functional spiking neural network as a function-approximator.

We also present results that support the prediction that, in order to allow for reliable temporal computation in a receiving neuron, the length of the rising segment of the post-synaptic potential needs to be longer than the length in time over which relevant spikes arrive (Maass, 1996). For spiking neurons, the post-synaptic potential describes the dynamics of a spike impinging onto a neuron, and is typically modeled as the difference of two exponentially decaying functions (Gerstner, 1998). 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 (Maass, 1996). From a biological perspective, these findings counter the common opinion among neuroscientists that fine temporal processing in spiking neural networks is prohibited by the relatively long time constants of biological cortical neurons (as noted for example by Diesmann, Gewaltig, and Aertsen (1999)).

This chapter is organized as follows: in section 3.2, we derive the error-backpropagation algorithm. In section 3.3 we test our algorithm on the classical XOR example, and we also study the learning behavior of the algorithm. By 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 3.4. The results of the experiments are discussed in section 3.5.

3.2 Error-backpropagation

We derive error-backpropagation, analogous to the derivation by Rumelhart et al. (1986). Equations are derived for a fully connected feedforward spiking neural 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 spiking neural network is modeled as explained in chapter 2, section 2.2.

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 factors on the right, we express t_j as a function of the thresholded post-synaptic input $x_j(t)$ around $t = t_j^a$. We assume 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 3.1. For such a small region, we approximate the threshold function $\delta t_j(x_j) = -\delta x_j(t_j)/\alpha$, 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, that is $\alpha = \frac{\partial x_j(t)}{\partial t}(t_j^a)$.

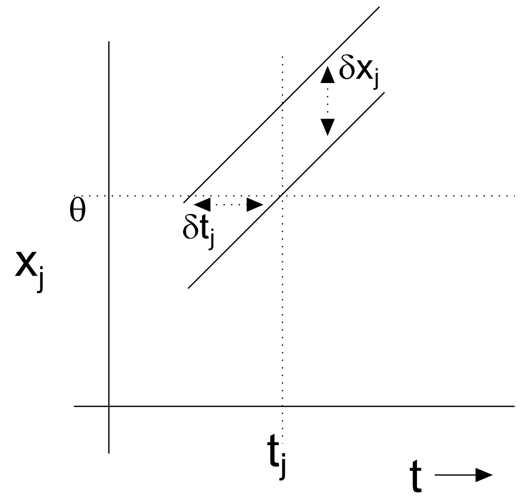


Figure 3.1: Relationship between δx_j and δt_j for an ϵ space around t_j.

The second factor in (3.3) 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 $\frac{\partial x_j(t_j^a)}{\partial t_j^a}$ for $\frac{\partial x_j(t)}{\partial t}(t_j^a)$.

We remark that this approximation only holds when the weights to a neuron are not altered such that the membrane potential no longer reaches threshold, and the neuron hence no longer fires. This is a potential problem, but can be countered by encoding input into the network in such a way that early spikes are automatically “more important” than later spikes. The encoding we outlined in chapter 2, section 2.3 is consistent with such spike-time evolution and should in general alleviate the problem: in our experiments it proved an effective solution. Note however that once a neuron no longer fires for *any* input pattern, there is no mechanism to "prop-up" the weights again. In our experiments, we set the initial weights such that each neuron in the network responded to at least part of the input-pattern. With this additional provision, we did not experience any problems with "silent" neurons.

Note also that the approximation might imply that for large learning rates the algorithm can be less effective. We will consider this issue in the application of the algorithm in section 3.3.1.

The first factor in (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 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:

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

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

As in (Bishop, 1995), in (3.11) we expand the local error $\frac{\partial E(t_i^a)}{\partial t_i^a}$ in terms 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 $\frac{\partial x_j(t_j^a)}{\partial t_i^a}$:

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

Hence,

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

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 $\Delta w_{hi}^{k} = -\eta y_{h}^{k}(t_i)\delta_i$ (3.14)

A simple modification of this scheme would be to include a momentum term:

$$\Delta w_{ij}^{k} = -\eta \Delta w_{ij}^{k}(t) + p w_{ij}^{k}(t-1), \tag{3.15}$$

with p the momentum variable. In a follow up to our work, Xin *et al.* (Xin & Embrechts, 2001) have shown that such modifications of the Spikeprop algorithm do indeed significantly speed up convergence.

3.3 The XOR-problem

In this section, we will apply the *SpikeProp* algorithm to the XOR-problem. 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 associate 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	$\rightarrow$	16
0	6	$\rightarrow$	10
6	0	$\rightarrow$	10
6	6	$\rightarrow$	16

The numbers in the table represent spike times, say in milliseconds. 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 use the feed-forward network architecture described in section 2.2. The connections have a delay interval of 15 ms; hence the available synaptic delays are from 1 to 16 ms. The PSP is defined by an α function as in (2.3) with a decay time $\tau = 7$ ms. Larger values up to at least 15 ms result in similar learning (see section 3.3.1).

The network was composed of three input neurons (2 coding neurons 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. With this configuration, the network reliably learned the XOR pattern within 250 cycles with $\eta = 0.01$. In order to "learn" XOR, 16 x 3 x 5 + 16 x 5 x $1 = 320$ individual weights had to be adjusted.

While using 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 using only 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 3.3.1). We remark that the introduction of inhibitory and excitatory neurons is not a limitation: 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 $f(x_1, x_2) : [0,1]^2 \rightarrow [0,1]$, like in (Maass, 1999). We translate this function to spike

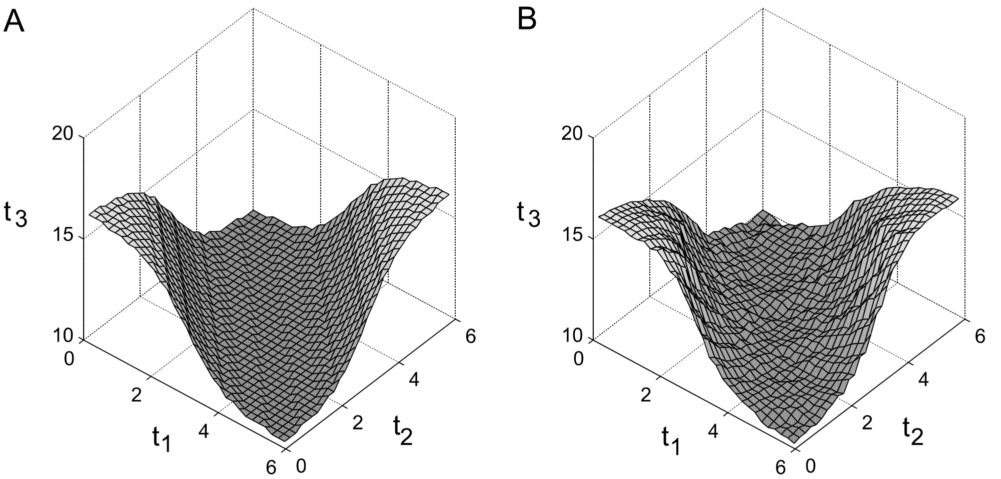


Figure 3.2: Interpolated XOR function $f(t_1, t_2) : [0, 6]^2 \rightarrow [10, 16]$. A) Target function. B) Network output after training. The network reached the sum squared error-criterion of 50.0 after learning 12996 randomly drawn examples from the 961 data-points.

times $f(t_1, t_2) : [0, 6]^2 \rightarrow t_3 : [10, 16]$, with times t_1, t_2 and t_3 in milliseconds (figure 3.2A). Using 3 input, 5 hidden and 1 output neurons, 961 input-output pairs of this function were presented to the network. The result of learning with *Spikeprop* is shown in figure 3.2B. The network can learn the presented input with an 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).

3.3.1 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 3.2, 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 3.3, where the average number of learning iterations required for the XOR function are plotted for a number of time-constants τ.

In figure 3.4, 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

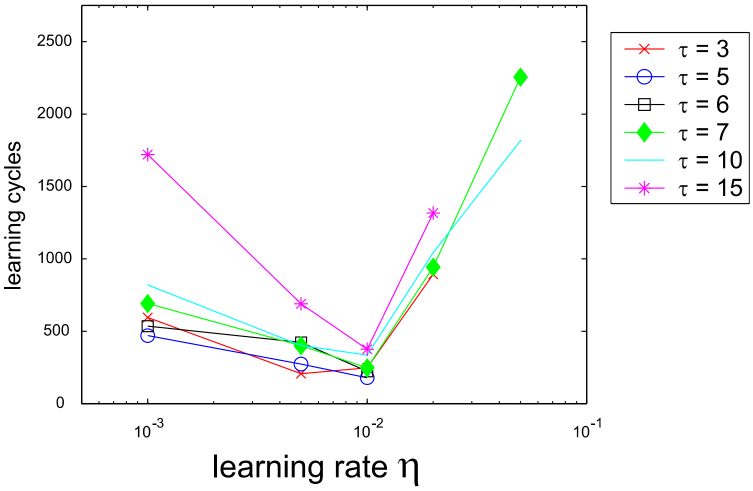


Figure 3.3: Learning XOR: Average number of required learning iterations to reach the sum squared error target (SSE) of 1.0. The average was calculated for those runs that converged.

(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 3.5A-B and show the same pattern. These results confirm the results as obtained by Maass (1996), where it was shown that theoretically the time-constant τ needs to be larger than the relevant coding interval ΔT. This observation can also be made for the results presented in section 3.4: a substantial speedup and somewhat better results were obtained if the time constant τ was slightly larger than the coding interval.

3.4 Other Benchmark Problems

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

The datasets are encoded into temporal spike-time patterns by population coding by the type of population coding we outlined in chapter 2, section 2.3. We independently encode the respective input variables: each input-dimension is encoded by an array of 1-dimensional receptive fields.

Output classification was encoded according to a winner-take-all

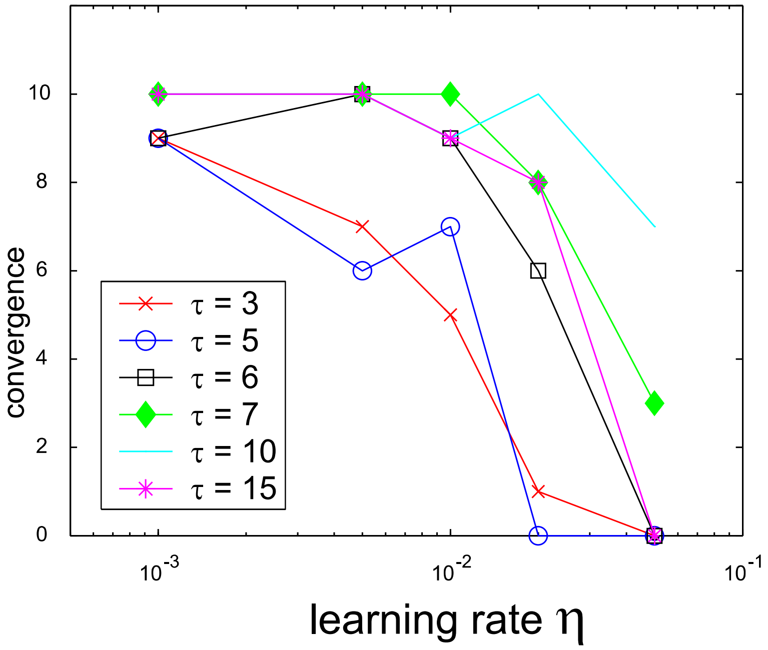


Figure 3.4: Learning XOR: Number of runs out of 10 that converged.

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 coding interval ΔT. A classification was deemed to be 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 was performed based on the current and previous membrane-potentials.

We tested our framework on several benchmark problems in which this temporal encoding 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 3.1. We also obtained results on the same dataset from a sigmoidal neural network as implemented in Matlab v5.3, using the default training method (Levenberg-

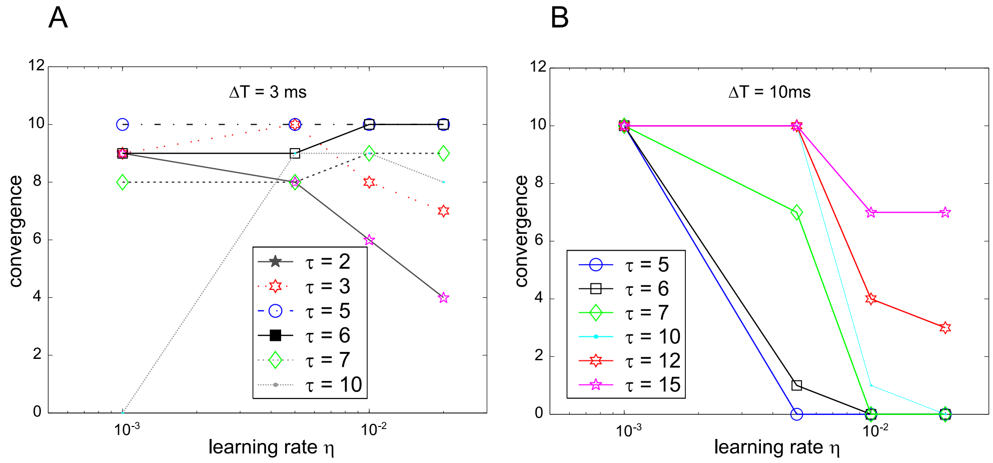


Figure 3.5: Number of runs out of 10 that converged for different values of τ. A) For a coding interval $t = 0 \ldots 3$, $[0, 3]^2 \rightarrow [7, 10]$. Target was an SSE of 0.7. B) For a coding interval $t = 0 \ldots 10$, $[0, 10]^2 \rightarrow [15, 25]$. Target was an SSE of 3.0.

Marquardt, LM) and simple gradient descent (BP). The input presented to both networks is preprocessed in the same way, so both methods can be compared directly.

Wisconsin Breast Cancer data-set

The breast cancer diagnosis problem is described in (Wolberg, 1991). The data is from the University of Wisconsin Hospitals and contains 699 case entries, divided into 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 3.1. The results as we obtained them are on par with values reported in literature for BP learning on this dataset: for instance, in a benchmark study by Roy, Govil, and Miranda (1995) 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

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 (Michie, Spiegelhalter, & Taylor, 1994). This dataset consists the Landsat dataset as described in the StatLog survey of machine learning algorithms (Michie, Spiegelhalter, & Taylor, 1994). 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 3.1. The results obtained by the Statlog survey are summarized in table 3.2.

Again, the results with *SpikeProp* are similar to the Matlab LM results, though even for twice the number of iterations, the Matlab LM results are somewhat worse. Compared to the best machine learning methods as reported in (Michie et al., 1994), 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 the 3x3 pixels in the environment. Note though that in (Michie et al., 1994) the Multi-Layer-Perceptron BP results lag other ML methods for this particular classification problem. After extended learning however, the Matlab LM training-method produces results that are significantly better than the reported values in (Michie et al., 1994) for MLP-BP learning. Such extended learning was not feasible for the SpikeProp algorithm due to time-constraints, although the results show increasing classification accuracy with extended learning.

From the results, we conclude that the application of the *SpikeProp* algorithm on temporally encoded versions of benchmark problems yields similar results compared to those obtained from the Levenberg-Marquardt (LM) learning algorithm on sigmoidal neural networks. Typically, less learning epochs were required with *SpikeProp*, however, in a single epoch 16 times more weights were adjusted. The results reported for the default Matlab LM routine were comparable or better as compared to those reported in literature.

3.5 Discussion

We discuss two kinds of implications: the computational implications of the algorithm we presented, and some considerations regarding biological relevance.

Iris dataset						
					accuracy	
algorithm	in	hidden	out	iterations	training-set	test-set
SpikeProp	50	10	3	1,000	97.4% ± 0.1	96.1% ± 0.1
Matlab BP	50	10	3	$2.6\cdot 10^6$	98.2% ± 0.9	95.5% ± 2.0
Matlab LM	50	10	3	3,750	99.0% ± 0.1	95.7% ± 0.1

Wisconsin Breast Cancer dataset						
algorithm	in	hidden	out	iterations	training-set	testset
SpikeProp	64	15	2	1500	97.6% ± 0.2	97.0% ± 0.6
Matlab BP	64	15	2	$9.2\cdot 10^6$	98.1% ± 0.4	96.3% ± 0.6
Matlab LM	64	15	2	3,500	97.7% ± 0.3	96.7% ± 0.6

Statlog Landsat set						
algorithm	in	hidden	out	iterations	training-set	test-set
SpikeProp	101	25	6	60,000	87.0% ± 0.5	85.3% ± 0.3
Matlab LM	101	25	6	110,078	83.6% ± 1.3	82.0% ± 1.5
Matlab LM	101	25	6	1,108,750	91.2% ± 0.5	88.0% ± 0.1

Table 3.1: The Iris data set was split into two sets of 75 items each for cross-validation. The Matlab LM routine was run for 50 epochs, or 1500 iterations. The Wisconsin Breast Cancer dataset was split in two for cross-validation. For the Statlog Landsat, we trained for 25 epochs, or 100,000+ iterations as well as for 250 epochs, or 1,000,000+ iterations. All results are averaged over 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 SpikeProp 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" (LM) or "TRAINGD" (BP) 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 3.2: Benchmark results on Landsat database from literature. Results on the k-Nearest Neighbor algorithm are given because it is the best classification method in the Statlog survey.

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 demonstrate that in order to reliably learn patterns of temporal width ΔT, the time-constant used to define the PSP is important: the time-constant has to be larger than the pattern that is being learnt.

For machine learning purposes, the event nature of spikes allows for efficient implementations of spiking neural networks (Delorme et al., 1999; Mattia & Giudice, 2000), especially in the case of sparse activity as achieved by the channel encoding (chapter 2.3). 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 for Levenberg-Marquardt learning in Multi-Layer-Perceptrons (LM-MLP) on the same pre-processed datasets. We did not attempt much parameter tuning or convergence optimizations like adding a momentum-term. 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 experiments was quite large. Convergence in our network seemed to be more reliable compared to the Matlab experiments: in experiments with spiking neural networks on the real-world datasets, the *SpikeProp* algorithm always converged, whereas the comparative LM-MLP experiments occasionally failed (<10%). Research into 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 depending on this feature has been used 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 (Pieri, 2001).

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 (Kuwabara & Suga, 1993), the barn owl (Carr & Konishi, 1990) and the electric fish (Heiligenberg, 1991)). As noted in chapter 7, a respectable amount of neurophysiological evidence now suggests that such neuronal activation with millisecond precision most likely transmits important information.

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 (1996) already refuted this notion (as also argued in (Diesmann et al., 1999)), but the results presented here demonstrate networks of spiking neurons with biologically plausible (relatively long) time constants capable of performing complex non-linear classification tasks. This was feasible for spike times encompassing a temporal coding interval 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.

A temporal code for transmitting information between neurons is especially interesting as it has been shown theoretically that it is very efficient in terms of spiking neurons required in the case of fast processing of information (Maass, 1999). Due to the all-or-nothing 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 (Maass, 1999; Shadlen & Newsome, 1994). It has also been reported that pooling the activity of more than 100 neurons does not increase accuracy due to correlated input noise (Shadlen & Newsome, 1998).

With our supervised learning algorithm, we demonstrated that efficient spiking neural networks based on temporal coding can be build and trained in the same way as traditional sigmoidal MLP networks. We have demonstrated a spiking neural network trained to approximate XOR that requires an order of magnitude less spiking neurons than networks based on instantaneous rate-coding, albeit with less robustness, as less neurons are involved. However, as neurons may come at considerable expense, this may be more desirable in many situations. We note however that other representational schemes may be devised that are equally efficient under these circumstances.

3.6 Conclusion

In this chapter, 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.

4

A FRAMEWORK FOR POSITION-INVARIANT DETECTION OF FEATURE-CONJUNCTIONS

ABSTRACT The design of neural networks that are able to efficiently encode and detect conjunctions of features is an important open challenge that is also referred to as "the binding-problem". In this chapter, we propose a framework for the efficient position-invariant detection of such feature conjunctions. For features placed on an input grid, the framework requires a constant number of neurons for detecting a conjunction of features, irrespective of the size of the input grid (retina). We implement the framework in a feedforward spiking neural network, and in an experiment, we demonstrate how the implementation is able to correctly detect up to four simultaneously present feature-conjunctions.

4.1 Introduction

The representation of structured information in neural networks has so far remained elusive at best, though it is thought to be required for efficiently solving a number of notoriously hard problems (Minsky & Papert, 1969; von der Malsburg, 1999). In a linguistic sentence like *The red apple and the green pear*, grammar implies the structuring of elements "red", "green", "apple", and "pear" into semantic composites, e.g. structure denoted with brackets: $\{\{red,apple\}, \{green,pear\}\}$. The *binding-problem* refers to the problem of how to encode and detect such structured representations in neural networks. We can easily identify elements like *red*, *green*, *apple*, and *pear* each with a neuron that is activated when the element is used. However, the embodiment of the structural brackets has been much debated, as far back as Hebb (1949). Some have even argued that such structural representation is impossible in neural networks (Fodor & Pylyshyn, 1988).

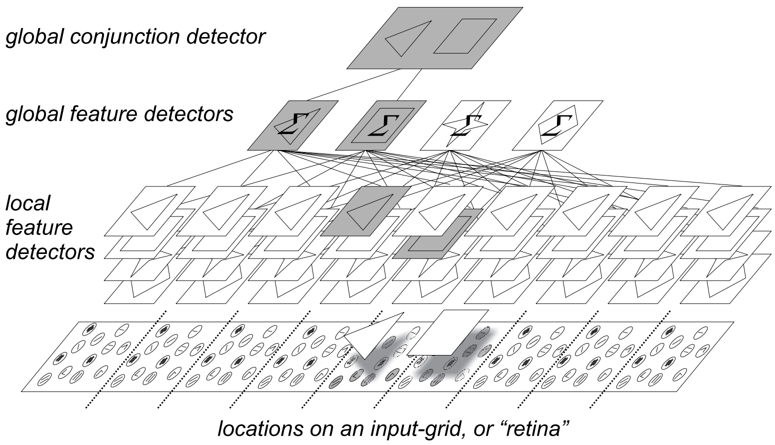

global conjunction detector
global feature detectors
local feature detectors
locations on an input-grid, or "retina"

Figure 4.1: Position-invariant conjunction detection via aggregation of activity in local feature detectors.

The classical binding-problem was originally posed in the context of visual perception, e.g. Rosenblatt (1961). Here the main concern is how to efficiently detect feature conjunctions on a retina, such as *red* and *apple*. Importantly, this conjunction of features can essentially appear anywhere on an input-grid (retina). Creating a *red apple* detector for every location on the retina seems too expensive, at least for every sensible conjunction of features (von der Malsburg, 1999). The straight-forward solution would seem to first create position-invariant *apple* and *red* detectors by combining the responses of the respective local detectors. These position-invariant detectors thus respond to the feature irrespective of its location. The conjunction of *red* and *apple* can then be gleaned from the co-activation of these position-invariant detectors to efficiently detect the *red apple*. However, this architecture is prone to errors in the presence of multiple conjunctions – red apple *and* green pear – since there are no structuring "brackets" present in the encoding by neural activation (von der Malsburg, 1999): the implicit links between *red* and *apple*, and *green* and *pear* are not represented, and the representation is ambiguous in the sense that the same detectors are activated in the presence of a *green apple* and a *red pear*.

The same perceptual binding-problem occurs when binding shape-primitives into compositional shapes based on relative position, e.g. triangle-on-top-of-square → "house-shape". Without "brackets" that signal the relative positions, the presence of multiple shapes in different places might lead to the incorrect detection of composites in position-invariant detectors. For example, the presence of a "triangle-star" *and* a "diamond-square" conjunction on the grid of figure 4.1 would activate the position-invariant "triangle" and "square" neurons, and would subsequently *wrongly* activate the position-invariant "triangle-next-to-square" neuron. ("ghosting"). The loss of local structure information in position-invariant detectors (the "brackets") is also referred to as the "superposition catastrophe".

Recently, progress has been reported on structured representation of symbolic structures in neural networks (Plate, 1995; Kanerva, 1996; Rachkovskij & Kussul, 2001). Such structured representations use *vectors* of binary neural activity as the primary data-structure, and binding is achieved via manipulation of these vectors to signify structure. To apply these results to the perceptual binding-problem, we need to solve the specific problem of position-invariant detection of feature conjunctions. Additionally, we remark that the solution should work in a feed-forward neural network, as the (human) detection of whole objects (e.g. *red apple*) is a very fast process (Thorpe et al., 1996), suggesting that the combination of local features into wholes, e.g. {*red*},{*apple*} → {*red apple*}, can be achieved in a feed-forward type network.

In this chapter, we propose a framework that addresses these issues. It enables efficient position-invariant conjunction detection in a feed-forward neural architecture. We then outline an implementation of the framework in spiking neural networks. We use spiking neural networks, since we explicitly exploit particular properties of these neurons, such as the ability to act as a comparator, or "coincidence detector".

The key idea we present is to separate the detection of local feature-conjunctions into two parts: we locally detect the presence of *a* feature-conjunction in a local *universal* conjunction-detector, where these conjunction-detectors do not identify the features, but responds to the presence of *any* conjunction. A local universal conjunction-detector uses a fixed procedure to encode the local conjunction in its output vector. The vector output of all the local universal conjunction detectors is then aggregated to yield a position-invariant *universal* conjunction detector. The correct – position-invariant – identification of the feature-conjunction is then possible from the position-invariant universal conjunction detectors in a *specific* conjunction detector – through the structure that was encoded locally in the vector outputs, and that survives aggregation. In experiments, we show that the framework can detect conjunctions of features, also in the presence of multiple other conjunctions.

The superposition of the output-vectors of local detectors in position-invariant detectors is aided by the use of spiking neural networks, that we argue are more suitable for this task than traditional sigmoidal neurons: a spiking neuron that receives single timed spikes from n input locations can superimpose these n inputs by emitting n timed spikes (unless some spikes occur simultaneously). Thus, in principle all n values are preserved, whereas a sigmoidal neuron would squash the n values into a single output value. We use this property in combination with a local procedure for encoding the (local) presence of conjunction of two features, like *red* and *apple*, or *green* and *pear*.

Rachkovskij and Kussul (2001) describe a procedure for encoding feature-binding via *Context Dependent Thinning* (CDT) operating on vectors of neural activity. We design a feed-forward CDT procedure for vectors of timed-spikes via *conditional shunting*. This procedure is implemented in local universal conjunction-detectors that locally encode feature-binding. In the architecture, the local presence of a feature is presented as input to the system as a vector of timed-spikes. The detectors process these vectors as the neural data-structure. The local universal conjunction detector receives two such vectors as input. It generates an output-vector via the CDT-procedure, if the input vectors indicate the presence of *any* local feature-conjunction (without identifying the actual features). In effect, the local CDT-procedure "binds" the two conjunctive features together (the "brackets" considered earlier). The *specific* contents of the feature-conjunctions are decoded at a global, or position-invariant level in specialized feature-conjunction detectors.

We demonstrate our architecture in an example that binds features based on relative proximity, as on the grid of fig. 4.1. In this architecture, a position-invariant detector for the conjunction of say {triangle,square} consists of some N neurons, a value independent of the number of input locations. With such position-invariant detectors, we can detect up to about 4 or 5 *similar* conjunctions simultaneously. We note that visual processing seems to be limited in the same way (Luck & Vogel, 1997).

This chapter is organized as follows: we outline the architecture in section 4.2. The implementation of this framework in networks of spiking neurons is given in section 4.3, and the detection of conjunctions is demonstrated in section 4.4. We discuss and conclude the architecture in sections 4.5 and 4.6. A formal definition of the framework is developed in chapter 5.

4.2 Local Computation with Distributed Encodings

In this section, we outline a feedforward architecture for the global detection of feature-conjunctions, and present the idea of *local computation with distributed encodings*: local network nodes process vectors of neural activity. The local information is thus distributed over the elements of a vector: a local distributed encoding. The proposed architecture is implemented in spiking neural networks in section 4.3.

4.2.1 Architecture. We propose an architecture as shown in figure 4.2. We introduce two local universal conjunction-detectors, denoted $(X|Y)_R$ and $(Y|X)_L$, in addition to the local feature-detectors, denoted A, B, C, etc. . . The local conjunction-detectors detect and encode the presence of a conjunction of *any* two features. In our example, we consider the binding of shape-*right-next-to*-shape; the same framework can be applied to binding say color-to-shape. The signals of the local detectors are aggregated in respective global feature and conjunction detectors, denoted ΣA, ΣB, etc..., and $\Sigma(X|Y)_R$, $\Sigma(Y|X)_L$. The presence of particular feature-conjunctions is then detected from the combined information of the global universal conjunction and feature-detectors in dedicated, global detectors (ΣAB, ΣCA, ΣBA etc..). As we will show, the vector nature of the neural activity processed in these detectors enables the detection of the correct feature-conjunctions, also in the presence of multiple other conjunctions.

The local detection of features can easily be considered in terms of activity-vectors. We assume that all (discrete) locations on an input-grid are populated with identical sets of diversely tuned basic neurons (e.g. grid in fig. 4.1). The presence of a feature like A is then characterized in distributed fashion by the activity (spikes) it elicits in such a set of some N basic neurons. The timings of the spikes of the neurons for each set are collected in a vector, where each vector-element contains the activity of one neuron. The detectors in the proposed architecture process such *spike-time* vectors.

At the level of local detectors, we have local feature-detectors that look for a specific feature, say A, B, C etc. . .. Each such detector looks at one set of basic neurons. If it detects that the local input vector sufficiently matches

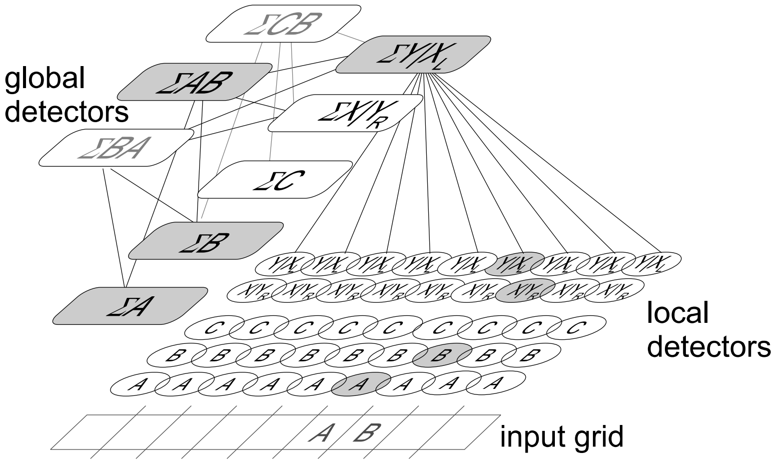


Figure 4.2: Architecture for global detection of conjunctions.

the preferred vector, it propagates the input-vector, with some delay due to computation: e.g., only if presented with input A, a local A-detector outputs A.

We also have local detectors that detect and signal the conjunction of *any* two features. These detectors consider two sets of basic neurons. The idea is that they detect the presence of features in both locations by only considering the actual *amount* of activity. In our example, these nodes look at two locations next to each other in the grid. We have complementary right-facing and left-facing detectors $(X|Y)_R$ and $(Y|X)_L$. In the presence of say A-next-to-B, these detectors respectively output vectors $\mathrm{A}\backslash b$ and $\mathrm{B}\backslash a$, vectors that each look like A respectively B (we define this in section 4.2.4).

The next level in the architecture combines the results of the local detectors. Here we exploit a specific property of spiking neurons: suppose we have two neurons each emitting a spike-train containing k spikes. These two spike-trains can be combined into one spike-train which then contains $2k$ spikes (if the spikes all have different times).

The vectors from the respective local detectors are combined to the output-vectors of global feature detectors (“there is a triangle”) and global conjunction-presence detectors (“there are two active consecutive locations”). In the output-vector of a global detector, an element contains a spike-train obtained from the concatenation of the spike-times of (active) spikes in the corresponding elements from the local detector-vectors (an element i in the global vector contains the timed spikes from all elements

i in the local vectors). Thus, we can obtain global aggregate vectors by combining the local vectors, where the use of spiking neurons alleviates the "superposition catastrophe" encountered with sigmoidal neurons (von der Malsburg, 1999).

Finally, the activity vectors from the global detectors are used to detect the presence of *specific* consecutive features in a global feature-conjunction detector. The detection of the specific-features next-to-each-other from the global detectors is possible, because at the local level, we make use of a special procedure: the output vector of a local universal conjunction-detector resembles the vector associated with one of the two features, but this vector is "watermarked" with the vector associated with the other feature. This "watermarking" entails the removal of some spikes in one feature-vector due to the presence of the other feature-vector. The detector and "watermarking"-details are given below, the idea of global conjunction detection via (conditional) vector-propagation is depicted in figure 4.3, with detector outputs denoted as vectors.

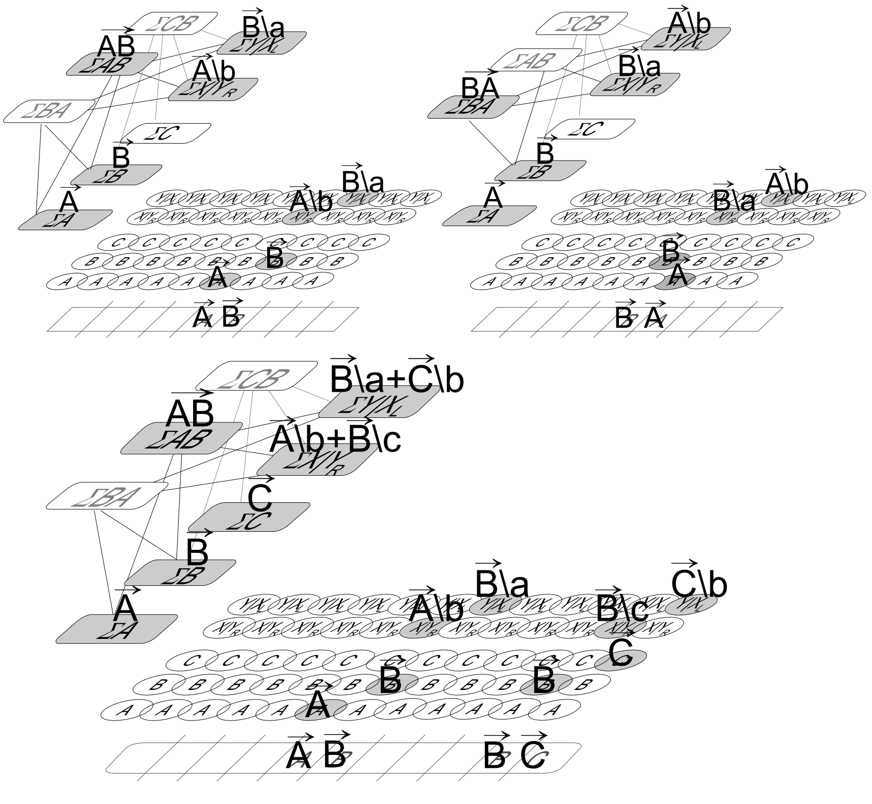


Figure 4.3: Vector propagation in a vector-based architecture: correct global conjunction detection.

4.2.2 Neural data-structure. The presence of feature like *triangle* is characterized by the distributed activity vector that its presence elicits in the local set of basic neurons. We let these basic neurons each emit at most one, precisely timed spike. The collected spikes of N neurons then yield a *spike-time vector*: $\mathrm{S} =< t_1, t_2, \ldots, t_n >$, with t_i the time of the spike emitted by neuron i. Should a neuron emit multiple spikes, then the spike-time vector generalizes to a *spike-train vector*: $\mathrm{S(t)} =< \mathrm{t_1, t_2, \ldots, t_n} >$, where $\mathrm{t_i}$ is a vector of spike-times. Detectors in the architecture operate on these spike-train vectors: this is the neural data-structure of the network.

4.2.3 Local Feature Detection. A local feature detector like A in figure 4.2 detects the local presence of a feature A. It receives the local activity vector S, and, if input vector sufficiently corresponds to the activity vector A associated with the presence of a feature A, the local activity vector is propagated, albeit with some delay due to computation. Otherwise, no activity is propagated.

4.2.4 Local Feature Binding. The local universal conjunction detectors $(X|Y)_R$ and $(Y|X)_L$ in figure 4.2 perform local universal feature binding, and are the first step in enabling *correct* global detection of feature-conjunctions. These detectors detect and signal "there are two active locations next to each other". To signal the local conjunction, we adapt the idea of *Context Dependent Thinning*(CDT) as in (Rachkovskij & Kussul, 2001). In (Rachkovskij & Kussul, 2001), it is observed that the binding of one vector, say A, and another vector, say B, can be signaled by setting part of the active elements ("1's") in the vector A to inactive ("0's"), as a function of B. This *contextually thinned* vector, denoted by $\mathrm{A}\backslash b$ is then indicative for the AB conjunction.

We design a feed-forward CDT procedure using spiking neurons based on *shunting inhibition*, e.g. (Thorpe & Gautrais, 1997). A local universal conjunction-detector $(X|Y)^i_R$ receives as input two spike-time vectors, in our example the spike-time vectors from two consecutive locations, i and $i+1$. We denote these spike-time vectors with X and Y respectively. The detector determines whether there are sufficient spikes present in X and Y to assume the presence of two features (a conjunction). In that case, it propagates X, with part of X *shunted* by Y. Shunting is defined as follows: a spike in an element j of Y inhibits the propagation of *later* spikes in a set Γ^i_j of elements in X, where Γ^i_j is fixed via inhibitory connections. With inputs $\mathrm{X} =< t^x_1, \ldots, t^x_n >$ and $\mathrm{Y} =< t^y_1, \ldots, t^y_n >$, the spike in t^x_i is propagated if not shunted, i.e. if $\forall k \in \Gamma^i_j : t^x_i < t^y_k$. The complementary detector $(Y|X)^i_L$ shunts Y with X.

Importantly, different thinned spike-time vectors can be superimposed without losing the different vector-patterns, thus alleviating the superposition-catastrophe (up to some point). For two vectors containing (sparse) random spikes, first half the spikes in each vector are removed, and then the two shunted vectors are superimposed. In the new vector, say Σ, an aggregate element Σi will thus contain all spikes from local elements i, but most of these local elements will be empty. In the rare case that more than one spike has to be superimposed, it is very unlikely that these spikes will occur simultaneously, and hence the corresponding aggregate element will simply contain both (all) these spikes.

This idea also works in the worst case, when the conjunctions are similar in at least one component: say in one location, a vector like A is shunted by a vector B ($A\backslash b$), and in another location, a vector like A is shunted by a vector C: $A\backslash c$. When these two shunted versions of an A-vector are superimposed, the following happens: part of the two input vectors are the same; a part of the original "A"-activity is present only in $A\backslash b$; and another part only in $A\backslash c$. Hence, a part of the removed spikes in $A\backslash b$ is also not present in $A\backslash c$, and these absent spikes are specific for the combination of vectors $A\backslash b$ and $A\backslash c$. Note that increasingly adding more "similar" conjunctions ($A\backslash d$, $A\backslash e$, etc...) will ultimately fill in all the removed spikes. Since a conjunction is signaled by two complementary parts aggregated in different global detectors (e.g. AB by $A\backslash b$ *and* $B\backslash a$), this limit is only reached when similar complementary conjunctions are provided. The capacity for simultaneous representation is then some 4–5 *similar* conjunctions (an example of such a situation is depicted – and tested – in section 4.4).

4.2.5 Conjunction detection. A global conjunction detector ΣAB for A-left-next-to-B (AB) consists of an input-layer for detecting correspondence of the input to the conjunction AB, and an output-layer that propagates the activity in the input-layer if this activity is larger than some threshold (fig 4.4, dark detector). The input-layer is set to consist of N ordered elements, corresponding to the length of the spike-time vector. Input elements are exclusively connected to the corresponding elements in either the ΣA and $\Sigma X|Y_R$ detector, or to ΣB and $\Sigma(Y|X)_L$. A connection to a pair is made based on the following. When the architecture is presented with AB, elements are activated in ΣA, ΣB, $\Sigma(X|Y)_L$ and $\Sigma(Y|X)_R$. The output vectors of ΣA and ΣB then correspond to respectively A and B (when A signals feature A, same for B); due to shunting by B, $\Sigma(X|Y)_R$ contains a particular fraction of the vector A, and vice versa $\Sigma(Y|X)_L$ a particular fraction of B. Thus, *for these fractions*, corresponding elements i fire coincidently in the output vector of both ΣA and $\Sigma(X|Y)_R$, or ΣB and $\Sigma(Y|X)_L$. An element i in the input layer of the ΣAB detector is connected to elements i in a Σ-detector pair, if the pulses from these elements i would be coincidental when presented with AB. The threshold for the ΣAB input-elements is set to two coincident pulses, the threshold of the ΣAB output-elements equals the number of elements that are activated in this input-layer if the conjunction AB is presented. An example of such connectivity is depicted in figure 4.4A,B. When an active vector contains n active elements/spikes, the presence of an AB-conjunction activates $n/2$ elements in $\Sigma X|Y_R$ and in $\Sigma Y|X_L$, which then activate n elements in the input-layer of AB (e.g. 4.4A). This n-element pattern is then propagated by the output-layer. For a BA conjunction however, the active elements in the global universal conjunction-detectors are interchanged, and no input-elements in ΣAB receive coincidental spikes (fig 4.4B).

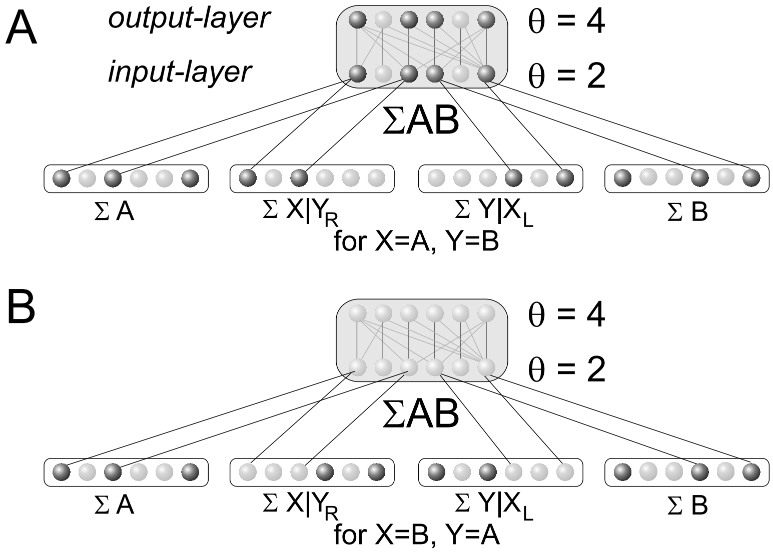


Figure 4.4: A) Connectivity for detecting global feature conjunctions. Dark elements are active. B) Active elements in same global conjunction-detector for interchanged feature positions.

With the conjunction AB present, the presence of additional conjunctions on the input-grid yields additional spikes in the global conjunction-detectors, but these added spikes do not disturb the AB spikes that remain present. With the correct features present in non-related conjunctions ("ghosting"), the spike-patterns in the global conjunction-detectors only partially match the AB pattern, and less input elements are activated in the input layer of AB: hence the AB output-elements do not reach threshold and the conjunction-detector is not turned "on".

These conjunction-detectors can correctly detect the conjunction in the presence of up to some M other, similar conjunctions on the grid: the presence of conjunctions AC, AD, AE, etc..., gives rise to the additional superposition of local vectors $\mathrm{A}\backslash c$, $\mathrm{A}\backslash d$, and $\mathrm{A}\backslash e$ in $\Sigma(X|Y)_L$, which will increasingly "fill in" the locally shunted spikes in $\mathrm{A}\backslash b$ (and similarly for $\mathrm{B}\backslash a$ in $\Sigma(Y|X)_R$). The value of M depends on the amount of shunting, i.e. section 4.4.

Spiking neurons can learn this connectivity via temporal Hebbian learning rules as in (Natschläger & Ruf, 1998): presented with single AB conjunction, such a learning rule would translate the synchronous activation of neurons in either of the detector pairs into selectively enhanced weights.

4.3 Implementation

In this section, we outline the implementation of the neural detectors outlined in section 4.2 in networks of spiking neurons. Previous research has demonstrated that these neurons operating on timed spikes can perform a number of required tasks, such as pattern detection (Hopfield, 1995; Natschläger & Ruf, 1998; Bohte, Kok, & La Poutré, 2002c; Bohte et al., 2002b; Thorpe & Gautrais, 1997; Ruf & Schmitt, 1998), and an effective means of comparing simultaneously active inputs through coincidence detection. The spiking neurons we use are leaky-integrate-and-fire neurons modeled as Spike Response Neurons (Maass & Bishop, 1999). These neurons sum incoming spikes as post-synaptic potentials (PSPs) to calculate an internal variable called "membrane potential". When this potential reaches a threshold θ, a spike is generated and a refractory (negative) response is added to the potential. The PSP's are parameterized by the decay constant τ and is set to 7ms, unless stated otherwise (a detailed description of spiking neural networks is given in chapter 2, section 2.2).

In our setup, each location on the input grid is populated with N diversely tuned basic neurons. We denote a basic neuron j by S_j. When presented with a local feature, some $n \leq N$ basic neurons emit a spike, generating a spike-time vector S with elements s_j.

A local feature-detector A aims to detect the presence of a spike-time vector A in the input vector S, and, conditional on this presence, then transmits input vector S. The feature detector consists of N spiking neurons. Each detector neuron, A_i, receives equally weighted input from all elements in the input vector S that are active if S would correspond to the desired vector A. The connection between neuron A_i and the input-elements s_j have a delay such that when an input vector S equals vector A, all input spikes arrive at the detector-neuron simultaneously, e.g. for an input element s_j that would fire at time a_j if vector A is present, the delay between input neuron s_j and detector neuron A_i, d_{ij}, is set such that $a_j + d_{ij} = c_i$, with c_i constant for all connections from any input neuron s_j to detector neuron A_i. This effectively detects the specific temporal pattern that makes up vector A, e.g. (Natschläger & Ruf, 1998). The constant c_i is set to $c_i = c + a_i$, with c some constant. The result is that an input vector resembling A is effectively propagated (fig. 4.7A). Additionally, the A detector responds in a graded manner to "A-ish" vectors, as increasingly different vectors are propagated with increasing delay and decreasing activity (see also (Natschläger & Ruf, 1998; Bohte et al., 2002c)).

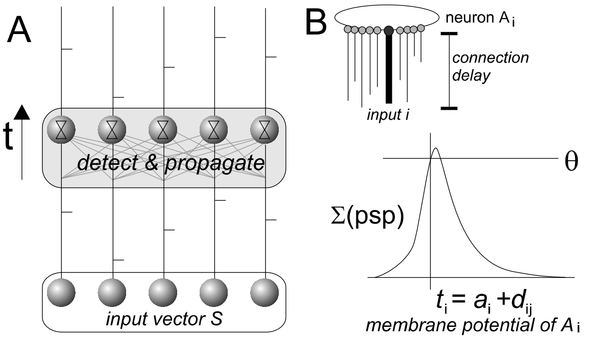


Figure 4.5: A)Local feature detector (dark box): each sphere is a spiking neuron, dark neurons are active. Horizontal ticks are timed spikes. B) Top: weights and delays of connections to input. The different lengths of the connections to the neuron depict the delay of the connection: tuning the delays allows a neuron to be sensitive to only a specific temporal pattern in the input. Bottom: time-course of membrane-potential for preferred feature.

A local universal conjunction-detector $(X|Y)$ (fig. 4.6A) receives input from two consecutive locations. The respective sets of basic neurons are denoted by S and T, with activity vectors S and T and neurons S_i and T_i. The detector has to detect the presence of features in both locations, i.e. at least some n spikes in S, and at least some n spikes in S. If this condition is met, part of the spike-time vector that is S has to be propagated as output for downstream detectors. Which part is propagated is determined by the shunting operation where the presence of the vector T inhibits part of the vector S from being propagated.

A local universal detector, say $(X|Y)_L$, receiving input from consecutive locations S and T, consists of N spiking neurons $(X|Y)_i$, where each neuron is connected to all basic neurons in S and T. To (selectively) propagate a timed spike from S_i in detector neuron $(X|Y)_i$, this neuron first has to establish that sufficient activity is present in both input locations S and T. This can be done by setting the weights to all input elements from S and T such that if both input locations contain sufficient spikes, these "background" spikes generate at least a potential ϑ_b for some duration of time. The value of ϑ_b is such that the PSP evoked by the additional presence of a spike in S_i pushes the membrane potential of $(X|Y)_i$ through threshold and generates a spike. To implement CDT, the weights from shunting inputs $T_j \in \Gamma_j^i$ are set such that they strongly inhibit $(X|Y)_i$, effectively shunting $(X|Y)_i$ if spikes from these elements $T_j \in \Gamma_j^i$ precede a spike from S_i (fig. 4.6B, bottom).

To propagate the temporal structure in the input vector S, we arrange the connections from the "background" inputs such that for sufficient activity in both locations, the resulting membrane potential from these inputs in an element $(X|Y)_i$ first steeply rises, and then remains flat some time before falling off again. The idea is that if a (strongly weighted) input spike from S_i is added to this elevated flat potential, it causes a spike $(X|Y)_i$ some fixed Δt after s_i. Thus, the timing of a spike in S_i only incurs a fixed delay. In our experiments, the flat effective membrane potential is achieved by modifying the shape of the PSPs evoked by the background neurons to this shape (this can be thought of as to model a wide distribution of weighted synaptic delays from these background neurons).

For such a construction, the input/output spike-times for a $(X|Y)_L$ detector for some randomly generated input vectors S and T are shown in figure 4.7B: most of the non-shunted part of the input spike-time vector is generated as output by the local conjunction-detector (with a fixed delay added to the original input vector). As can be seen however, late spikes are more likely to be shunted, and the relative spike times of early spikes are somewhat delayed due to the still increasing membrane potential early on.

The global feature-detectors and the global universal conjunction-detectors each consist of a single layer of N spiking neurons. One such

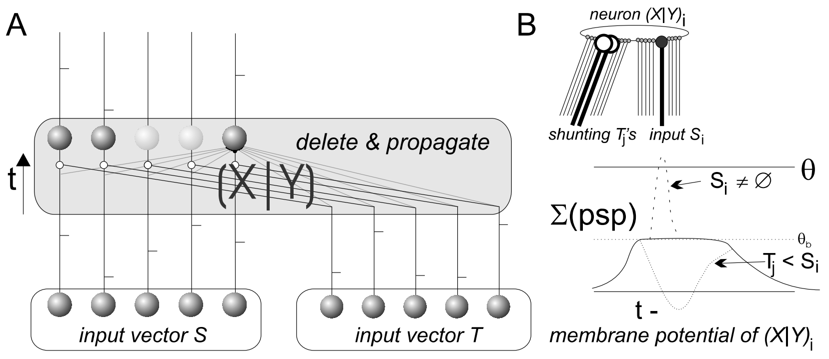


Figure 4.6: A)The local universal conjunction detector (dark box) deletes and propagates parts of input S. B) Top: weights and delays of connections to input. Open weights are inhibitory, solid weights are excitatory. Bottom: time-course of membrane-potential (the sum of all inputs, ΣPSP), given the presence of some two features. Background input spikes cause "flat" raised potential, in the absence of a spike from input s_i (denoted as $s_i = \varnothing$). An additional spike from input s_i (denoted as $s_i \neq \varnothing$) will generate a spike; earlier spikes from the set of shunting elements from T (denoted ($T_j < S_i$ evoke inhibitory PSP's, preventing a spike from s_i from propagating.

a neuron, Σ_i, is connected only to corresponding neurons i in the respective local detectors (fig. 4.8A), and its membrane time-constant τ is set to 4ms. The weights are such that the threshold is reached by a single spike (fig. 4.8B).

A global feature-*conjunction* detector consists of an input-layer of N spiking neurons, with τ set to 4ms, and an output-layer of N spiking neurons, with τ set to 7ms. The input neurons are connected to global feature and global universal conjunction detectors, as outlined in section 4.2, and fig. 4.4A. The neurons in the output layer are connected to all neurons in the input-layer, like a feature-detector, and detect the presence of the n active spikes.

4.4 Experiments

In this section, we show the ability of the proposed architecture, implemented as outlined in section 4.3, to correctly detection conjunctions of features. We tested the architecture with a conjunction-detector selective for the feature-conjunction *triangle*-next-to-*square*, i.e. our "*AB*"-conjunction. In the experiments, we placed a number of feature-

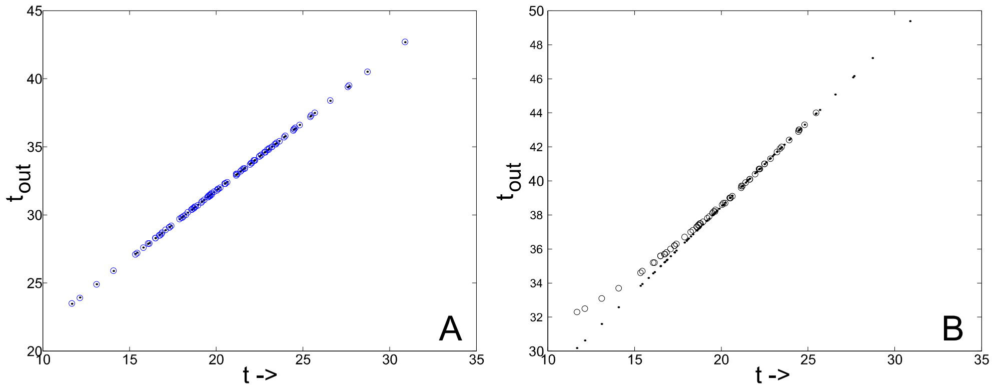


Figure 4.7: Vector propagation: input vs. output times. Circles: actual output, dots: delayed input, for: A) local feature detector A, presented with A, B) local universal conjunction detector, presented with some S and T.

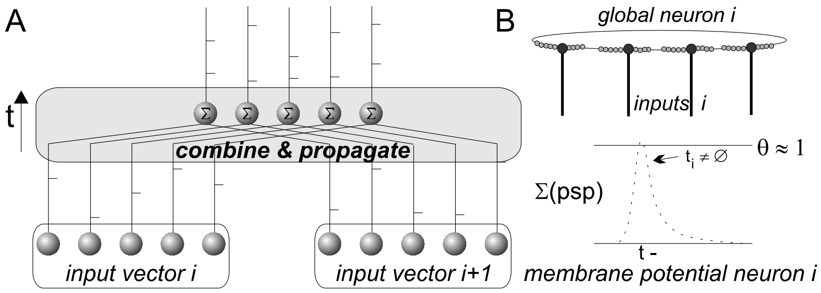


Figure 4.8: A) Dark box: global feature or universal conjunction detector. B) Top: weights and delays of connections to input vector. Bottom: time-course of membrane-potential for impinging spike.

conjunctions on a grid, and we measured the number of activated neurons in the input-layer of AB. The different feature-conjunctions placed on the grid are shown in figure 4.10. Scenes *(a)* and *(b)* reflect the uncluttered conjunction-detection problem. Scene *(c)* would cause "ghosting" without a special feature-binding operation, *(d)*, *(e)* and *(f)* test increasing feature-conjunction clutter without and with the target conjunction present.

In the experiments, the neural input vectors were of length $N = 500$, with $n = 100$ active (spiking) neurons. Each separate feature was defined in advance by a randomly drawn set of spike-times from a normal distribution, with $\sigma = 3.5$ms. In the local universal conjunction-detectors, approximately half the input spikes were shunted. The experiment was repeated 10 times and the average activations in the input-layer of AB are shown in figure 4.11A. As can be seen, there is a clear difference between i

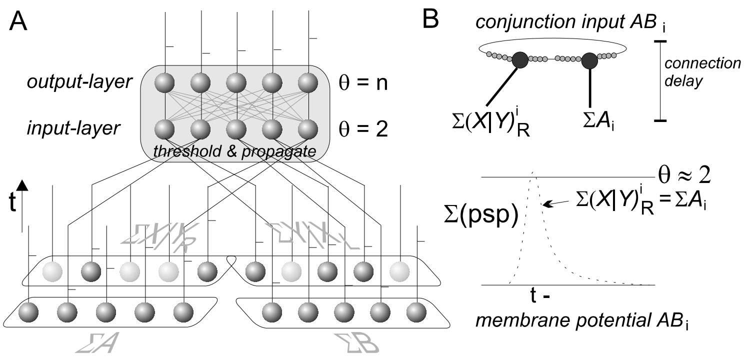


Figure 4.9: A) Global conjunction detecting node: dark spheres indicate active neurons, horizontal ticks the timed spikes. B) Top: weights and delays of connections to input vector. Bottom: membrane-potential due to two synchronous impinging spikes.

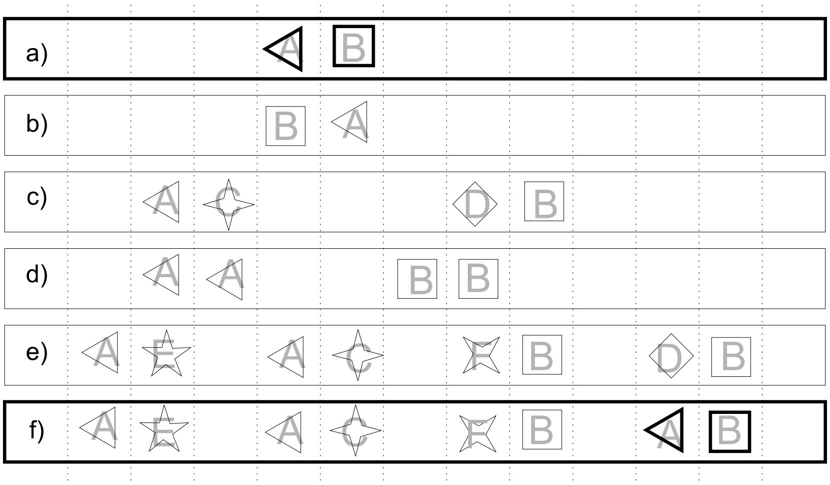
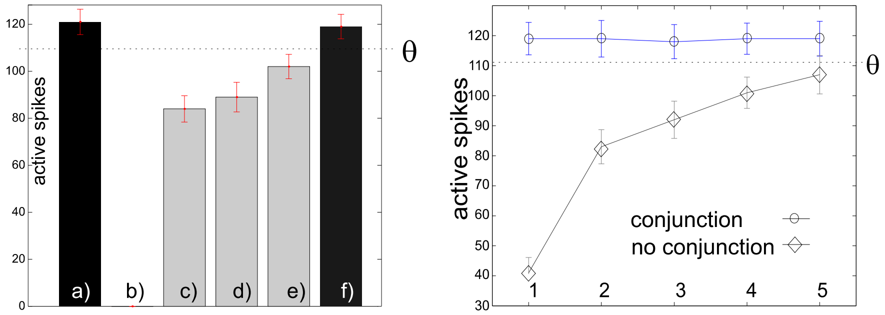


Figure 4.10: (a)-(f) Feature-conjunctions on a grid. (a) and (f) contain the sought-after *triangle-left-of-square* conjunction, the other scenes contain the features, as well as distracter conjunctions, but not the actual conjunction.

conjunction-scenes that do not contain the AB-conjunction, but merely its constituent features, and those that do contain the conjunction. The threshold θ for detecting AB was set to $\theta \approx n$, as determined from the uncluttered presentation of AB.

We performed a systematic comparison between configurations with and without AB conjunction. Plotted in figure 4.11B is the AB-activation for an increasing number of “distracter” conjunctions, with AB present (cir-



(A) (B)

Figure 4.11: A) Activation in AB detector. B) Activation as a function of scene clutter. Error-bars indicate min/max activation in 10 simulations.

cles), and absent (diamonds). Distracters are defined as AC, DB, AE, FB etc... The difference in activation in the conjunction detector becomes too small to reliably discriminate the two cases for more than 4 conjunctions. A higher "simultaneous representation capacity" can be obtained by increased local shunting, but at the expense of lowering activation in the conjunction detector as compared to the original input vectors. Instead, "attentional" mechanisms could determine a region of interest for a more reliable detection when presented with too many conjunctions.

At first sight, it might seem expensive to dedicate some 500 neurons to the detection of a single conjunction, especially since this detection is only reliable if no more than 4–5 similar conjunctions are present simultaneously on the input grid. However, from a computational perspective, under the restriction of a limitation in the perceptive abilities, the framework incurs a fixed neural "cost" per conjunction-detector, rather than a neural cost linear in the size of the input grid, as would be required for a complete solution where a conjunction detector is placed on every location of the grid. The complexity of the learning task also scales with the number of dedicated detectors (the 500 similar neurons in our position-invariant detector can all "learn" at the same time). From a biological perspective, we remark that the traditional artificial neuron, in the form of a sigmoidal neuron, implicitly models a group (100–1000) spiking neurons. The increased power of discrimination in our framework results from the added precision in the spike-timings of single neurons, and the specificity of the wiring patterns.

4.5 Discussion

In this paper, we proposed a network architecture that processes *activity vectors* rather than single *scalars* in its nodes. Implicitly, the scalar output of a sigmoidal neuron in traditional networks is taken to model the activity of a population of (biological) spiking neurons. The assumption is then made that the population signal can be described by the scalar value of its average *firing rate*. We model a more refined computation in the spiking neuron population by explicitly computing with vectors of spike-activity.

In the proposed framework, we take the activity of a population of spiking neurons as a (sparse) spike-time vector, and we interpret this activity as a distributed representation of the local input. Distributed representations have been extensively studied as a means of encoding and processing more information in a population of neurons as compared to single neurons, e.g. (Rumelhart, McClelland, & PDP Research Group, 1986; Abbott & Sejnowski, 1999). Distributed codes have also been shown to be remarkably robust and efficient, and suitable for associative memories (Kanerva, 1988).

We use the properties of distributed representations for essentially local computation: the local presence of an object generates local activity vectors that can be processed by local detectors. One important property of distributed representations that we exploit is the ability to encode compositional structure by combining vectors in a specific way (Kanerva, 1996; Plate, 1995; Rachkovskij & Kussul, 2001). In our local feature-binding detectors, the compositional structure of "somethingX-left/right-somethingY", is signaled by a fixed binding-encoding procedure that removes part of the spikes in "somethingX" given the vector "somethingY" in one detector, and vice versa in a complementary detector. We propose this specific method, as it allows for a straight-forward aggregation of local output into respective position-invariant global detectors. The locally imposed compositional structure – the encoded feature-binding – is then recoverable for a specific position-invariant feature-conjunction detector from position-invariant feature and universal conjunction detectors (up to some point). The use of local distributed representations thus alleviates the "superposition catastrophe".

The proposed local universal binding detectors perform a variant of dynamic variable binding (Browne & Sun, 1999): the variable "something-right-next-to-something" is equated to the vector that becomes its output. The specific vector itself is not identified at this stage, as the local variables are all "copied" to a global variable, where a global detector then performs the identification. Dynamic variable binding is an important ingredient for computing with symbolic structures with neural networks. This suggests that the ideas proposed in this paper should be extensible to symbolic-inference type of applications; for now however this remains "future work".

We chose the example of feature-binding of consecutive shapes on an input grid, to demonstrate how our framework allows a "natural" feed-forward network to perform feature-binding. For this particular example it is especially clear that an extension of the framework to include hierarchical compositional integration enables the efficient recognition of increasingly complex conjunctions in a global, or *position-invariant* manner. Although we have worked out possible solutions, the full implementation remains as future work. However, keeping this goal in mind, we used two complementary CDT-detectors, $(X|Y)_L$ and $(Y|X)_R)$, resulting in two "bound" vectors, e.g. $\mathrm{A} \setminus b$ and $\mathrm{B} \setminus a$. The combination of these two vectors preserves the initial (average) number of spikes in a feature-vector for processing downstream, that is, if CDT removes about half the active spikes. The intuition is, that if less spikes are removed, the feature-conjunction is signaled weaker; if more are removed, the total number of spikes that can be used downstream decreases. As noted, if CDT shunts about half the spikes, the number of *similar* conjunctions that can be detected simultaneously is about 4 (sections 4.2.5 and 4.4). We note that the human brain seems to perform similarly, i.e. Luck & Vogel (Luck & Vogel, 1997).

To detect more than 4 *similar* conjunctions simultaneously, say for further integration, a solution would be to use multiple copies of the same detector and assign these copies to a region of the input retina such that it is unlikely to encounter more than 4 similar conjunctions. The output of these "localized" detectors can then again be aggregated to achieve a truly position-invariant conjunction-detector. Importantly, this would only apply to detectors for features-conjunctions that are often present in numbers larger than the capacity of our framework. If seen as a biological model, the prediction then is that the density of feature-detectors sensitive to particular conjunctions is proportional to the probability of co-occurrence of multiple such conjunctions. As such, having to allocate and learn multiple detectors for often occurring conjunctions seems quite feasible.

The "synchrony hypothesis"(von der Malsburg, 1999; Singer & Gray, 1995) has so far been the main theory on dynamic feature-binding to allow for the simultaneous representation of multiple feature conjunctions. However, criticism with regard to the viability of this scheme has been mounting, e.g. (Shadlen & Movshon, 1999). In particular, if feature-bindings are to be signaled by the synchronous firing of neurons coding for parts of the same object, it is not clear how the correct synchronization is determined. This alone would seem to require solving the binding-problem first: the particular encoding of the solution then seems non-relevant.

The scheme we propose does explicitly answer the question of how to detect and encode a local feature-conjunction, and allows for the simultaneous detection of multiple feature-conjunctions. It has the advantages that it uses an inherently distributed code; enables feed-forward spatial feature-binding, and can be implemented in biologically reasonable spiking neural networks. We also remark that the vector-structure is more a formalization than a spatially localized necessity. The required connectivity only connects similarly tuned neurons from different locations to global neurons. The collective distributed activity of neurons thus connected can be interpreted as a spike-train vector data-structure.

We need to remark that within neuroscience, an alternative approach is also being advocated. In this approach, it is argued that only one object at a time is effectively being processed, and attentional mechanisms allow the visual system to switch between attended percepts. Then, only one object is processed at any one time, and it is argued that the binding-problem can thus be avoided (Roelfsema, 1998; Shadlen & Movshon, 1999; Mel & Fiser, 2000; O'Reilly & Busby, 2001). This view has the unfortunate side-effect that it does not explain the productivity of symbolic structural representations, such as those expressed by speech, whereas an integrative compositional vision system would go a long way. Some have noted that using distributed coding in principle allows n local neurons to locally detect an encode a large number (up to 2^n) of features (Pollack, 1990; Mel & Fiser, 2000; O'Reilly & Busby, 2001). With such a large encoding space, all relevant local feature conjunctions could be detected. This however still requires local learning for every local conjunction, a problem that position-invariant conjunction detection eliminates.

For an integrative compositional vision system, we believe our framework highlights the importance of overcoming the superposition catastrophe to enable structured representation in a position-invariant way. Many approaches to the binding-problem have focused on the issue of structured representation of symbolic information, e.g. (Pollack, 1990; Hinton, 1990; Smolensky, 1990; Plate, 1995). With the additional requirement in vision of achieving some sort of position-invariance, these theories do not seem to straightforwardly apply, as they describe procedures for signaling a local binding. For these procedures, it is unclear how multiple local bindings could be superimposed to "extract" the binding at a position-invariant level. We explicitly solve this problem by separating the detection of a specific feature-conjunction (e.g. *red apple*) into the local detectors for respectively: *apple*, *red*, and *a-color-and-a-shape*. The information encoded in the output of these detectors can be aggregated in respective position invariant *apple*, *red*, and *a-color-and-a-shape* detectors, from which a specialized, position-invariant *red apple* detector can correctly extract whether or not the particular conjunction is present.

In the experiments, we used biologically reasonable spiking neurons implemented in the Spike-Response Model (SRM) as defined in (Gerstner, 1995). These neurons were used for three reasons: firstly, their ability to emit multiple spikes to implement the global universal conjunction detectors. Secondly, the global feature-conjunction detector uses their ability to detect coincident timing of spike-trains from a pair of global feature and universal conjunction detectors. Thirdly, they can implement the feedforward CDT procedure via shunting inhibition. Opposed to traditional sigmoidal neurons, all these (different) tasks could be implemented by spiking neurons only differing by threshold and time-constant τ. Note though that we had to (implicitly) take into account the limitations of the neurons with regard to temporal precision and limited firing-frequency. To this end we used sparse vectors and presented only a limited number of conjunctions simultaneously. Thus, global neurons only had to superimpose a limited number of spikes, keeping the firing-rate low, and the subsequent coincidence detection was not required to be too precise. We note that using sparse codes is generally considered efficient both from an information-theoretical (robustness of signal transfer) as well as from a metabolic point of view (low energy expenditure) (Földiák, 1990; Földiák & Young, 1995; Olshausen & Field, 1996).

4.6 Conclusions

In this chapter, we proposed an architecture for the position-invariant detection of feature-conjunctions. We have described the main idea of locally using the properties of distributed coding, and given a formal definition the proposed architecture. We have thus argued how the temporal dimension of individual spikes combined with the introduction of a novel local feature-binding operator can be employed to detect feature-conjunctions from position-invariant (aggregate) feature-detectors, in the presence of other conjunctions. In the actual implementation of the framework in networks of spiking neurons, the weights for feature detection were set similar to those obtained with temporal Hebbian learning in (Natschläger & Ruf, 1998; Bohte et al., 2002c), suggesting that the architecture could thus be learned. The incorporation of unsupervised learning in the framework is thus a logical addition. As noted by Von der Malsburg (1999), the issue of dynamic binding and structured representations is important in the field of neural networks and (sub)symbolic AI. We believe that as such, the framework developed should enable new ways of dealing with these issues.

5

FORMAL SPECIFICATION OF POSITION-INVARIANT DETECTION OF FEATURE-CONJUNCTIONS

ABSTRACT. In chapter 4, we proposed a framework for the efficient position-invariant detection of feature conjunctions. In this chapter, we formally define this framework for neural nodes that process activity in the form of tuples of spike-trains. It describes the framework in terms of well defined operators and data-structures.

5.1 Introduction

The efficient detection of feature-conjunctions on an input-grid (like the retina) is an important open issue in neural networks, with particular importance for the field of computer-vision. In chapter 4, we proposed a framework for detecting local feature-conjunctions in specialized position-invariant detectors. We used the properties of distributed coding to locally encode a feature-conjunction, and then decode, and detect, the specific content of this feature-conjunction at a position-invariant level. We showed how the proposed framework is able to correctly represent and detect multiple feature-conjunctions simultaneously.

In this chapter, we give a formal definition of the local type of encoding proposed, and the operations that are carried out on this code to ascertain the presence of specific feature-conjunction in a position invariant manner. This chapter describes these network operations in terms of well defined operations.

5.2 Formal Description

The presence of a particular feature on the input grid is characterized by the spike-trains it elicits in a set of basic neurons. For the purpose of a formal definition, we collect the timings of the spike trains for each set in a tuple, where each tuple element describes the spike train of one neuron. The local "nodes" of the architecture outlined in chapter 4 process tuples of spike-trains (TST's). The formal operations that nodes of the architecture can perform on TST's are defined in the following.

In the remainder, we use the notation like $(d, e, \ldots \in)D$ to introduce a domain D and some typical elements d, e. That is, for the subsequent use of a variable d or e, one can assume that it is an element of D.

DEFINITION 1 *Let* $(a, b, \ldots \in)\mathbb{R}$ *be the domain of single spikes.*
Let $(s, t, u, v, \ldots \in)ST \subseteq \mathbb{R}^*$ *be the domain of spike-trains with as elements the strictly monotonically increasing sequences denoting the timing of single spikes. An empty sequence is denoted by* ε*. The concatenation operator is denoted by* $\odot$*: given single spike* a *and spike-train* s*, concatenation yields the new spike-train* $a \odot s$*, which is the same as* s*, except that the element* a *is added in front.*

The set TST of tuples of n *spike-trains is defined by:* $(S, T, U, V, \ldots \in)TST = \prod_{i=1}^{n} \mathbb{R}^*$*. By* S_i *we denote the projection of* S *on the* i*-th component. An empty TST is denoted by* $\mathfrak{E}$ *(i.e.* $\mathfrak{E} = (\varepsilon, \ldots, \varepsilon)$*).*

An example of the use of the concatenation operator: $1 \odot <2.6, 4> = <1, 2.6, 4>$. A neuron may produce a spike-train $s = <3.1, 7, 12.34>$ or an empty spike-train ε. Some n neurons together produce a tuple S of spike-trains, e.g. $(S_1, S_2, \ldots, S_n)$.

DEFINITION 2 *The combination of two TST's is defined as* $S \parallel T = (S_1 \parallel T_1, \ldots, S_n \parallel T_n)$*, with* $\varepsilon \parallel s = s$*,* $s \parallel \varepsilon = s$*, and with s and t non-empty:*

$$(a \odot s) \parallel (b \odot t) = \begin{cases} a \odot (s \parallel (b \odot t)) & \text{if } a < b \\ b \odot ((a \odot s) \parallel t) & \text{if } b < a \\ a \odot (s \parallel t) & \text{if } a = b. \end{cases}$$

The operator Σ *extends* $\parallel$ *and denotes the combination of more than two TST's.*

EXAMPLE *The combination* $S \parallel T$ *of two TST's* $S = (<2, 3.1, 4>, <2.5, 3.1>)$ *and* $T = (<3.9, 4.2>, \varepsilon)$ *equals* $(<2, 3.1, 3.9, 4, 4.2>, <2.5, 3.1>)$.

DEFINITION 3 *For spike-trains* s, t, *we have* $s \subseteq t$ *if there is a spike train* u *such that* $s \parallel u = t$. *We say* $S \subseteq T$ *if for all projections* $S_i \subseteq T_i$, $i = 1, \ldots, n$.

EXAMPLE *For example,* $<1, 2, 3> \subseteq <0.5, 1, 1.3, 2, 3>$ *because* $<1, 2, 3> \parallel <0.5, 1.3> = <0.5, 1, 1.3, 2, 3>$.

Now, we define the *watermarking*-operator (Context-Dependent Thinning, CDT).

DEFINITION 4 *A CDT operation* $\lhd$ *for binding is defined by:* $S \lhd T = (S_1 \lhd T_1, \ldots, S_n \lhd T_n)$, *with:* $t \lhd \varepsilon = t$, $\varepsilon \lhd t = \varepsilon$, *and for non-empty spike-trains:*

$$(a \odot s) \lhd (b \odot t) \begin{cases} a \odot (s \lhd (b \odot t)) & \text{if } a < b \\ \varepsilon & \text{otherwise.} \end{cases}$$

Note that for all i, either $(S \lhd T)_i = \varepsilon$ or $(T \lhd S)_i = \varepsilon$ (or both equal ε). Another interesting property is that for all S, T, U, we have $S \lhd (T \parallel U) = (S \lhd T) \lhd U$.

EXAMPLE *The CDT operation on spike-trains* $s = <1, 2, 3>$ *and* $t = <2.5, 3.5, 4.6>$, *results in a spike-train* $s \lhd t = <1, 2>$. *The operation* $S \lhd T$ *removes some spikes in* S *due to the presence of spikes in* T, *that is* $(S \lhd T) \subseteq S$.

DEFINITION 5 *We define* $\Gamma^m(S) = T$, *with*

$$T_i = \sum_{k=i-m}^{i+m} \begin{cases} S_n & \text{if } k \bmod n = 0 \\ S_{k \bmod n} & \text{otherwise.} \end{cases}$$

We abbreviate $S \lhd (\Gamma^m(T))$ by $S \lhd^m T$. For TST's with some n elements, it is possible to select m such that the CDT procedure removes approximately half the non-empty elements, see section 4.3.

For the construction of networks, we next define local feature-detectors, local binding-detectors and conjunction-detectors.

DEFINITION 6 *A local feature-detector lfd for feature S in T is defined by:*

$$\mathit{lfd}(S,T) = \begin{cases} T & \text{if } S \subseteq T \\ \mathfrak{E} & \text{otherwise.} \end{cases}$$

DEFINITION 7 *A generic local binding-operator glb is defined by:*

$$glb(S,T) = \begin{cases} S \lhd^m T & \text{if } |S| \cdot |T| > \theta \\ \mathfrak{E} & \text{otherwise.} \end{cases}$$

Here, the number of spikes $|S|$ in S is defined by $|S| = |S_1| + \ldots + |S_n|$, with $|s| = \text{length}(s)$ and $|\varepsilon| = 0$. The threshold θ is the required input-activation.

The idea is that a non-empty TST, S, is partially propagated by the *glb* operator if there is also a non-empty TST, T, present (a conjunction). The CDT operation is then performed on S using T, resulting in a TST like S, except for that some elements from the spike-trains in S are removed. We use two complementary *glb* operators, denoted $(X \lhd Y)$ and $(Y \lhd X)$. When presented with *any* non-empty TSTs S and T, these operators yield watermarked versions of S or T, respectively. For global conjunction detection, the presence of the correct watermarked TST is determined by the Ω operator:

DEFINITION 8 *We define the presence operator Ω by:*

$$\Omega(S,T) = (\Omega(S_1,T_1), \ldots, \Omega(S_n,T_n)), \text{ where } \Omega(s,t) = \begin{cases} s & \text{if } s \subseteq t \\ \varepsilon & \text{otherwise.} \end{cases}.$$

The $\Omega(S,T)$ operation checks which spike trains of S are present in T and outputs those spike-trains that are present.

DEFINITION 9 *The conjunction detector $cd(S,T,U,V)$ for the S,T conjunction is defined by:*

$$cd(S,T,U,V) = \begin{cases} \Omega((S \lhd^m T),U) \parallel \Omega((T \lhd^m S),V) & \text{if } |\Omega((S \lhd^m T),U)| + \\ & |\Omega((T \lhd^m S),V)| \geq \alpha \\ \mathfrak{E} & \text{otherwise.} \end{cases}$$

The threshold α detects matching, we set it to to $|S \lhd^m T| + |T \lhd^m S|$.

The conjunction detector propagates a specific mix of the input TST's if both sufficiently match the patterns. It checks whether watermarked versions of S and T are present in U and V, respectively.

Now we have all the (feed-forward) elements for our three-level architecture:

1. a first level in which we have:
   – for each location i local feature detectors *lfd*(A, U^i), *lfd*(B, U^i) looking for specific patterns $A, B \in$ TST in the activity U^i of the local set of neurons.
   – for pairs of locations next to each other generic local binding operators *glb*(U^i, U^{i+1}) and *glb*(U^{i+1}, U^i) that look for pairs of sufficient activation, and then output watermarked features $U^i \lhd^m U^{i+1}$ and $U^{i+1} \lhd^m U^i$.
2. a second level combining local feature and generic binding detectors via $\|$ into respective global feature and generic binding detectors: Σ_i*lfd*(A, U^i), Σ_i*lfd*(B, U^i), Σ_i*glb*(U^i, U^{i+1}), Σ_i*glb*(U^{i+1}, U^i).
3. a third level, for conjunction detection. A *cd*-operator connects to two global feature detectors and two global generic binding detectors:
   cd$(\Sigma_i$*lfd*$(A, U^i), \Sigma_i$*lfd*$(B, U^i), \Sigma_i$*glb*$(U^i, U^{i+1}), \Sigma_i$*glb*$(U^{i+1}, U^i))$.

Up to now we have abstracted from the fact that computations take time. In order to detect a feature a node can only produce output if it has seen the feature. Therefore, to actually implement the network, one has to build in a delay in all nodes in the network. This can be done by using a constant Δ, which models the maximal computation time.

DEFINITION 10 *For a number $\Delta \in \mathbb{R}$ the operation of delayed propagation Δ of a TST S, is defined as: $\Delta(S) = (\Delta(S_1), \ldots, \Delta(S_n))$, with the propagation of an empty sequence ε defined as $\Delta(\varepsilon) = \varepsilon$, and the propagation on a non-empty spike-train is recursively defined as: $\Delta(a \odot s) = (a + \Delta) \odot (\Delta(S))$.*

EXAMPLE *A the Δ-operator applied to a TST S with $\Delta = 1$, $S_1 =< 1, 2.1, 3 >$ and $S_2 =< 1.5, 2.1 >$, yields $1(S) = (< 2, 3.1, 4 >, < 2.5, 3.1 >)$.*

As an example we go through the detection of conjunctions *BA* and *CD* of features A, B, C, D, when both conjunction are simultaneously presented on an input grid. The corresponding TST's are shown in table 5.1. We

$$
\begin{array}{l}
A = (< 2.1 >, < 3.4 >) \\
B = (< 4.2 >, < 1.1 >) \\
C = (< 1.0 >, < 4.1 >) \\
D = (< 3.0 >, < 1.2 >) \\
\\
U^L = \Sigma_i glb(U^i, U^{i+1}) = (B \lhd A) \parallel (C \lhd D) = (< 1.0 >, < 1.1 >) \\
U^R = \Sigma_i glb(U^{i+1}, U^i) = (A \lhd B) \parallel (D \lhd C) = (< 2.1 >, < 1.2 >) \\
\\
\Sigma_i lfd(A, U^i) = A = (< 2.1 >, < 3.4 >) \\
\Sigma_i lfd(B, U^i) = B = (< 4.2 >, < 1.1 >) \\
\Sigma_i lfd(C, U^i) = C = (< 1.0 >, < 4.1 >) \\
\Sigma_i lfd(D, U^i) = D = (< 3.0 >, < 1.2 >) \\
\\
cd(\Sigma_i lfd(B, U^i), \Sigma_i lfd(A, U^i), U^L, U^R) = (< 2.1 >, < 1.1 >) \\
cd(\Sigma_i lfd(C, U^i), \Sigma_i lfd(D, U^i), U^L, U^R) = (< 1.0 >, < 1.2 >) \\
cd(\Sigma_i lfd(C, U^i), \Sigma_i lfd(A, U^i), U^L, U^R) = (< \varepsilon >, < \varepsilon >) \\
cd(\Sigma_i lfd(B, U^i), \Sigma_i lfd(D, U^i), U^L, U^R) = (< \varepsilon >, < \varepsilon >)
\end{array}
$$

Table 5.1: Output of operators when only the conjunctions BA and CD are present. Detectors for BA and CD output a TST, detectors for "ghosts" CA and BD do not.

see that there is only output from the *cd*-detectors for the existing conjunctions. For simplicity, we took TST's with very few elements, and we left out the delays.

5.3 Conclusion

Summarizing, we have formally defined the spike-time vector as a data-structure (the TST), and defined the operations on such a data-structure that, when put together, enable the correct detection of simultaneously present feature conjunctions.

6

THE EFFECTS OF PAIR-WISE AND HIGHER ORDER CORRELATIONS ON THE FIRING RATE OF A POST-SYNAPTIC NEURON

ABSTRACT Coincident firing of neurons projecting to a common target cell is likely to raise the probability of firing of this post-synaptic cell. Therefore synchronized firing constitutes a significant event for post-synaptic neurons and is likely to play a role in neuronal information processing. Physiological data on synchronized firing in cortical networks is primarily based on paired recordings and cross-correlation analysis. However, pair-wise correlations among all inputs onto a post-synaptic neuron do not uniquely determine the distribution of simultaneous post-synaptic events. We develop a framework in order to calculate the amount of synchronous firing that, based on maximum entropy, should exist in a homogeneous neural network in which the neurons have known pair-wise correlations and higher order structure is absent. According to the distribution of maximal entropy, synchronous events in which a large proportion of the neurons participates should exist, even in the case of weak pair-wise correlations. Network simulations also exhibit these highly synchronous events in the case of weak pair-wise correlations. If such a group of neurons provides input to a common post-synaptic target, these network bursts may enhance the impact of this input, especially in the case of a high post-synaptic threshold. Unfortunately, the proportion of neurons participating in synchronous bursts can be approximated by our method only under restricted conditions. When these conditions are not fulfilled, the spike trains have less than maximal entropy, which is indicative of the presence of higher order structure. In this situation, the degree of synchronicity cannot be derived from the pair-wise correlations.

6.1 Introduction

In this chapter, we study the population behavior of groups of interconnected spiking neurons. In particular, we study the (likely) interpretation of biologically measurements on *neural correlations* related to the precision with which single spiking neurons fire. As such, this chapter makes a considerable effort to take biological considerations into careful consideration.

The occurrence of correlations in the spike-trains of neurons responding to the same object has raised considerable excitement during the last decade (reviewed by Singer and Gray (1995)). Correlations between pairs of neurons are thought to reflect a high degree of synchronous firing within a larger assembly of neurons (Singer, 1995; Engel et al., 1992) and can have a high temporal precision, in the range of a few milliseconds (Eckhorn et al., 1988; Gray et al., 1989; Konishi, 1991; Roelfsema et al., 1997; Alonso et al., 1996; Abeles et al., 1993; Gray et al., 1989). Von der Malsburg (1981) suggested that assemblies of neurons might convey additional information by firing in synchrony, since synchrony could be instrumental in forming relationships between the members of such an assembly.

However, the possible relevance of fine temporal structure in spike-trains opposes another widespread belief. In real nervous systems, the irregular timing of cortical action potentials is often attributed to stochastic forces acting on the neuron (Bair et al., 1994; Shadlen & Newsome, 1994). In such a stochastic model, the information is thought to be conveyed to the next processing stage (cortical layer) by pools of neurons using a noisy rate code. Each individual neuron is considered to be a slow, unreliable information processor, reflecting changes in its receptive field by modulating its average firing rate. Only by pooling the information from a larger number of neurons, a reliable rate code can be obtained. Obviously, this scheme does not need precise timing of the individual spikes to convey information.

These two opposing views on the role of temporal structure of neuronal information processing are subject of considerable debate (König, Engel, & Singer, 1996; Shadlen & Newsome, 1995; Softky & Koch, 1993). This debate has focused on two important questions. First, is the cortical neuron a coincidence detector (on the millisecond time-scale) and second, how much coincident input is there?

The first question refers to the relevance of synchronous pre-synaptic spikes. It has been suggested that synchronous input induces a higher firing rate in the post-synaptic target cell. Does this assumption hold, especially on a millisecond time-scale? This question has been amply recognized, and several studies have attempted to answer it. Shadlen and Newsome (1995) argue that, based on physiological considerations, a cortical neuron is not capable of detecting very tightly synchronized input. However, others have argued that cortical neurons might have a high sensitivity for the synchronicity in their input (Softky, 1995; König et al., 1996). Softky (1995) pointed out that the biological data available leave too many parameters undetermined to draw any definite conclusions on biological properties that distinguish the various models. Two further studies on the impact of synchronized input on a post-synaptic target reinforce this observation. Using detailed models of groups of neurons, Bernander, Koch, and Usher (1994), and Murthy and Fetz (1994) studied the impact of coincident input on the firing rate of a post-synaptic neuron. Their conclusions are similar to Softky's: within the biologically plausible parameter ranges synchrony may either increase or decrease the firing rate of post-synaptic neurons.

In the present study we attempt to shed more light on the second question: how much synchrony is there? In general, it is implicitly assumed that pair-wise correlations provide a good estimate of the amount of synchrony in a pool of neurons from which recordings are obtained. However, to date there are no direct electrophysiological measurements of large synchronous pools of cortical neurons. Most of the physiological data on neuronal synchronization so far have been obtained using cross-correlation techniques (with the notable exception of the work of Abeles et al. (1993)). These techniques merely provide information on the pair-wise correlation: the probability of finding a *pair* of neurons that fire at the same time (that is, within some time-window). Unfortunately, pair-wise correlations only provide an indirect estimate of the probability of higher-order events, like the coincident firing of, say, 5 or 50 neurons. Even when the pair-wise correlations between all neurons of a network are fixed, the probability of these higher-order events remains undetermined, as is illustrated in figure

6.1. Pair-wise correlation is defined as the difference between the probability of two neurons firing simultaneously, and the product of their firing rates (the coincidence rate dictated by chance). This value is the same for any pair of neurons in the three panels of figure 6.1C. However, the number of spike-triplets differs considerably from panel to panel. Given this example, it is quite clear that the number of neurons that fire within some time-window (the measure of coherence) is not exclusively determined by the pair-wise correlation coefficient. And, although this is an artificial example, it is already quite difficult to determine intuitively how many of these triplets can be attributed to higher-order correlation, and how many result from two "doublets" that happen to occur at the same time. In a first attempt to quantify the incidence of higher-order correlations, Martignon, Hasseln, Grün, Aertsen, and Palm (1995) analyzed data from six cortical neurons. Unfortunately, we found that their methods cannot be used for the analysis of large numbers of neurons (as will be discussed). The main

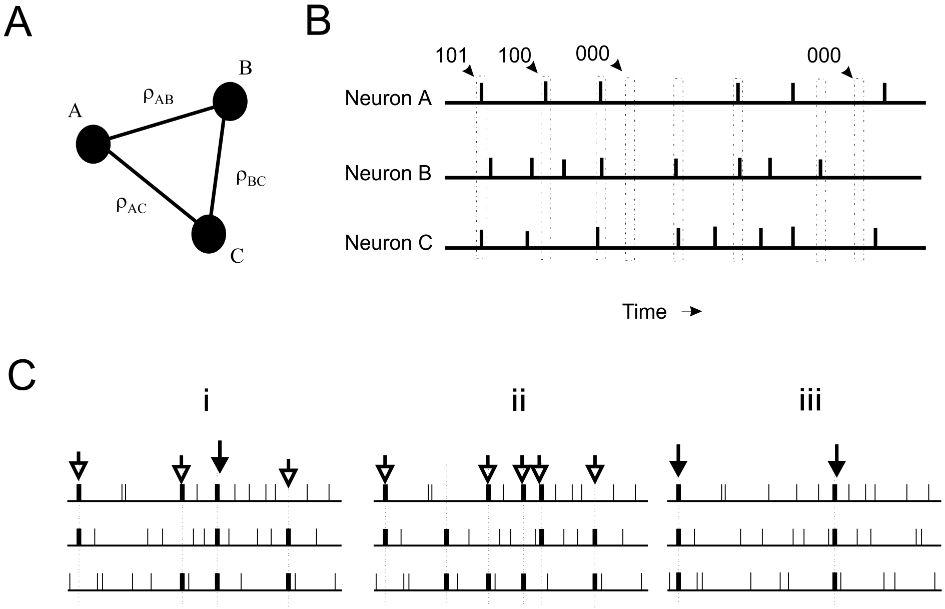


Figure 6.1: (A) Three neurons with correlated activity. The pair-wise correlation coefficients are ρ_{AB}, ρ_{AC} and ρ_{BC}. (B) Examples of different spike configurations in windows of the same size. The horizontal lines represent the spike trains, and each tick denotes a spike. Spikes occurring with a time-window (dotted box) are considered to be coincident. (C) Three examples of correlated spike trains. Pairs of neurons in the three panels have the same pair-wise correlation. Spike-doublets are shown as unfilled arrows, and triplets as filled arrows. It can be seen that the number of triplets differs from panel to panel.

goal of the present chapter is to study the relationship between pair-wise correlations and the amount of synchronicity in a pool of neurons, and to determine the impact of the synchronous events on a post-synaptic target cell.

6.2 Mathematical Solution of the three-neuron problem

To illustrate the general methodology used for estimation of the probability of higher order events, we examine the three neuron network of figure 6.1. We wish to calculate the probability that a triplet (or an N-cluster, where N equals 3) occurs within a given time-window. The null hypothesis is that no structure is present in the spike-trains other than the pair-wise correlations. That is, all triplets should be due to the occurrence of two doublets at the same time, by chance forming a triplet. Csiszar (1975) has proved the unique existence of a distribution with just this property. The basic approach for calculating the probability distribution involves maximizing the informational entropy of the data, while preserving the pair-wise correlation and the firing rate (see also Martignon et al. (1995)). This informational entropy is a measure of the "order" in the data: the more structured the data, the lower the entropy. By measuring the pair-wise correlations and the firing rates (the first order correlation), a certain degree of order is fixed. Taking these constraints into account, maximizing the entropy will minimize all higher-order correlations since higher-order correlations will add "structure" to the distribution, further lowering the entropy. Therefore, maximal entropy implies minimal higher-order correlations. Our aim is to obtain the distribution of N-clusters that has maximal entropy. We will illustrate the procedure by computing this distribution for three connected neurons.

The neurons are labeled A, B and C, their firing probabilities f_{1A}, f_{1B} and f_{1C} and the pair-wise correlation coefficients are denoted as ρ_{AB}, ρ_{BC} and ρ_{AC} (f_{1i} denotes the probability that neuron i fires in a particular time-window, or *time bin*. Given the width of the time-bin, division of the firing probability by the length of the time bin in seconds yields the firing rate of the neuron. The rationale of the suffix 1 will become clear below). Within a time bin, we assume that only one possible configuration is realized, i.e. a neuron fires at most one spike within a time bin. Given these coefficients as constraints, probabilities G_{abc} for all 2^n possible events within a single time bin can be calculated. For example G_{010} designates the probability of finding a time bin where B is firing and A and C are silent (Fig. 6.1B).

There are 8 probabilities to solve for. Since firing probabilities and pair-wise correlation coefficients are fixed, 7 equations can easily be obtained: The sum of the probabilities of all possible events equals one:

$$G_{000} + G_{001} + G_{010} + G_{011} + G_{100} + G_{101} + G_{110} + G_{111} = 1 \tag{6.1}$$

The firing-probability of a neuron A is:

$$G_{100} + G_{101} + G_{110} + G_{111} = f_{1A} \tag{6.2}$$

(The equations for f_{1B} and f_{1C} are derived analogously) Let f_{2AB} denote the probability that A and B fire simultaneously (the suffix now indicates a cluster of size 2). f_{2AB} is determined by the correlation coefficient ρ_{AB}, since

$$\rho_{AB} = \frac{f_{2AB} - f_{1A}f_{1B}}{\sqrt{(f_{1A} - f_{1A}^2)(f_{1B} - f_{1B}^2)}} \tag{6.3}$$

This implies that ρ_{AB} determines f_{2AB} and since f_{1A} and f_{1B} are fixed, this yields:

$$G_{110} + G_{111} = f_{2AB} \tag{6.4}$$

and corresponding equations are derived for ρ_{AC} and ρBC. These equations yield one free parameter (G_{111}, for example), which determines the entropy of the probability distribution. By calculating the value at which the distribution has maximal entropy, we fix this last free parameter such that the probability contains the least structure in terms of higher-order correlations. The entropy-function is defined as:

$$S \equiv -\sum_i G_i \ln(G_i) \qquad i \text{ over all possible spike configurations } \{0,1\}^3 \tag{6.5}$$

The entropy is maximized by solving the following equation, which has a unique solution:

$$\frac{\partial S}{\partial G_{111}} = 0. \tag{6.6}$$

Using this equation the probability distribution of configurations with the desired zero higher-order correlation can easily be calculated (see also Martignon et al. (1995)).

6.3 Calculating the Distribution with N Identical Neurons

For more than three neurons, the equations can no longer be solved easily by analytical means. For instance, for 4 neurons, the same calculations yield the entropy as a function of 5 free parameters. Calculating the maximum of this function is already quite complicated. Extending this to N neurons, only $1 + N(N+1)/2$ equations are determined by the pair-wise correlations and firing-probabilities, with 2^N parameters to solve for. To overcome this problem, an iterative algorithm for calculating the maximal entropy distribution has been provided by Gokhale and Kullback (1978). Unfortunately, this algorithm has a serious drawback: the number of calculations increases exponentially with the number of neurons. If one is interested in the behavior of many correlated neurons, computational restrictions prohibit this method of calculating the distribution for more than 20 neurons on a workstation, thus putting the more serious number of neurons out of reach of even the fastest supercomputers.

To overcome the problem of increasing computation time we made the additional assumption that all neurons have identical properties, which implies a major degeneration of the probability-space. Since any two neurons are equal, the same will hold for the probability of all permutations of spike configurations. In addition, the N firing probabilities and the $N(N-1)/2$ pair-wise correlation coefficients will also be equal. The distribution of spike configurations is described by $N+1$ variables, D_0 through D_N, where D_i denotes the probability of a particular spike configuration in which exactly i neurons fire. Note that there are $\binom{N}{i}$ such configurations.

In the case of $N = 7$: $D_1 = G_{1000000} = G_{0100000} = \ldots = G_{0000001}$; $D_2 = G_{1100000} = G_{1010000} = G_{0010010}$, etc. For the general case of N neurons three equations are derived from the pair-wise correlation. Since all permutations of i spiking and $N-i$ silent neurons have the same probability D_i, the requirement is that the firing probability and the sum of all probabilities should add up to 1. First, the probabilities should add up to 1:

$$1 = \sum_{i=0}^{N} \binom{N}{i} \cdot D_i \tag{6.7}$$

Second, the firing probability f_1 is fixed. For example, in the case that $N = 7$ it is easy to see that f_1 equals:

$$f_1 = \sum_{i=0}^{1}\sum_{j=0}^{1}\sum_{k=0}^{1}\sum_{l=0}^{1}\sum_{m=0}^{1}\sum_{n=0}^{1} G_{1ijklmn} = \sum_{i=1}^{7} \binom{6}{i-1} \cdot D_i \tag{6.8}$$

In the general case of N neurons this reads (see also appendix A):

$$f_1 = \sum_{i=1}^{N} \binom{N-1}{i-1} \cdot D_i \tag{6.9}$$

Under the assumption of identical neurons, in (6.3) f_{1B} equals f_{1A}. Rewriting (6.3), replacing f_{1A} and f_{1B} by f_1, and f_{2AB} by f_2 yields:

$$\rho = \frac{f_2 - f_1^2}{f_1(1 - f_1)} \tag{6.10}$$

Thus, f_2 can once again be calculated from the firing probability of a neuron and the correlation. f_2, the probability that any two particular neurons fire at the same time equals:

$$f_2 = \sum_{i=2}^{N} \binom{N-2}{i-2} \cdot D_i \tag{6.11}$$

By maximizing the entropy S, we derive the remaining $N-2$ equations: The entropy is defined as:

$$S = \sum_{i=0}^{N} \binom{N}{i} \cdot D_i \ln(D_i) \tag{6.12}$$

Maximization of entropy yields:

$$\frac{\partial S}{\partial D_i} = 0 \qquad \text{for } i = 3 \ldots N \tag{6.13}$$

We now solve these equations for the maximal entropy.

First, we eliminate D_0, D_1 and D_2 with the relationships fixed in equations

(6.9)-(6.11):

$$\begin{aligned}
D_0 &= 1 - \sum_{i=1}^{n} \binom{n}{i} D_i \\
&= 1 - nD_1 - n\binom{n}{2} D_2 - \sum_{i=3}^{n} \binom{n}{i} D_i && (6.14) \\
D_1 &= f_1 - \sum_{i=2}^{n} \binom{n-1}{i-1} D_i \\
&= f_1 - [n-1]\, D_2 - \sum_{i=3}^{n} \binom{n-1}{i-1} D_i && (6.15) \\
D_2 &= f_2 - \sum_{i=3}^{n} \binom{n-2}{i-2} D_i && (6.16)
\end{aligned}$$

Solving for D_0 , D_1 and D_2:

$$\begin{aligned}
D_0 &= 1 - nf_1 + n(n-1)f_2 - \binom{n}{2} f_2 + \\
&\quad + \left[\binom{n}{2} - n(n-1)\right] \cdot \sum_{i=3}^{n} \binom{n-2}{i-2} D_i + \\
&\quad + n\sum_{i=3}^{n} \binom{n-1}{i-1} D_i - \sum_{i=3}^{n} \binom{n}{i} D_i && (6.17) \\
D_1 &= f_1 - (n-1)f_2 + (n-1)\sum_{i=3}^{n} \binom{n-2}{i-2} D_i + \\
&\quad - \sum_{i=3}^{n} \binom{n-1}{i-1} D_i && (6.18) \\
D_2 &= f_2 - \sum_{i=3}^{n} \binom{n-2}{i-2} D_i && (6.19)
\end{aligned}$$

With the entropy defined as in eq. (6.12), it is easy to verify that this is a concave function, and therefore it has a single, unique maximum. Using eq. (6.13), and eqs. (6.17)-(6.19), maximization yields:

$$\begin{aligned}\frac{\partial S}{\partial D_i} &= -\left\{\left[\binom{n}{2} - n(n-1)\right]\binom{n-2}{i-2} + n\binom{n-1}{i-1} - \binom{n}{i}\right\}\ln(D_0) + \\ &\quad -n\left\{(n-1)\binom{n-2}{i-2} - \binom{n-1}{i-1}\right\}\ln(D_1) + \\ &\quad +\binom{n}{2}\binom{n-2}{i-2}\ln\left(D_2\right) + \\ &\quad -\binom{n}{i}\ln\left(D_i\right) = 0 \qquad (6.20)\end{aligned}$$

We note that the equations (6.17)-(6.19) are linear in D_i, and that a concave function remains concave on a linear subspace. In other words: equations (6.17)-(6.19) and (6.13) taken together, also have a unique solution. Equation (6.20) can be rewritten as:

$$\begin{aligned}\ln(D_i) &= -(-\frac{1}{2}i + 1)(i-1)\ln(D_0) - i(i-2)\ln(D_1) + \\ &\quad +\frac{1}{2}i(i-1)\ln(D_2) \qquad \text{for } i = 3\ldots N \qquad (6.21)\end{aligned}$$

Inserting these values for D_i into equations (6.17-6.19) yields three equations with three unknowns (since D_i is replaced with functions in D_1, D_2 and D_3) and a unique solution which we approximate using the Newton-Raphson method, as described in (Press, Flanney, Teukolsky, & Vetterling, 1986).

The maximal entropy distribution, thus defined, depends on only two parameters, the firing probability f_1, and the pair-wise correlation ρ. Figure 6.2 illustrates the shape of the N-cluster distribution with maximal entropy for a network of 150 neurons, and its dependence on f_1 and ρ. Figure 6.2 shows the probability P_i that a particular *number* of neurons fire simultaneously, where P_i equals the sum of the probabilities of all configurations in which exactly i neurons fire:

$$P_i = \binom{N}{i} \cdot D_i \qquad (6.22)$$

For small values of ρ, the distribution of N-clusters approaches a binomial distribution. This determines the first peak of the N-cluster distribution, corresponding to small cluster sizes. For larger values of ρ, a second peak appears in the distribution, the amplitude of which grows with increasing

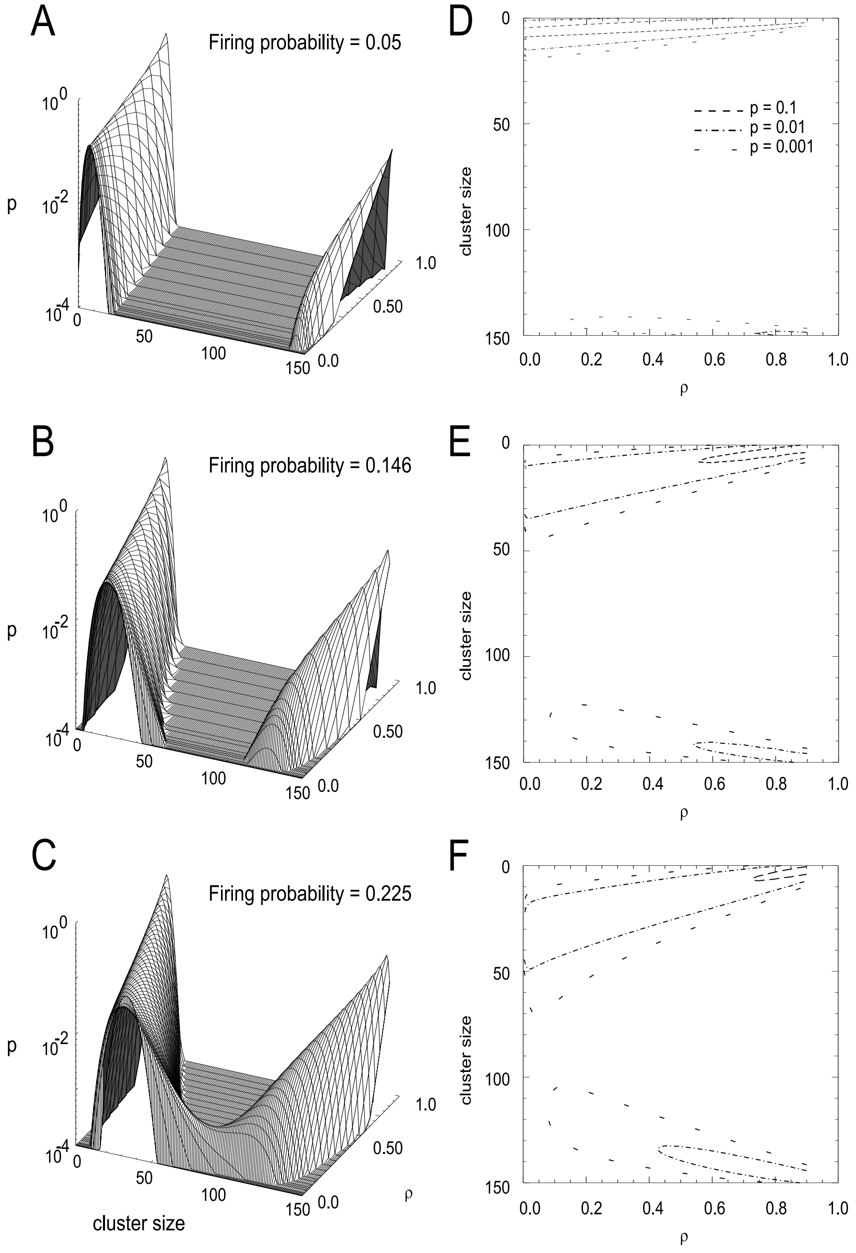


Figure 6.2: (A-C) Probability of N-clusters (p) as a function of the pair-wise correlation ρ and the cluster size, for three firing probabilities, $f_1 = 0.05$ (A), 0.146 (B) and 0.225 (C). Probabilities are clipped at a value of 10^{-4}. (D-F) Contour plots of the same data show that the separation between the two peaks becomes larger with increasing ρ. Contour levels indicate probabilities of 0.001, 0.01 and 0.1.

ρ. A second influence of increasing ρ is a divergence of the two peaks. This divergence can be seen most clearly in the contour plots of figure 6.2D-F. Indeed, in the limiting case of $\rho = 1$, the first peak approaches a cluster size of 0, and the second peak a cluster size of 150, since all neurons fire at exactly the same time.

An increase in f_1 shifts the first peak of the distribution to larger values, as is predicted by the binomial distribution (Fig. 6.2A-C). Remarkably, an increase in f_1 is also associated with a shift of the second peak, to *smaller* values. Thus, the maximal entropy distribution predicts that low firing probabilities are associated with sparse, but highly synchronized bursts (eg: the average size of the second peak in A at $\rho = 0.165$ equals 142 vs 110 in C, whereas the cumulative probability increases from 0.009 in A to 0.076 in C. The second peak is defined as all clusters with $p > 10^{-4}$, starting on the positive slope after the first peak). With a higher firing probability, synchronous bursts occur more frequently, but comprise fewer spikes.

6.4 An artificial neural network

We now compare N-cluster distributions obtained from simulations of an artificial neural network to the maximal entropy distribution. Any difference between the two distributions can then be attributed to higher-order correlations. The network used is based on a network described by Deppisch et al. (1993), which, in turn, is based on the work of MacGregor and Oliver (1974). This network was chosen since it could easily be adapted to consist of identical neurons.

6.4.1 The neuron model The chosen neuron model is a less abstract, biological model when compared to the previously used Spike Response Models, in that it describes how impacting spikes mediate charged currents entering and leaving the neuron. This interaction, expressed in coupled partial differential equations, describe the resulting changes in the membrane potential that (can) result in the generation of an action potential. In the simple model we use, only the primary charge-carrying current (potassium) is modeled. Thus, the neuron model involves four variables: the neuron's membrane potential $E(t)$, the potassium current $g(t)$, the spiking threshold $\theta(t)$ and the neuronal output $o(t)$. The dynamics of a neuron i are described by four coupled equations:

$$
\begin{aligned}
\tau_E \frac{dE_i(t)}{dt} &= -(E_i(t) - E^0) - (g_i(t) - g^0)(E_i(t) - E^k) + \eta_i(t) + \\
&\quad - \left[\sum_j w_{ij} \cdot o_j(t - \tau_i j) \right] (E_i(t) - E^{ex}) \qquad (6.23) \\
\tau_\theta \frac{d\theta_i(t)}{dt} &= -(\theta_i(t) - \theta^0) + c(E_i(t) - E^0) 0 \leq c \leq 1 \qquad (6.24) \\
\tau_g \frac{dg_i(t)}{dt} &= -(g_i(t) - g^0) + \tau_g b o_i(t) \qquad (6.25) \\
o_i(t) &= \begin{cases} 1 & \text{if } E_i(t) > \theta_i(t) \\ 0 & \text{otherwise} \end{cases} \qquad (6.26)
\end{aligned}
$$

Without input, the membrane potential $E_i(t)$ is driven towards its resting value E^0 with a time constant τ_E. An influx of potassium drives the potential towards the potassium equilibrium potential E^k. Excitatory input moves the potential towards the equilibrium value E^{ex}. In the simulations, the values used in (Deppisch et al., 1993) were adopted: $E^0 = 0$, $\tau_E = 2.5$ msec, $E^k = -1$, $E^{ex} = 7$. We remark that the membrane time constant

τ_E may appear to be rather small, although some have argued that it may be within the biologically plausible range (König et al., 1996; Bernander et al., 1991). In the discussion we will address the dependence of our results on this particular choice. Input to the cells was strictly excitatory, with synaptic delay $\tau_{ij} = 0.5$ msec. External input to the network's neurons is provided by external stochastically spiking neurons and is treated equivalent to internal input. Internal noise is added by the additive noise term $\eta_i(t)$ in (6.23), with a standard deviation of $0.06E_i(t)$. Equation (6.24) describes the slow adaptation of the threshold θ_i to the membrane potential, modeling an adaptation of the neuron to excitation ($\theta_0 = 1$, $\tau_\theta = 10$ msec, $c = 0.3$). Equation (6.25) describes the dynamics of the potassium current in response to incoming spikes. The current is driven towards $g_0 = 0$ with time constant τ_g (5 msec). In the case of a input spike, the potassium current rises by an amount $b(4.0)$ corresponding to the outward potassium current. A spike is generated each time the membrane potential exceeds the threshold (6.26), after which, due to refraction of the membrane potential, the neuron cannot fire another spike for another 1.5 msec. In the actual numerical implementation of the neuron model, a discrete time analogue of (6.23)-(6.26) was used. These equations were iterated with time-steps corresponding to 0.5 msec of real time.

Each neuron was connected to every other neuron with a fixed homogeneous synaptic weight $w_{ij} = \frac{w_{int}}{N} > 0$. Here, w_{int} denotes the sum of the strengths of all synapses from within the network impinging on a single cell. External input consisted of N independent stochastic elements generating spikes with $p = 0.05$ per msec (50 Hz), connected in a 1 to 1 fashion with the network neurons with a constant weight $w_{ext} = 0.8$.

The parameter w_{int} was varied during the simulation. Increasing the value of the weights drives the network from a stochastic mode of seemingly random firing into an oscillatory mode of bursts of tightly synchronized firing (Deppisch *et al.*, 1993). Somewhere within this range of synaptic weights the network alternates between the stochastic and oscillatory mode.

All results are based on simulations with a network with $N = 150$ neurons, with the exception of the results in section 6.4.6. This was a compromise between a realistic number of neurons, and a network consideration: the absence of inhibitory neurons makes a network very quickly prone to saturation, a state where the constant excitation induces a very sharp oscillation. A further increase in the number of neurons also narrows the weight-range in which the network is in the alternating state. The network output was obtained by recording the output of all 150 neurons. A sample of the output of 50 neurons is shown in figure 6.3.

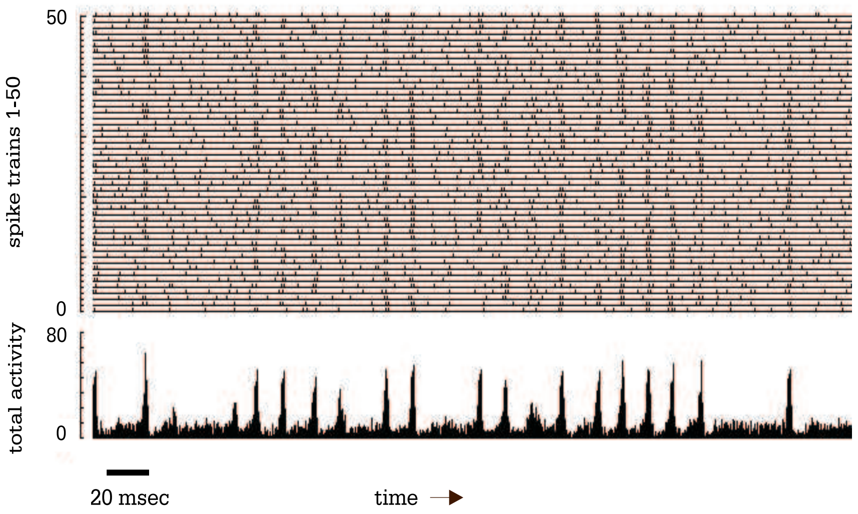


Figure 6.3: Upper panel, activity of 50 of the 150 neurons. On the horizontal axis time is displayed (sampling-time, 0.5 msec). The vertical ticks indicate action potentials. Oscillatory episodes are alternated by periods of more random activity. In the lower panel the summed activity of all 150 neurons is shown.

6.4.2 Network Simulations In order to vary the average pair-wise correlation, simulations were performed with different values of the synaptic weight w_{int}. As was noted by Deppisch et al. (1993), the network exhibits three distinct modes, which depend on the value of the synaptic weight. The first, at low values of the synaptic weights, is the stochastic mode in which the average correlation between pairs of neurons is near zero. At the other end of the scale is the mode with high values of internal weight. In this mode the network activity is highly oscillatory. In between these extremes are synaptic weight values for which the network exhibits episodes in which many neurons fire synchronously, alternated by episodes in which neurons are synchronized to a lesser degree. Typical activity of the neurons in these three network modes is shown in figures 6.4A-C. Also plotted are the cross-correlations between a pair of neurons in the network (Fig. 6.4D-F). As observed in (Deppisch et al., 1993), these cross-correlograms show qualitative resemblance to data obtained from electrophysiological recordings (e.g König, Engel, Roelfsema, and Singer (1995)). Remarkably, the network with intermediate synaptic strength ($w = 3.875$, $\rho = 0.03$, for 2 msec bin-size) exhibits occasional population bursts, which hardly show up in the correlation function (Fig. 6.4B,E). Thus, a network state in which a number of highly synchronous population bursts occur, can be associated with correlation functions indicative of weak pair-wise coupling. When investigating the relationship between occurrences of N-clusters and the pair-wise correlation, an additional assumption has to be made with regard to the maximal time-difference between two spikes that are considered to be synchronous. As a first approach we used a time-window of 2 msec. The maximal firing rate of the neurons is determined by the refractory period (1.5 msec) and the duration of an action potential (0.5 msec). As all experiments, with the exception of those described in section 6.4.5, have a bin-width of 2 msec, the firing rate (in Hz) in these experiments can be obtained by multiplying the firing probability with a factor of 500. During network bursts, neurons reach this maximal firing rate, and a single spike occurs in each 2 msec bin. The effect of changing the bin-width on the distribution of cluster-sizes will be investigated in section 6.4.5.

Figure 6.5 shows the relationship between the occurrence of N-clusters and the pair-wise correlation. Plotted is the probability P_i (as defined in eq. (6.22)) of a particular number of coincident spikes, for simulations with different pair-wise correlations (different values of w_{int}). As can be seen in figure 6.5A, most 2 msec bins are occupied by N-clusters containing a fairly low number of spikes. This corresponds to the stochastic activity between bursts and the probability of these events is approximated by a binomial distribution. Synchronous bursts are represented by the second peak in the probability distribution, as can be seen more clearly in the logarithmic plot of figure 6.5B. Between these two peaks is a plateau of time bins in which an intermediate number of neurons fire. These events are caused by time bins that are aligned on the onset or end of a burst. Increases in the pair-wise correlation are associated with a slightly smaller first peak, and an enhanced second peak. For very large, and biologically implausible, pair-wise correlations a third peak containing intermediate cluster sizes is visible which can be attributed to the onset and offset of bursts. Let us now consider the impact of this distribution on a post-synaptic cell, receiving input from all the network neurons. For such a cell, the exact number of synchronous spikes (the quantity plotted in figure 6.5A,B) is not important. Rather, for a post-synaptic neuron with a

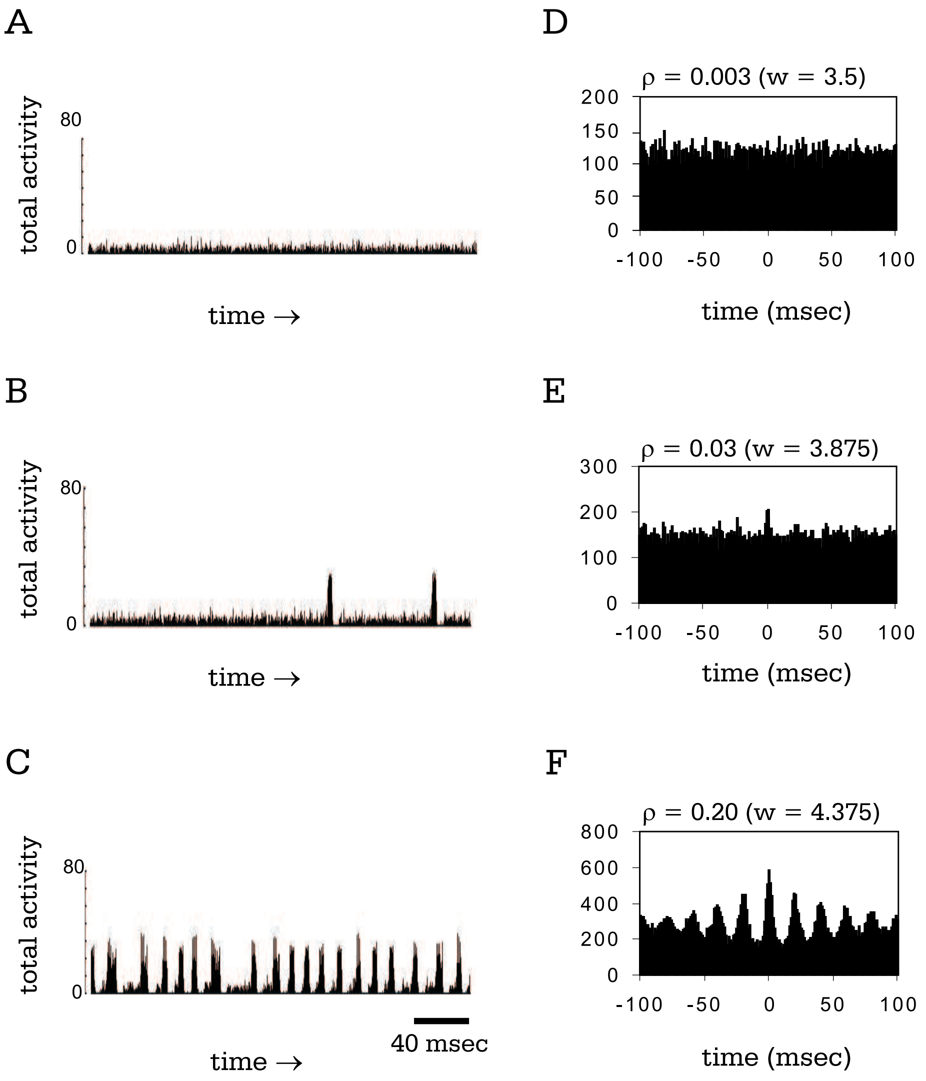


Figure 6.4: Network activity for 3 values of the synaptic weight, w_{int}. (A-C) Summed activity of the 150 neurons. The synaptic weight was 3.5 (A, $\rho = 0.003$), 3.875 (B, $\rho = 0.03$) and 4.375 (C, $\rho = 0.20$) Pair-wise correlations were determined for coincidences within a window of 2 msec. (D-F) Cross-correlation functions of 2 randomly selected neurons for the same synaptic weights as are plotted on the left. Even for a low pair-wise correlation ($\rho = 0.03$) highly synchronous network bursts occur, but these hardly show up in the cross-correlogram.

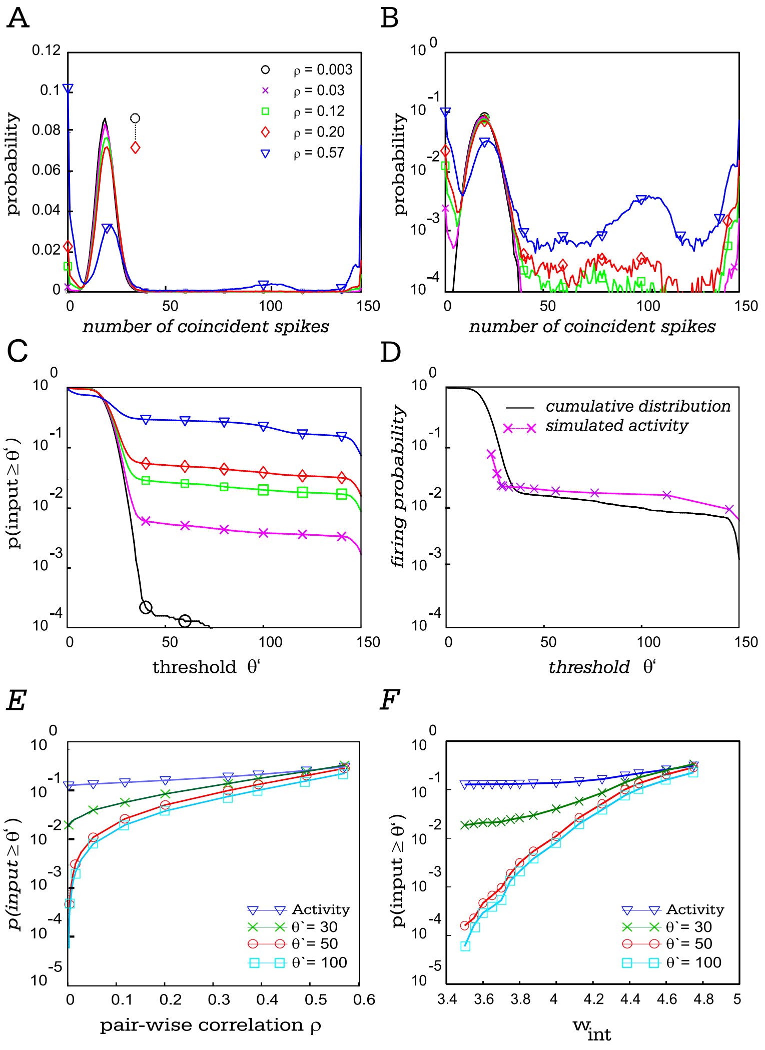


Figure 6.5: (A,B) Probability distribution of N-clusters, plotted on a normal scale (A) and a logarithmic scale (B). (C) Cumulative probability of N-clusters. Abscissa, cluster-size (θ'). Ordinate, probability of a cluster with a size that is equal to, or larger than θ'(D) Comparison of the cumulative N-cluster distribution (solid line) to the firing-probability of a post-synaptic neuron, which receives input from all 150 neurons in the network. The threshold θ' is defined as $\theta/w_{i,151}$. (E) Dependence of the firing probability of a neuron receiving input from all 150 neurons on its relative threshold θ' and the pair-wise correlation in the network. Different curves correspond to different values of θ'. Triangles, dependence of the average firing probability f_1 on the pair-wise correlation. (F) Same as (E), but the firing probability of the post-synaptic cell and f_1 are plotted as a function of w_{int}.

firing-threshold of m EPSP's the incidence of m or *more* coincident spikes is a much more significant quantity, since this determines its firing rate to a large extent. Therefore, we computed a *cumulative probability distribution*: the probability that m or *more* neurons fire synchronously. The cumulative probability distributions (the cumulatives of Fig. 6.5B) are plotted in figure 6.5C. Suppose that the membrane-potential of a post-synaptic neuron, which receives input from all cells in our network, equals the resting potential at time t. The probability that this neuron fires at time $t + 1$ is equal to the probability of θ' or more coincident spikes, where θ' is the ratio between the post-synaptic threshold (θ) and the EPSP amplitude. In other words, the curves in figure 6.5C illustrate the relation between the threshold and firing probability of a neuron, which receives input from all neurons in the network, in the case of a very short membrane time constant. The curve for a very low pair-wise correlation coefficient ($\rho = 0.003$) shows the quickly declining probability of higher order events. The curves for network activity with a larger pair-wise correlation exhibit a plateau before their decline at very high numbers of synchronized spikes. This plateau results from the second peak in the N-cluster probability distribution. It is remarkable that even small changes in the strength of the pair-wise correlations, which are not physiologically implausible (Gray et al., 1989; Livingston, 1996), exhibit a strong effect on the firing probability in case of an intermediate threshold θ'. Indeed, the firing probability of a neuron with a threshold of, say, 50 EPSP's is raised by more than an order of magnitude by an increase in the pair-wise correlation as small as 0.08 (compare the distributions for $\rho = 0.03$ and $\rho = 0.11$).

6.4.3 Firing-rate of a post-synaptic neuron The question remains how closely the firing-rate of a post-synaptic neuron with non-zero membrane time-constant is approximated by the cumulative probability distribution. There would obviously be a one-to-one relationship for a post-synaptic neuron that has "zero" memory, i.e. a neuron which is only influenced by the input in the previous time-step (the bin-size for which spikes are considered synchronous, as discussed above).

However, typical neurons have parameters with larger time constants, including the refractory period, and the time-constant of the membrane. In order to assess the impact of the "non-zero memory", an additional 151^{th} neuron was included in the network. This cell was identical to the other 150 neurons from which it received input, but it did not project back to them. The pair-wise correlation was fixed at 0.12 and the membrane-time constant was fixed to the same value as the other 150 neurons, thus realizing a non-zero memory post-synaptic neuron. Simulations were performed with different values of the post-synaptic threshold θ', by varying $w_{i,151}$ $(1 < i < 150)$, the strength of the synapses projecting onto neuron 151. The dependence of the firing probability on θ' $(\theta/w_{i,151})$ is shown in figure 6.5D. Superimposed on this graph is the distribution of N-clusters. As expected, some minor differences between the firing probability and the cumulative probability distribution are observed. Most of these differences can be explained, given the fact that the non-zero average activity in a pool of neurons keeps the membrane potential of individual neurons at a somewhat higher level than the resting potential. This lowers the average number of coincident spikes the neuron would require to cross threshold. On the whole however, the firing probability of the post-synaptic neuron with a non-zero membrane constant is predicted with good accuracy by the cumulative probability distribution. A remarkable feature of figure 6.5D is that the firing probability of a neuron with a threshold of for example 40 does not differ much from that of a neuron with a much higher threshold (e.g. 130). Most of the spikes of a post-synaptic cell with a threshold larger than 40 are triggered by synchronous bursts in which the majority of neurons participate. As these bursts are separated in time by about 15 msec, a relatively large number compared to the membrane time-constant, this result shows that regarding the firing rate of a post-synaptic neuron in our model, the actual time-structure in the cluster-distribution is of far less importance than the sheer number of bursts.

Using this interpretation of the cumulative probability distribution, the effect of an increase in the pair-wise correlation in the network on post-synaptic neurons was investigated by varying the synaptic weight (w_{int}). Figure 6.5E shows the relationship between the average pair-wise correlation and the firing probability of a hypothetical post-synaptic neuron with threshold θ'. Calculated were the firing-probabilities for thresholds θ' = 30, 50 and 100. It can be seen that in the biologically relevant range of pair-wise correlations $(\rho = 0 - 0.2)$, there is a monotonic relationship between the post-synaptic firing probability and the pair-wise correlation. For values of ρ below 0.2, changes in the pair-wise correlation are associated with a relatively large increase in the firing probability of the post-synaptic neuron. Importantly, higher values of ρ are also associated with an enhanced firing rate of the pre-synaptic network neurons (Fig. 6.5E). This increase in activity undoubtedly contributes to the enhanced probability of higher order events. However, it should be noted that the probability of higher order events exhibits a steeper dependence on ρ than the firing probability of the network neurons (Fig. 6.5E). Figure 6.5F illustrates the dependency of the firing probability of the post-synaptic neuron on w_{int}, the parameter that was actually varied during the simulations.

6.4.4 Estimation of the N-cluster distribution by entropy maximization

In most electrophysiological experiments only data is available on firing probabilities and pair-wise correlations. Using the mathematical framework developed in section 6.3, we investigated whether the observed relationship between the probability of N-clusters and the correlation coefficient could have been predicted from these two types of measurements. For the network behavior at different values of w_{int} (3.5, 3.875 and 4.125), distributions of N-clusters were calculated which maximized entropy under the constraints of the observed firing probability and pair-wise correlations. A comparison between distributions based on the maximal entropy calculation and the experimental distribution is shown in figure 6.6A-C. The distribution that maximized the entropy deviates somewhat from the observed distribution for larger values of ρ.The maximal entropy calculation underestimates the incidence of clusters between 60 and 100 spikes, which occur during start and end of population bursts, as was discussed above. In addition, the second peak in the maximal entropy distribution is located at a cluster size of 130, whereas the actual location of this peak is 150. Typically, all neurons fire within a 2 msec window during a network burst. In order to estimate the effect of these discrepancies on the firing probability of a neuron receiving input from the network, the cumulative probability distributions are plotted in figures 6.6D-F. It can be seen that the underestimation of the incidence of intermediate cluster-sizes by the maximal entropy calculation is compensated by the overestimation of the incidence of clusters with a size between 100 and 140. For values of the threshold θ' smaller than 130, the largest deviation is about a factor of 2. This approximation is reasonable, since the maximal entropy calculation depends on only 2 parameters, which are estimated from the network activity: the firing probability and the pair-wise correlation. Nevertheless, large deviations occur for values of θ' larger than 130. However, these high threshold values are physiologically implausible, because they would imply that a post-synaptic neuron would only fire when almost every input is active within a narrow time-window.

6.4.5 The effects of varying bin-width on the maximal entropy distribution.

The results of the maximal entropy calculation described so far,

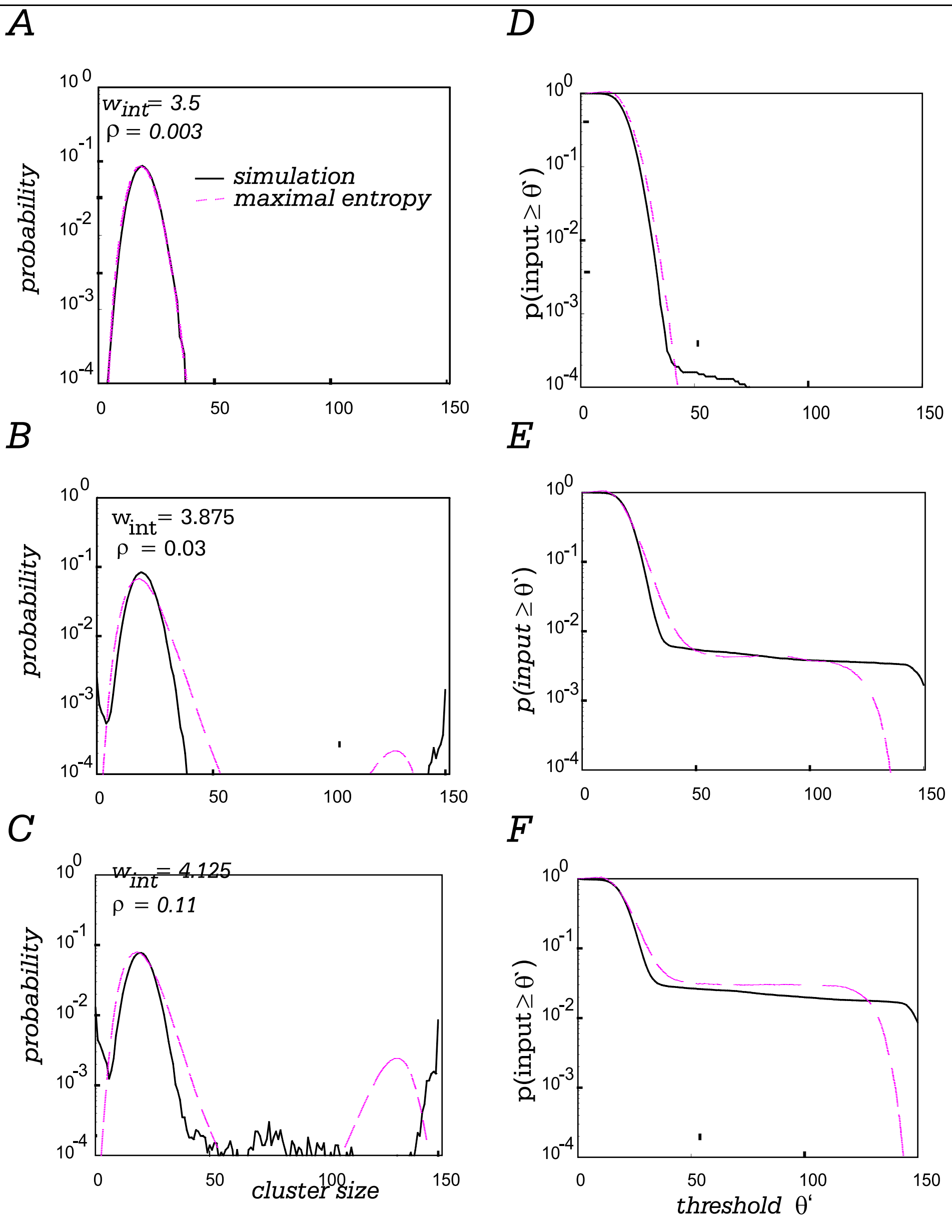


Figure 6.6: Comparison of the probability of N-clusters in the network simulation, and their probability based on maximal entropy. (A-C) Probabilities of N-clusters in the simulation are shown as solid curves for three values of the synaptic weights. Dashed line: the corresponding maximal entropy probability distributions with the same firing probability and pair-wise correlation. (D-F) Relationship between firing probability (ordinate) and threshold (abscissa) of a neuron with an integration window of 2 msec that receives input from all 150 neurons of the network (cumulatives of A-C).

were obtained with a bin-width of 2 msec, which equals the minimal inter-spike interval. Figure 6.7 shows the dependence of the N-cluster distribution and the maximal entropy estimation on the bin-width. Spike trains obtained with a synaptic weight (w_{ij}) of 4.125 were re-binned, using bin-widths of $0.5, 1.0, 2.0$ and 4.0 msec. The rebinning process influenced the pair-wise correlation, and f_1, the probability of firing in a time-bin (eq. 6.9). Smaller bin-widths reduce f_1. This results in a leftward shift in the location of the first peak in the distribution of cluster-sizes, which represents stochastic activity between bursts (compare Fig. 6.7A-B to Fig. 6.7C). A second effect of reducing the binwidth is a disappearance of clusters with sizes larger than 110. This disappearance is related to the refractory period, which prohibits neurons from firing in consecutive bins during population bursts. Spikes fired by different neurons during these bursts are therefore divided between successive time-bins, and no bins remain in which all neurons fire simultaneously. This modification of the distribution of cluster-sizes is not captured by the maximal entropy estimation, which underestimates the incidence of clusters with sizes between 20 and 110, and grossly overestimates the incidence of clusters with a size larger than 130 (Fig. 6.7A-B). In other words, the refractory period adds structure to the spike trains, which therefore have less than maximal entropy. When a bin-width is used that is longer than the refractory period, this additional structure is lost, and the maximal entropy calculation may provide a reasonable estimate of the distribution of cluster sizes.

For bin-sizes that are longer than the minimal inter-spike interval, there are time-windows, during which individual cells fire more than a single spike. It is possible, in principle, to adapt the maximal entropy estimation to this situation. One approach, in the case of a bin-size that may include 2 spikes, is to compute the probability that N neurons fire once, and M neurons fire twice in a bin, for each combination of N and M. Unfortunately, this increases the number of variables that should be calculated from 151 (D_0 to D_{150}) to more than 10.000. The computational requirements increase further if 3 or more spikes can occur in a single bin. Therefore, we took an alternative approach in which the state of a neuron was labeled "off", in the case of no spike, and labeled "on" in the case of one or more spikes within a time bin. This keeps the computational requirements within bounds, but at the cost of losing spikes in the process. Figure 6.7D compares the distribution of cluster sizes to the maximal entropy distribution for a bin-width of 4 msec. The experimental distribution is largely described by a broad first peak, which is shifted to the right, and a very narrow second peak at a cluster size of 150. An increase

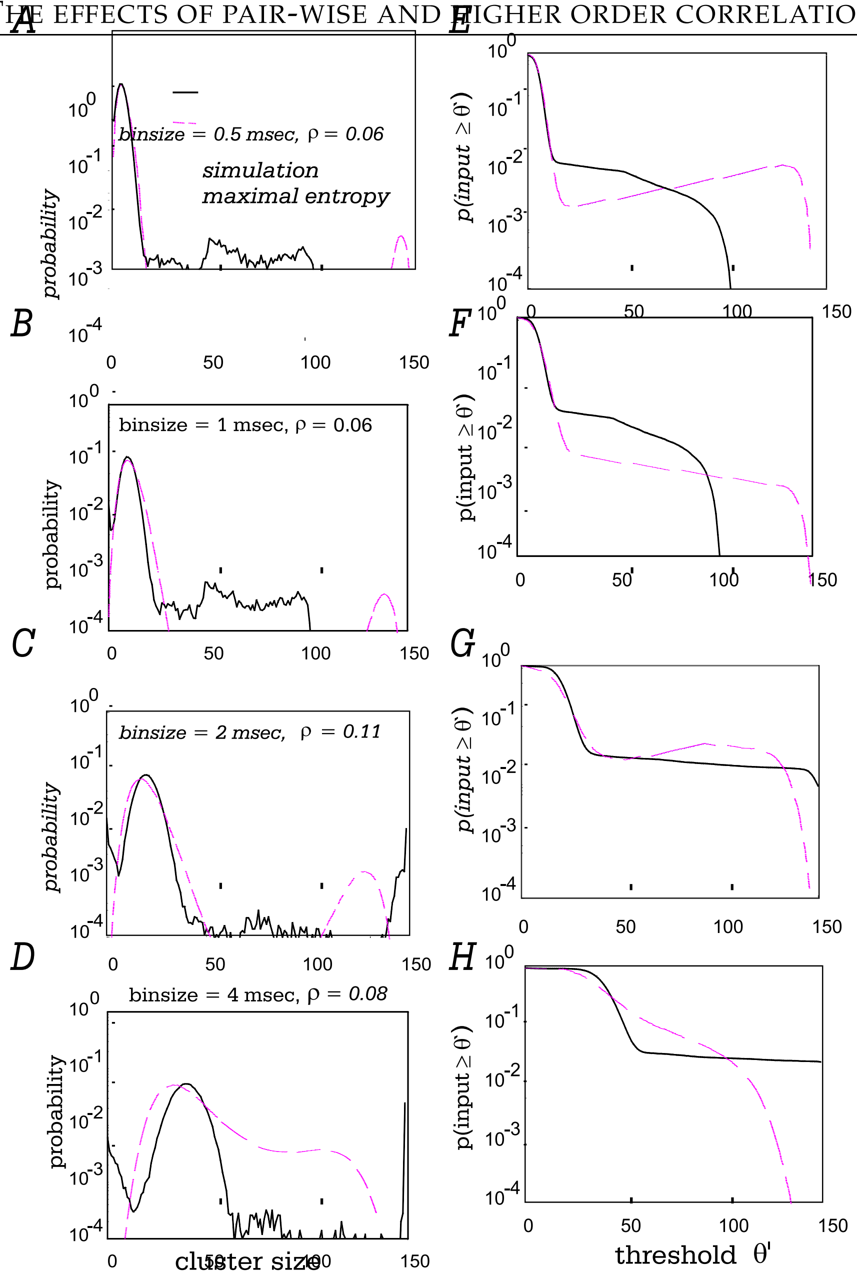


Figure 6.7: (A-D) Comparison of the maximal entropy distribution to the distribution of N-clusters in the simulation, for bin-sizes ranging from 0.5 msec to 4 msec. The quality of the maximal entropy approximation depends on bin-size, i.e. on what is considered coincident. In our simulations, a bin-size of 2 msec yields the best approximation (C). For smaller binsizes (A: 0,5 msec, B: 1 msec), the actual and predicted distributions deviate considerably. (E-H) The impact of these deviations on a post-synaptic neuron with threshold θ'.

in f_1 also shifts the first peak of maximal entropy distribution to the right. However, larger values of f_1 are also associated with a leftward shift of the second peak in the distribution of maximal entropy, as was discussed in section 6.3. This causes large deviations for cluster sizes between 60 and 120, the incidence of which is overestimated by the maximal entropy estimation. Moreover, the narrow peak at a cluster size of 150 is absent in the distribution of maximal entropy.

In summary, reasonable estimates of the N-cluster distribution are only obtained for a bin width, that is equal to the minimal interspike interval. In the discussion we will address the question whether these results can be generalized to other network architectures.

6.4.6 The effects of network scaling In order to investigate how the network behavior and the goodness of fit of the maximal entropy distribution depends on network size, simulations were run with networks composed of 75, 150 and 200 neurons. Figure 6.8A shows the cumulative distribution of N-clusters for a network of 75 neurons with a w_{int} of 3.55, which exhibited a pair-wise correlation of 0.07. It should be noted that w_{int} represents the sum of the synaptic weights w_{ij} of all inputs converging onto a single neuron (section 6.4.1). When the network size is doubled (fig. 6.8B), each cell receives input from twice as many neurons, and the strength of the individual synapses w_{ij} was reduced accordingly, in order to maintain a constant value of w_{int}. Nevertheless, a doubling of the network size resulted in a clear leftward shift of the distribution of N-clusters, and a reduction of the pair-wise correlation to 0.003. This indicates that a constant value of w_{int} is not sufficient to guarantee a qualitatively similar network behavior when the network size is increased. Indeed, a larger number of synapses with reduced weight impinging on a network unit results in a reduction of the fluctuations in the input, as long as the network is in the stochastic mode. This reduces the probability of bursts in the network. Tsodyks and Sejnowski (1995) have suggested that the variance in the input to network units may be kept approximately constant, by reducing the release probability of the synapses, rather than reducing their strength w_{ij}, when network size is increased. Figure 6.8C shows the cumulative distribution of N-clusters for a network of 150 neurons, in which the effective input strength was kept constant by reducing the release probability to 50%. The pair-wise correlation was 0.12, and the cumulative distribution of N-clusters was qualitatively similar to that of the smaller network, in accordance with the findings by Tsodyks and Sejnowski (1995). Figure

6.8D shows a similar result for a network composed of 200 neurons. In all cases the estimate based on maximal entropy was reasonable (dashed lines in Fig. 6.8A-D), which indicates that the quality of the maximal entropy calculation does not depend critically on the size of the network.

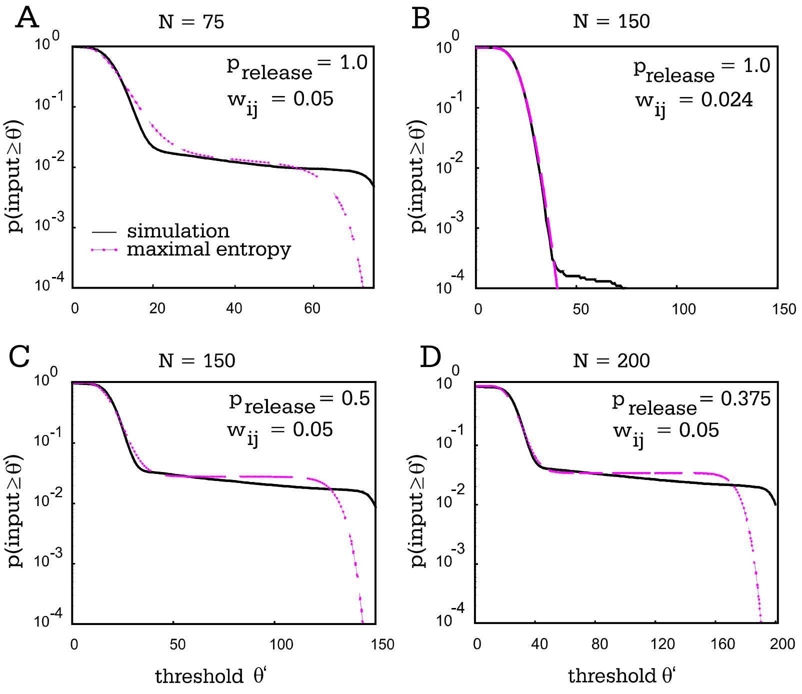


Figure 6.8: Effect of scaling the network size. (A) Continuous line, probability of N-clusters with a minimal size of θ', as a function of θ', for a network with 75 neurons. Dashed line, distribution of maximal entropy with the same firing probability and pair-wise correlation. Parameters of the simulation: $w_{ij} = 4.73 * 10^{-2}$, $\rho = 0.07$. (B) N-cluster distribution for a larger network with 150 neurons. The total input converging on a neuron, w_{int}, was kept constant by reducing w_{ij} to $2.37 * 10^{-2}$. Nevertheless, the distribution of N-clusters was shifted to the left, and the pair-wise correlation was reduced to 0.003. (C) Same as B, but the effective w_{int} was kept constant by reducing the release probability to 50% rather than by reducing w_{ij}. The synaptic strength w_{ij} was $4.73 * 10^{-2}$, and $\rho = 0.12$. (D) N-cluster distribution for a network of 200 neurons, with a ρ of 0.15. The synaptic weight w_{ij} was identical to that in (A), but release probability was reduced to 37.5%. Note the similarity of the distributions in (C) and (D) to that in (A).

6.5 Discussion

In most physiological studies on the synchronization behavior of cortical neurons, recordings are obtained from pairs of neurons, or pairs of cell clusters. The present results illustrate that these data can only supply limited information about the probability of higher order events, like the probability that, for example, 30 or more neurons project that to a target neuron fire simultaneously. As an approximation for the probability of higher order events, we used the distribution of maximal entropy. This method provides the most unstructured distribution, given the constraints supplied by the firing probabilities and the pair-wise correlations. The N-cluster distribution with maximal entropy for a homogeneous network without higher-order correlations exhibits two peaks. The first peak represents stochastic activity, and resembles a binomial distribution. This peak also occurs in the absence of correlations. If the firing probability is moderate (< 0.25), a second peak occurs at a relatively large cluster size. The magnitude of this second peak depends on the strength of the pair-wise correlation. Thus, an absence of *higher order* correlations dictates that the network should generate coincidences in a number of tightly synchronized network bursts. The N-cluster distribution observed in the simulations also exhibited two peaks, although the position of the second peak was different from the second peak in the maximal entropy distribution. Questions about the generality of these results, and in particular, about their dependence on the details of the network implementation will have to await further experimentation.

Previous studies on the impact of correlations among neurons that project to a common post-synaptic target have used N-cluster distributions with a drastically different shape (Bernander et al., 1994; Murthy & Fetz, 1994). In these studies correlations were introduced in the input by forcing a subset of the pre-synaptic neurons to fire in perfect synchrony, but independently of the other pre-synaptic neurons. Since the resulting N-cluster distribution is relatively devoid of highly synchronous events, and has less than maximal entropy, the generality of the results obtained in these earlier studies may also be limited.

It is obvious that network-bursts in which a large number of neurons participate are rather effective in driving a post-synaptic neuron, especially in the case of a high post-synaptic threshold. Indeed, a recent study (Alonso et al., 1996) uncovered tight correlations, with a peak-width in the cross-corellogram of less than 1 msec, among pairs of neurons in the LGN structure of the brain. In cases in which the LGN neurons projected to a common cortical neuron, the impact of synchronous events was stronger than that of asynchronous spikes. We observed an orderly relationship between the incidence of highly synchronous network bursts and the pair-wise correlation. Relatively small increases in the pair-wise correlation, which are not physiologically implausible, may raise the incidence of highly synchronous events, and thereby the firing rate of a post-synaptic cell, by more than an order of magnitude (Fig. 6.5C,E).

We will now discuss the limitations of the maximal entropy estimation, and the way in which these limitations may be overcome by future studies. First, the quality of the approximation by the distribution of maximal entropy exhibited a strong dependence on the choice of the binsize. Reasonable results were only obtained for binsizes larger than the refractory period of the neurons. If the binsize was smaller than the refractory period, structure was added to the N-cluster distribution, which therefore had less than maximal entropy. This limitation is a direct consequence of the way in which the entropy was defined. The entropy was determined by the distribution of N-clusters within individual time-bins, and was independent of the order of N-clusters in successive bins. The equivalent situation for a cross-correlation study would be to only calculate the center bin of the cross-correlation functions, i.e. only the probability that two neurons fire at exactly the same time. A possible extension of the method would be to reformulate entropy in order to include 2nd order correlations with a time delay (i.e. the non-central bins in the auto- and cross-correlation functions) in the calculation. We wish to remark, however, that such an extension is likely to result in a substantial increase in the number of variables and equations.

Second, the quality of approximation by the maximal entropy distribution was also degraded if a binwidth was chosen that was larger than the minimal interspike interval (larger than 2 msec in our simulations). In this situation, multiple spikes of a single neuron occurred within a single time bin. The membrane time constant of the postsynaptic neuron provides a natural temporal window over which input is integrated, i.e. the natural coincidence window. At present, the value of the effective membrane time constant in cortical neurons is a topic of considerable debate (Bernander et al., 1991; König et al., 1996; Shadlen & Newsome, 1995). A coincidence window of 2 msec, as we used, is presumably at the lower end of the physiologically plausible range. However, it seems likely that an increase in the width of the bin in which spikes are considered to be coincident to 5 or even 10 msec may be unproblematic in the case of cortical neurons.

Typical peak firing rates of cortical neurons are around 100 Hz, which implies that the probability of multiple spikes within a 5 or 10 msec bin will still be rather low. Without multiple spikes, it is easy to see that widening the time-window than merely corresponds to a simple re-scaling of the time-axis.

Alternatively, the calculation of the maximal entropy may be adapted to allow for multiple spikes in a single bin, as was discussed in section 6.4.5. If a very large binsize is chosen, the distribution of the number of spikes fired by a single neuron in a bin may approach a normal distribution. In this case it is relatively easy to calculate the distribution of N-clusters if the only correlations are of second order.

Third, the method according to which we derived the distribution of maximal entropy is only valid for a homogeneous network. For a non-homogeneous network, the number of variables grows exponentially with the number of neurons (section 6.3). We have, for example, incorporated a single inhibitory neuron in our network, which received input from all excitatory cells and provided strong inhibitory feedback (data not shown). The inhibitory neuron added structure to the spike trains by curtailing population bursts. We did not explicitly incorporate the firing pattern of the inhibitory neuron in our calculations, which caused a considerable discrepancy between the maximal entropy distribution and the actual distribution of N-clusters.

These limitations, taken together, imply that it will be rather problematic to predict the probability of highly synchronous events from firing rates and pair-wise correlations in physiological experiments. In a physiological study on higher order events among neurons of the frontal cortex, Martignon et al. (1995) obtained discrepancies between the probability of actually occurring events and their probability predicted by entropy maximization. Unfortunately, even in this study, in which simultaneous recordings from 6 neurons were studied, sampling problems occurred which were caused by the exponentially growing number of spike configurations and the limited recording time available during such experiments. Another, presumably fruitful way in which the probability of highly synchronous events can be estimated is by direct, in vivo measurements of the distribution of the post-synaptic potential. A comparison of the post-synaptic potential to the size of individual excitatory PSPs, with and without blockade of inhibition, could provide valuable insight in the degree of synchronicity among neurons that project to a common target cell.

6.6 Conclusion

We conclude that, given the available information, the *value* of the pair-wise correlation between neurons can most likely be explained in terms of the likelihood of tightly synchronized population-bursts. This view is reinforced by observations of uncorrelated "noisy" neural activity intermittent with sharp synchronization as reported in several experimental studies (Vaadia & Aertsen, 1992; Riehle et al., 1997). Related theoretical studies have also supported this finding as a principal property of densely connected networks of spiking neurons (Deppisch et al., 1993; Tsodyks & Sejnowski, 1995; Tsodyks et al., 2000). In line with these findings, a recent study by Steinmetz et al. (2000) reports a distinct relation between synchronous activity between pairs of neurons, as reflected in an increased pair-wise correlation, and increased firing-rates. These results suggest that synchrony as measured via cross-correlations is a local network property of densely connected neuronal networks. Lamme and Spekreijse (1998) for instance have found that synchrony between neurons in early visual cortex, does not represent texture segregation, at least in this stage. Rather, the correlations found between neurons seem to reflect local connectivity.

These facts have contributed to the increasing criticism on the "synchrony hypothesis", also from theoretical considerations (Shadlen & Movshon, 1999). However, the flurry of research into the specific temporal properties of the spikes emitted by neurons has uncovered a remarkable precision in the timing of these spikes in more than a few cases, and it now seems highly plausible that neurons do transmit information in the timing of single spikes (Volgushev & Eysel, 2000). The form of this information processing however remains unclear.

7

THE BIOLOGY OF SPIKING NEURONS

This chapter provides a background into the biological workings of spiking neurons, and a (non-exhaustive) summary of recent biological findings regarding the behavioral relevancy of the precise timing with which real spiking neurons emit spikes. The literature suggests that in almost any system where the processing-speed of a neural (sub)-system is required to be high, the timing of single spikes can be very precise and reliable. This line of evidence thus provides additional motivation for researching the computational properties of networks of spiking neurons that compute with precisely timed spikes.

7.1 Real Neurons Spike

A description of the full workings of a neuron, as far as they have been uncovered, would extend even beyond a full book on its own, though a good start might be the reference work "Principles of Neural Science" (Kandel, Schwartz, & Jessell, 1991). In this section, a brief outline of the neuron-features that are used in this thesis is given.

The human brain consists of a vast number of neurons: estimates put it between 10 and 100 billion. These neurons are spread over a number of anatomically different structures, like brain-stem, cerebellum, and cortex. Within each of these structures, different types of neurons exist, with different connectivity-patterns, different typical responses to activity on the inputs, and ultimately different tasks. What all (at least most) of these neu-

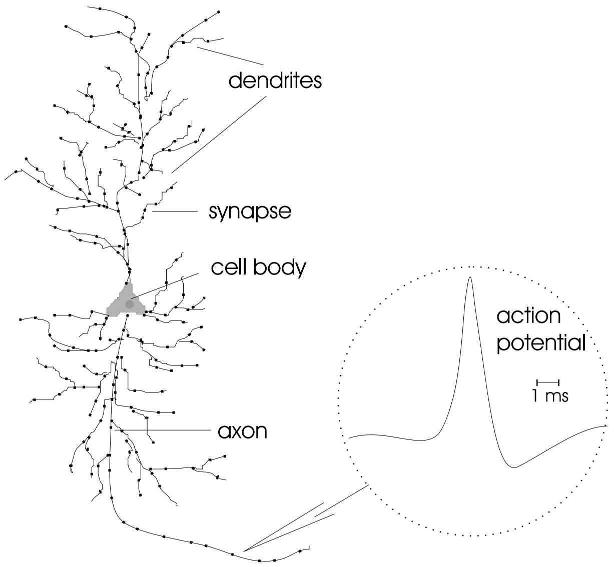


Figure 7.1: Outline of a single neuron, with dendritic and axonal tree fanning out from the cell body (after Gerstner (1998)).

rons share, is a network of connections to and from other neurons. The *dendritic* network of a neuron receives input from other neurons, and its *axonal* network delivers output to the dendritic networks of other neurons (depicted in figure 7.1). The sheer scale of the connectivity of a single neuron is impressive: it is estimated that a single cortical neuron is typically connected to 1000-10000 other neurons (Douglas & Martin, 1991).

A neuron communicates with other neurons (mostly) through the generation of *action potentials*, or *spikes* (inset figure 7.1). Roughly speaking, a neuron generates a spike when its internal state, as determined by its *membrane potential*, reaches some threshold. Connections between neurons are made at *synaptic terminals*. At such a site, the output spike of a *pre*synaptic neuron triggers the release of neurotransmitter, which in turn mediates a change in the membrane potential of the target (*post*synaptic) neuron. This change in the membrane potential of the postsynaptic neuron can be described by the *Postsynaptic Potential* (PSP), and may be negative, for *inhibitory* inputs (as these inputs inhibit the postsynaptic neuron from spiking), or *excitatory* (as they excite the neuron into spiking). The magnitude of the postsynaptic potential depends on the strength of the synaptic terminal. Input-spikes thus result in changes of the membrane potential that are effectively described as the integration of the postsynaptic potentials by the neuron. Since the contribution of a postsynaptic potential is limited

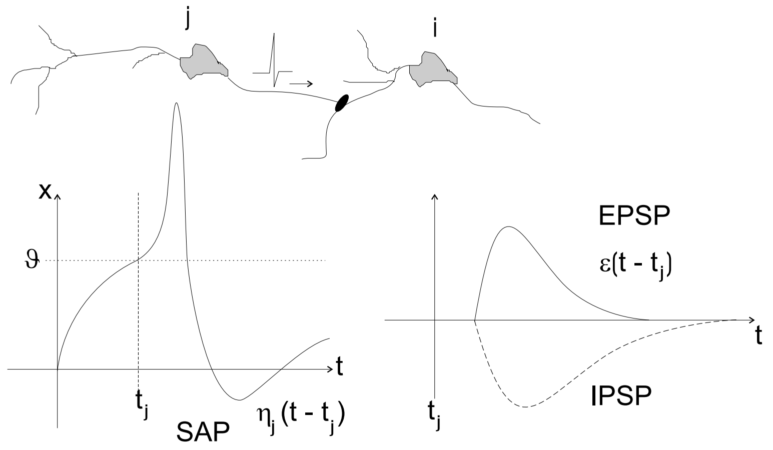


Figure 7.2: Generation and transmission of spikes: a neuron j generates an action potential when the membrane potential x crosses the threshold ϑ at time t_j. After generating a spike, the membrane-potential of neuron j is reset by the spike-after potential (SAP), described by $\eta(t - t_j)$. The spike evokes a response at the postsynaptic neuron i described by the spike-response function $\varepsilon_{ij}(t - t_j)$. The time-course of the voltage response to inhibitory or excitatory synapses is described by the IPSP and EPSP, respectively (after Gerstner (1998)).

in time and decays with some time-constant, neurons that respond in this manner to input are referred to as *leaky-integrate-and-fire* neurons. The general idea of the generation and transmission of spikes is depicted in figure 7.2.

The picture is somewhat complicated by the fact that spikes may arrive at different places on the dendritic tree. The effect of input arriving far or near to the cell-body is complicated, and differs for different types of neurons. This is very much an area of current neurophysiological research (e.g. Williams and Stuart (2002)), and we will not consider this in more detail, other than observing that the distance to the cell-body can for instance be translated into a delay between the arrival of the input and the actual effect that it has on the membrane potential of the neuron.

7.2 Precision and Reliability of Real Spikes

An important part of this thesis is concerned with neural networks of spiking neurons that compute with precisely timed action-potentials. In this section, we examine recent biological studies into the relevance of single spike-timing for the information processing in real neural systems. This section requires a rather intimate knowledge of the concepts and jargon of neuroscience; a short summary of the main findings can be found in the Introduction (chapter 1).

Whether or not real spiking neurons – or neural systems is general – exhibit a fine temporal precision in their action-potentials is a fundamental question that is much debated but has so far remained unresolved (Bialek, Rieke, Steveninck, & Warland, 1991; Softky & Koch, 1993; Shadlen & Newsome, 1994, 1995; Softky, 1995; Hopfield, 1995; Shadlen & Newsome, 1998; Mainen & Sejnowski, 1995; Laurent, 1999; Singer, 1999; Gray, 1999; Shadlen & Movshon, 1999). Neurophysiological experiments with *in vitro* cultures of neural tissue have shown that the integration process of individual cortical neurons is in principle reliable enough to support precise spike-time coding (Mainen & Sejnowski, 1995), and the brief summary in this section shows some of the evidence for precise and reliable spiking by real neurons that recent *in vivo* experiments have uncovered.

When it comes to real, behaving animals, there are a few specialized subsystems for which the relevance of temporal information in the spike-times has been clearly demonstrated. Prominent examples are the electrosensory system of the electric fish (Heiligenberg, 1991), the echolocation-system of bats (Kuwabara & Suga, 1993), and the auditory system of barn-owls (Carr & Konishi, 1990). In each of these instances, the relative timing of input spikes is used for the computation of the relative direction of the respective sources (AgmonSnir, Carr, & Rinzel, 1998). In the olfactory system of the locust, Laurent *et al.* have found that odor-recognition is associated with increased spike-time precision (Stopfer & Laurent, 1999), and that modification of the fine temporal structure of spike-trains disrupts the correct odor-classification (Laurent, 1999).

De Ruyter van Steveninck and Bialek *et al.* have extensively studied the time-response of the blowfly's motion sensitive neurone H1 during flight (Bialek et al., 1991; de Ryter van Steveninck, Lewen, Strong, Koberle, & Bialek, 1997; de Ruyter van Steveninck et al., 2001). Bialek *et al.* have developed a decoding method for reconstructing the (known) original environment as experienced by the blowfly (the input) from the signals measured from the H1 neurone. They find that when decoding the signal of H1, individual spikes contribute significantly to their velocity estimate at each point in time. The accuracy of the decoded signal was found to increase as the spikes were observed with greater temporal precision. In fact, patterns of spikes that differ only by millisecond shifts of the individual spikes could correspond to distinguishable velocity waveforms (Brenner, Strong, Koberle, Bialek, & Steveninck, 2000). Furthermore, the timing relationships between neural responses and stimulus events were found to be preserved with millisecond precision, even as signals pass through four stages of neural circuitry. A particular spike, distinguishable in a certain sequence, proved impressively reproducible, with a standard deviation on the onset time of 0.78ms. The spike directly following is reported to be even more precise, with a standard deviation of only 0.18ms (de Ruyter van Steveninck et al., 2001). They remark that the natural environment of the fly while airborne is characterized by signals with high temporal frequencies, and too cope, it seems that the fly's neural systems is efficient up to the physical limits of neural spike-timing.

Summarizing these findings, Fairhall et al. (2001) suggest that neurons in the fly's visual system employ a multilayered coding scheme, where the emitted spikes convey information in several ways: "the timing of individual spikes or short spike-patterns encodes stimulus features that are normalized to the stimulus ensemble, the statistics of interspike intervals on slightly longer timescales encode the stimulus ensemble, and the spike rate can carry information about the changes in ensemble on yet longer timescales."(from Fairhall et al. (2001)).

In a study of the cat's LGN, Liu et al. (2001) found neural responses that were highly reproducible in their spike timing (± 1-2ms). They further remark: "This degree of precision only became apparent when an adequate length of the stimulus sequence was specified to determine the neural response, emphasizing that the variable relevant to a cell's response must be controlled to observe the cell's intrinsic response precision." Bair and Koch (1996) report that the response of neurons in area MT of the macaque to a repeated stimulus was replicable with high temporal precision ($<$ 2ms), but only under certain conditions. Buracas, Zador, DeWeese, and Albright (1998) confirm and extend this finding, and suggest that the high temporal precision can be attributed to the fine temporal structure of the stimulus. The spike-time precision is then of the same magnitude as observed in the rabbit retina (Liu et al., 1997). A study by Beierholm, Nielsen, Ryge, Alstrom, and Kiehn (2001) adds to this theme: for generated by spinal neurons from the neonatal rat spinal cord, they report a reliable precision of singe spike in the order of 2–3ms, but only when the input into these cell's has a high frequency and amplitudes (3-30 Hz, 30-200 pA). They attribute this somewhat worse temporal precision of spinal neurons as compared to cortical neurons to the broad spikes in the former. Fellous et al. (2001) also report that spike-time reliability is highly dependent on the input frequency, with highest reliability obtained in the frequency range of the typical membrane potential oscillations (4-20 Hz for pyramidal cells in the rat's cortex).

Considerable research is uncovering the nature of the relationship between firing rate and the precise timing of spikes: in particular in the hippocampus the so-called phase-precession phenomenon is observed, i.e. (O'Keefe & Recce, 1993). Here, the timing of the pyramidal cell spikes relative to the theta rhythm is related to the instantaneous firing-rate (and hence position) of the neuron. Roughly, increasing firing-rate correlates with earlier spikes (relative to the phase of the theta rhythm. The theta rhythm is an oscillation in the local field potential at about 8Hz. The precision with which spikes are emitted relative to the theta-phase is remarkable: (Harris et al., 2002) report that in their experiments, mean phase, averaged over all behaviors, shifted from $295^\circ \pm 7^\circ$ at low instantaneous firing rates (1 spike per 2 cycles, $\approx$ 4Hz) to $141^\circ \pm 9^\circ$ at high instantaneous firing rates ($\geq$ 10 spikes per 2 cycles, $\approx$ 40Hz). Since the mean is taken as the circular mean $\pm 95\%$ confidence interval, this means that 95% of the spikes are fired within 2ms and 3.5ms respectively to the average phase. It is suggested that the phase-precession phenomenon is an effective means for transforming a rate-code into a temporal code: the proposed underlying mechanism by Mehta, Lee, and Wilson (2002) would faithfully reproduce the temporal order of activation of neurons on short ($<$ 10ms) time scales. The idea is that this would allow temporal sequence learning by compressing the relevant temporal order of events occurring on long time scales ($>$1000ms), into short times scales ($\approx$10ms) relevant for synaptic plasticity mechanisms, such as spike-time dependent plasticity (see below). Neural networks inspired by such ideas have been proposed by Hopfield and Brody (2000) for temporal sequence learning.

For learning, that is, for adaptively changing synaptic strengths, it has been well established that the temporal order of pre- and postsynaptic spikes can determine synaptic potentiation vs. depression (LTP vs. LTD). A presynaptic spike arriving *before* a postsynaptic spike is generated tends to lead to potentation of the particular connection; arrival *after* the postsynaptic spike leads to synaptic depression (Markram, Lübke, Frotscher, & Sakmann, 1997). Further investigations have revealed that the time-window for such order-dependent synaptic modification is about 20ms either way: the arrival of relatively earlier or later presynaptic spikes does not result in LTP or LTD (Bell, Han, Sugawara, & Grant, 1997; Zhang, Tao, Holt, Harris, & Poo, 1998; Feldman, Nicoll, Malenka, & Isaac, 1998; Bi & Poo, 1998). This phenomenon has become known as spike-time-dependent-plasticity (STDP).

A learning rule similar to STDP was already proposed by Gerstner et al. (1996) as a means for modeling the learning process that leads to the delay-selection for fine temporal processing by neurons in the auditory system of the barn-owl (as reported by Carr and Konishi (1990)). Such delay selection has also been reported in cultured hippocampal neurons, presumably allowing the conversion of temporal information coded in the timing of individual spikes into and stored as spatially distributed patterns of persistent modifications in a neural network (Bi & Poo, 1999).

In a study of the macaque's primary auditory cortex, deCharms and Merzenich (1996) report that populations of neurons can coordinate the relative timing of their action potentials such that spikes occur closer together in time during continuous stimuli. This way, the neurons can signal stimuli even when their firing rates do not change. deCharms and Merzenich argue that in this fashion, population coding based on relative spike-timing has a number of beneficial properties: the population can systematically signal stimulus features; the signal is topographically mapped; and the signal follows the stimulus time-course even when the mean firing rate does not. They also report similar observations in the awake animal and in the primary somatosensory cortex.

The fact that the rate at which neurons fire does not fully capture the information content conveyed in the spike-train is becoming increasingly well established. This also extends to the idea of population rate-codes like the pooled-response code: this is the proposed means for rapidly transmitting changes in firing-rate via the neural population response, e.g. (Gerstner, 2000). Reich, Mechler, and Victor (2001) recently showed explicitly that the pooled-response code carries significantly less information as compared to the code that keeps track of when which neuron emitted each signal. A significant part of the information is thus transmitted by means other than the firing-rate.

Taking all these data together, it seems clear that the fine temporal structure of impinging spikes, and their impact on the membrane potential, can carry important information, rather than being just noise. It is worth noting that this view is now also being voiced in the "perspective" sections of leading journals like Science and Nature (Volgushev & Eysel, 2000; Helmuth, 2001; Richmond, 2001). Intriguingly, Williams and Stuart (2002) recently reported that distal EPSPs are ineffective sources of background somatic excitation, but, through coincidence detection on the millisecond-scale, have a powerful transient signaling role. The traditional idea that membrane fluctuations should mostly be attributed to the impact of background excitation is thus at least partially challenged.

It is worth noting that the excitement about spiking neural networks did not start with the proposed computational function by Von der Malsburg (1981), but rather by the report of the predicted synchrony – or rather: correlations – in the monkey brain by Gray et al. (1989). Although the functional relevance of these correlations in terms of dynamic feature binding are now being questioned, these findings, and the availability of ever more sophisticated equipment, have encouraged a close examination of the significance of single spikes in neuronal systems. As demonstrated by the small sample outlined above, there are most certainly cases where the timing of single spikes is important. This seems particularly true when the fine temporal structure of the relevant stimuli encountered in the environment is high (30-300ms). When neural systems that have to process information from such a fast environment are studied in a much slower one, the temporal precision of single spikes is quickly lost, e.g. (de Ruyter van Steveninck et al., 2001).

The important open question is whether or not neurons in the (human) cortex encode information in the precise timing of spikes so far remains an unresolved question. The prediction that seems to emerge from the collected evidence would be that fast systems use fast spike-time coding schemes. Tellingly, the human visual system has been shown to be capable of performing very fast classification (Thorpe et al., 1996), where a participating neuron can essentially fire at most one spike. On the time scales involved, the relevant input thus consists of at most one spike, and, as shown by Stratford, Tarczy-Hornoch, Martin, Bannister, and Jack (1996), even when successive pairs of spikes are transmitted from LGN into the visual cortex, only one spike has a measurable impact on a spiky stellate neuron. The speed involved in decoding auditory information, and even the generation of speech also suggest that most crucial neural systems of the human brain operate quite *fast*.

PUBLICATIONS

Material in this thesis has been published in several journals and conference proceedings.

- Chapter 2 corresponds to the article "*Unsupervised Clustering with Spiking Neurons by Sparse Temporal Coding and Multi-Layer RBF Networks*", by Sander M. Bohte, Joost N. Kok and Han La Poutré, and appeared in **IEEE Transactions on Neural Networks**, 2002, 13(2), 426-435 (Bohte et al., 2002c), as well as CWI Technical Report SEN-R0037 (Bohte, Kok, & La Poutré, 2000c). A short paper on this work has appeared in the proceedings of the IJCNN 2000 conference as "*Unsupervised classification in a layered network of spiking neurons*", 2000, Como, Italy (Bohte, Kok, & La Poutré, 2000d). An abstract has been presented at the 11th BNAIC, 1999 (Bohte, La Poutré, & Kok, 1999).
- The results presented in Chapter 3 have been published as "*Error-Backpropagation in Temporally Encoded Networks of Spiking Neurons*", by Sander M. Bohte, Joost N. Kok and Han La Poutré, in **Neurocomputing**, 2002, 48(1–4), 17–37 (Bohte et al., 2002b). An early version appeared as CWI Technical Report SEN-R0036 (Bohte, Kok, & La Poutré, 2000a). A short version has appeared in the proceedings of ESANN 2000, as "*Spike-prop: error-backpropagation in multilayer networks of spiking neurons*", pp 419–425, Bruge, Belgium (Bohte, Kok, & La Poutré, 2000b). Abstracts of (parts of) the work have been presented at the 12th BNAIC, (Bohte, La Poutré, & Kok, 2000), and the Fifth International Conference on Cognitive and Neural Systems (Bohte, La Poutré, & Kok, 2001).
- Chapter 4 contains material published as "*Implementing Position-invariant Detection of Feature-conjunctions in a Network of Spiking Neurons*", presented at the IJCNN/WCCI'2002 conference, authored by Sander M. Bohte, Joost N. Kok and Han La Poutré (Bohte, Kok, & La Poutré, 2002a). An early abstract of this work has been presented at the International Workshop on Dynamical Neural Networks and Applications 2001 (Bohte, Kok, & La Poutré, 2001).
- Chapter 5 contains material published in "*Modeling Efficient Conjunc-*

tion Detection with Spiking Neural Networks", presented at the ESANN 2002 conference, authored by Sander M. Bohte, Joost N. Kok and Han La Poutré (Bohte, Kok, & La Poutré, 2002).

- The results presented in Chapter 6 have been published in the article *"The effects of pair-wise and higher order correlations on the firing rate of a post-synaptic neuron"*, by Sander M. Bohte, Henk Spekreijse, and Pieter R. Roelfsema, **Neural Computation**, 2000, 12(1), 153-179 (Bohte, Spekreijse, & Roelfsema, 2000). An abstract has been presented at the International Workshop on Neural Coding at the Hanse-Wissenschaftskolleg, Delmenhorst, 1999 (Bohte, Spekreijse, & Roelfsema, 1999).

Not included in this thesis is joint work with Dr. Pieter Roelfsema on the detection of connectedness with neural networks (Roelfsema, Bohte, & Spekreijse, 1999), and publications on adaptive software agents operating in an electronic market place:

- *"Competitive Market-based Allocation of Consumer Attention Space"*, by Sander M. Bohte, Enrico H. Gerding and Han La Poutré. Proceedings of the 3rd ACM Conference on Electronic Commerce (EC-01), 2001, pp 202-206 (Bohte, Gerding, & Poutré, 2001). An extended version of this paper has appeared as CWI Technical Report SEN-R0131 (Bohte, Gerding, & La Poutré, 2001), and has been submitted for journal publication. An abstract has been presented at the 13th BNAIC (Bohte, Gerding, & La Poutré, 2002). A joint CWI/KPN patent has been applied for on the mechanism developed (Bohte, Gerding, La Poutré, Driessen, & Bomhof, 2001). Related joint work has been published in the proceedings of the fourth workshop on Agent Mediated Commerce (AMEC) ('t Hoen, Bohte, Gerding, & La Poutré, 2002a), and an extended version has appeared as CWI Technical Report SEN-R0217 ('t Hoen, Bohte, Gerding, & La Poutré, 2002b).

BIBLIOGRAPHY

Abbott, L., & Sejnowski, T. (Eds.). (1999). *Neural codes and distributed representations.* MIT Press.

Abeles, M., Bergman, H., Margalit, E., & Vaadia, E. (1993). Spatiotemporal firing patterns in the frontal cortex of behaving monkeys. *J. Neurophysiol.*, *70*, 1629-1658.

AgmonSnir, H., Carr, C., & Rinzel, J. (1998). The role of dendrites in auditory coincidence detection. *Nature*, *393*, 268-272.

Alonso, J.-M., Usrey, W., & Reid, R. (1996). Precisely correlated firing in cells of the lateral geniculate nucleus. *Nature*, *383*, 815-819.

Bair, W., & Koch, C. (1996). Temporal precision of spike trains in extrastriate cortex of the behaving macaque monkey. *Neural Computation*, *8*(6), 1185-1202.

Bair, W., Koch, C., Newsome, W., & Britten, K. (1994). Power spectrum analysis of bursting cells in area mt in the behaving monkey. *J. Neurosci.*, *14*, 2870-2892.

Baldi, P., & Heiligenberg, W. (1988). How sensory maps could enhance resolution through ordered arrangements of broadly tuned receivers. *Biol. Cybern.*, *59*, 313-318.

Beierholm, U., Nielsen, C., Ryge, J., Alstrom, P., & Kiehn, O. (2001). Characterization of reliability of spike timing in spinal interneurons during oscillating inputs. *J. Neurophysiol.*, *86*, 1858-1868.

Bell, C., Han, V., Sugawara, Y., & Grant, K. (1997). Synaptic plasticity in a cerebellum-like structure depends on temporal order. *Nature*, *387*, 278–281.

Bernander, O., Douglas, R., Martin, K., & Koch, C. (1991). Synaptic background activity influences spatiotemporal integration in single pyramidal cells. *Proc. Natl. Acad. Sci. USA*, *88*, 11569-11573.

Bernander, O., Koch, C., & Usher, M. (1994). The effect of synchronized inputs at the single neuron level [Letter]. *Neural Computation*, *6*(4), 622–641.

Bi, B.-q., & Poo, M.-m. (1999). Distributed synaptic modification in neural networks induced by patterned stimulation. *Nature, 401*, 792-716.

Bi, G.-q., & Poo, M.-m. (1998). Synaptic modifications in cultured hippocampal neurons: Dependence on spike timing, synaptic strength, and postsynaptic cell type. *J. Neurosci., 18*(24), 10464–10472.

Bialek, W., Rieke, F., Steveninck, R. de Ruyter van, & Warland, D. (1991). Reading a neural code. *Science, 252*, 1854-1857.

Bishop, C. M. (1995). *Neural networks for pattern recognition.* Clarendon Press, Oxford.

Bohte, S. M., Gerding, E., & La Poutré, H. (2001). *Competitive Market-based Allocation of Consumer Attention Space* (Tech. Rep.). CWI Technical Report SEN-R0131, www.cwi.nl/∼sbohte.

Bohte, S. M., Gerding, E., & La Poutré, H. (2002). Competitive market-based allocation of consumer attention space. In *Proceedings of the Thirteenth Belgium-Netherlands Artificial Intelligence Conference (BNAIC).*

Bohte, S. M., Gerding, E., La Poutré, H., Driessen, C., & Bomhof, F. (2001, July). *Allocation of Consumer Attention Spaces via Competitive Market-based mechanisms.* Patent no. 75236.

Bohte, S. M., Gerding, E. H., & Poutré, H. L. (2001). Competitive Market-based Allocation of Consumer Attention Space. In M. Wellman (Ed.), *Proceedings of the 3rd ACM conference on Electronic Commerce (EC-01)* (p. 202-206). The ACM Press.

Bohte, S. M., Kok, J., & La Poutré, H. (2002). Modeling efficient conjunction detection with spiking neural networks. In M. Verleysen (Ed.), *Proceedings of the European Symposium on Artificial Neural Networks (ESANN) 2002* (p. 263-268). D-Facto.

Bohte, S. M., Kok, J. N., & La Poutré, H. (2000a). *Spike-prop: error-backpropagation in multi-layer networks of spiking neurons.* (Tech. Rep.). CWI Technical Report SEN-R0036, www.cwi.nl/∼sbohte.

Bohte, S. M., Kok, J. N., & La Poutré, H. (2000b). Spike-prop: error-backpropagation in multi-layer networks of spiking neurons. In M. Verleysen (Ed.), *Proceedings of the European Symposium on Artificial Neural Networks (ESANN) 2000* (p. 419-425). D-Facto.

Bohte, S. M., Kok, J. N., & La Poutré, H. (2000c). *Unsupervised classification in a layered network of spiking neurons.* (Tech. Rep.). CWI Technical Report SEN-R0037, www.cwi.nl/∼sbohte.

Bohte, S. M., Kok, J. N., & La Poutré, H. (2000d). Unsupervised classification in a layered network of spiking neurons. In *Proceedings of International Joint Conference on Neural Networks (IJCNN) 2000* (pp. 249–285).

Bohte, S. M., Kok, J. N., & La Poutré, H. (2001). Rapid dynamic feature binding in feedforward spiking neural vector networks. In *Proceed-*

ings of the International Workshop on Dynamical Neural Networks and Applications. (pp. 84–86).

Bohte, S. M., Kok, J. N., & La Poutré, H. (2002a). Implementations of efficient conjunction detection in spiking neural networks. In *Proceedings of International Joint Conference on Neural Networks (IJCNN) 2002* (p. 1097-1103).

Bohte, S. M., Kok, J. N., & La Poutré, H. (2002b). Spike-prop: error-backpropagation in multi-layer networks of spiking neurons. *Neurocomputing*, *48*(1–4), 17–37.

Bohte, S. M., Kok, J. N., & La Poutré, H. (2002c). Unsupervised classification in a layered RBF network of spiking neurons. *IEEE Trans. Neural Networks*, 426–435.

Bohte, S. M., La Poutré, H., & Kok, J. N. (1999). Unsupervised classification in a layered rbf network of spiking neurons. In E. Postma & M. Gyssens (Eds.), *Proceedings of the Eleventh Belgium-Netherlands Artificial Intelligence Conference (BNAIC)* (p. 257).

Bohte, S. M., La Poutré, H., & Kok, J. N. (2000). Spike-prop: backpropagation for networks of spiking neurons. In A. van den Bosch & H. Weigand (Eds.), *Proceedings of the Twelfth Belgium-Netherlands Artificial Intelligence Conference (BNAIC)* (p. 321).

Bohte, S. M., La Poutré, H., & Kok, J. N. (2001). Learning in spike-time encoded neural networks. In *Proceedings of the Fifth International Conference on Cognitive and Neural Systems.* (pp. 31–32).

Bohte, S. M., Spekreijse, H., & Roelfsema, P. R. (1999). The effects of pair-wise and higher-order correlation on the firing rate of a post-synaptic neuron. In K. Pawelzik (Ed.), *Proceedings of the International Workshop on Neural Coding.*

Bohte, S. M., Spekreijse, H., & Roelfsema, P. R. (2000). The effects of pair-wise and higher-order correlation on the firing rate of a post-synaptic neuron. *Neural Computation*, *12*(1), 135-159.

Brenner, N., Strong, S. P., Koberle, R., Bialek, W., & Steveninck, R. R. de Ryter van. (2000). Synergy in a neural code. *Neural Computation*, *12*(7), 1531–1552.

Browne, A., & Sun, R. (1999). Connectionist variable binding. *Expert Systems*, *16*(3), 189-207.

Buracas, G., Zador, A., DeWeese, M., & Albright, T. (1998). Efficient discrimination of temporal patterns by motion-sensitive neurons in primate visual cortex. *Neuron*, *20*, 959-969.

Campbell, S., Wang, D., & Jayapraksh, C. (1999). Synchrony and desynchrony in integrate-and-fire oscillators. *Neural Computation*, *11*, 1595-1619.

Carr, C., & Konishi, M. (1990). A circuit for detection of interaural time differences in the brain stem of the barn owl. *J. Neurosci.*, *10*, 3227-3246.

Chen, K., & Wang, D. (1999). Perceiving without learning: from spirals to insideoutside relations. In M. Kearns, S. Solla, & D. Cohn (Eds.), *Advances in neural information processing systems* (Vol. 11). The MIT Press.

Csiszar, I. (1975). I-divergence geometry of probability distributions and minimization problems. *Ann. Probab.*, *3*, 146-158.

de Ruyter van Steveninck, R., Borst, A., & Bialek, W. (2001). Real-time encoding of motion: answerable questions and questionable answers from the fly's visual system. In J. Zanker & J. Zeil (Eds.), *Motion vision – computational, neural, and ecological constraints.* Berlin Heidelberg New York: Springer Verlag.

de Ryter van Steveninck, R., Lewen, G., Strong, S., Koberle, R., & Bialek, W. (1997). Reproducibility and variability in neural spike trains. *Science*, *275*, 1805–1808.

deCharms, R., & Merzenich, M. (1996). Primary cortical representation of sounds by the coordination of action-potential timing. *Nature*, *381*, 610-613.

Delorme, A., Gautrais, J., VanRullen, R., & Thorpe, S. (1999). Spikenet: A simulator for modeling large networks of integrate and fire neurons. *Neurocomputing*, 989-996.

Deppisch, J., Bauer, H.-U., Schillen, T., König, P., Pawelzik, K., & Geisel, T. (1993). Alternating oscillatory and stochastic states in a network of spiking neurons. *Network: Computation in Neural Systems*, *4*, 243-257.

Diesmann, M., Gewaltig, M.-O., & Aertsen, A. (1999). Stable propagation of synchronous spiking in cortical neural networks. *Nature*, *402*, 529-33.

Douglas, R., & Martin, K. (1991). Opening the grey box. *Trends in Neurosciences*, *14*, 286-293.

Eckhorn, R., Bauer, R., Jordan, W., Broch, M., Kruse, W., Munk, M., & Reitboeck, H. (1988). Coherent oscillations: A mechanism of feature linking in the visual cortex? *Biol. Cybern.*, *60*, 121-130.

Engel, A., König, P., Kreiter, A., Schillen, T., & Singer, W. (1992). Temporal coding in the visual cortex: new vistas on integration in the nervous system. *Trends Neurosci.*, *15*, 218-226.

Eurich, C. W., & Wilke, S. D. (2000). Multi-dimensional encoding strategy of spiking neurons. *Neural Computation*, *12*, 1519-1529.

Fairhall, A., Lewen, G., Bialek, W., & de Ryter van Steveninck, R. (2001). Efficiency and ambiguity in an adaptive neural code. *Nature*, *412*, 787–792.

Feldman, D., Nicoll, R., Malenka, R., & Isaac, J. (1998). Long-term depression at thalamocortical synapses in developing somatosensory cortex. *Neuron*, *21*, 347–357.

Fellous, J.-M., Houweling, A., Modi, R., Rao, R., Tiesinga, P., & Sejnowski, T. (2001). Frequency dependence of spike timing reliability in cortical pyramidal cells and interneurons. *J. Neurophysiol.*, *85*, 1782–1787.

Fodor, J., & Pylyshyn, Z. (1988). Connectionism and cognitive architecture: a critical analysis. *Cognition*, *28*, 3-71.

Földiák, P. (1990). Forming sparse representations by local anti-hebbian learning. *Biological Cybernetics*, *64*, 165-170.

Földiák, P., & Young, P. (1995). Sparse coding in the primate cortex. In M. Arbib (Ed.), *The handbook of brain theory and neural networks.* Cambridge, MA: MIT Press.

Gerstner, W. (1995). Time structure of the activity in neural network models. *Phys. Rev. E*, *51*, 738-758.

Gerstner, W. (1998). Spiking neurons. In W. Maass & C. M. Bishop (Eds.), *Pulsed neural networks* (p. 3-55). The MIT Press.

Gerstner, W. (2000). Population dynamics of spiking neurons: Fast transients, asynchronous states, and locking. *Neural Computation*, *12*(1), 43-89.

Gerstner, W., Kempter, R., Hemmen, J. van, & Wagner, H. (1996). A neuronal learning rule for sub-millisecond temporal coding. *Nature*, *383*, 76-78.

Gokhale, D., & Kullback, S. (1978). *The information in contingency tables.* Dekker, New York.

Goren, Y. (2001, January). *Clustering with spiking neurons.* (http://www.cs.technion.ac.il/Labs/Isl/Project/Projects_done/SpikingNeurons/)

Gray, C. (1999). The temporal correlation hypothesis of visual feature integration: Still alive and well. *Neuron*, *24*, 31-47.

Gray, C., König, P., Engel, A., & Singer, W. (1989). Oscillatory responses in cat visual cortex exhibit inter-columnar synchronization which reflects global stimulus properties. *Nature*, *338*, 334-337.

Guedalia, I. D., London, M., & Werman, M. (1999). An on-line agglomerative clustering method for nonstationary data. *Neural Computation*, *11*(2), 521-540.

Harris, K., Henze, D., Hirase, H., Leinekugel, X., Dragoi, G., Czurkó, A., & Buzsáki. (2002). Spike train dynamics predicts theta-related phase precession in hippocampel pyramidal cells. *Nature*, *417*, 738-741.

Hebb, D. (1949). *The organization of behaviour.* New York: Wiley.

Heiligenberg, W. (1991). *Neural nets in electric fish.* MIT Press.

Helmuth, L. (2001). Synchronizing the brain's signals. *Nature*, *292*, 2233.

Hinton, G. E. (1990). Mapping part-whole hierarchies into connectionist networks. *Artificial Intelligence*, *46*(1), 47-75.

Hopfield, J. (1995). Pattern recognition computation using action potential timing for stimulus representation. *Nature*, *376*, 33-36.

Hopfield, J., & Brody, C. (2000). What is a moment? transient synchrony as a collective mechanism for spatiotemporal integration. *Proc. Natl. Acad. Sci. USA*, *97*, 13919-13924.

Kandel, E., Schwartz, J., & Jessell, T. (Eds.). (1991). *Principles of neural science* (third ed.). Amsterdam, London, New York: Elsevier/North-Holland.

Kanerva, P. (1988). *Sparse distributed memory.* MA, USA: MIT Press.

Kanerva, P. (1996). Binary spatter-coding of ordered k-tuples. In *ICANN* (p. 869-873).

Kemenade, C. van, La Poutré, H., & Mokken, R. (1999). Unsupervised class detection by adaptive sampling and density estimation. In A. Stein & F. van der Meer (Eds.), *Spatial statistics and remote sensing.* Kluwer.

Koenderink, J. (1984). The structure of images. *Biol. Cybern.*, *50*, 363-370.

König, P., Engel, A., & Singer, W. (1996). Integrator or coincidence detector? the role of the cortical neuron revisited. *Trends NeuroSci.*, *19*, 130-137.

König, P., Engel, A. K., Roelfsema, P. R., & Singer, W. (1995). How precise is neuronal synchronization? [Letter]. *Neural Computation*, *7*(3), 469–485.

König, P., & Schillen, T. B. (1991). Stimulus-dependent assembly formation of oscillatory responses: I.Synchronization [Letter]. *Neural Computation*, *3*(2), 155–166.

Konishi, M. (1991). Deciphering the brain's codes [Review]. *Neural Computation*, *3*(1), 1–18.

Kuwabara, N., & Suga, N. (1993). Delay lines and amplitude selectivity are created in subthalamic auditory nuclei: the branchium of the inferior colliculus of the mustached bat. *J. Neurophysiol.*, *69*, 1713-1724.

Lamme, V., & Spekreijse, H. (1998). Neuronal synchrony does not represent texture segregation. *Nature*, *396*, 362 – 366.

Laurent, G. (1999). A systems perspective on early olfactory coding. *Science*, *286*, 723-728.

Liu, R., Tzonev, S., Rebrik, S., & Miller, K. (1997). The structure and precision of retinal spike trains. *Proc. Nat. Acad. Sc. USA*, *94*, 5411–5416.

Liu, R., Tzonev, S., Rebrik, S., & Miller, K. (2001). Variability and information in a neural code of the cat lateral geniculate nucleus. *J. Neurophys.*, *86*, 2789–2806.

Livingston, M. (1996). Oscillatory firing and interneuronal correlations in squirrel monkey striate cortex. *J. Neurophysiol.*, *75*, 2467-2485.

Luck, S. J., & Vogel, E. K. (1997). The capacity of visual working memory for features and conjunctions. *Nature*, *390*, 279-281.

Maass, W. (1996). Lower Bounds for the Computational Power of Networks of Spiking Neurons. *Neural Computation*, *8*(1), 7-42.

Maass, W. (1997). Networks of spiking neurons: The third generation of neural network models. *Neural Networks*, *10*(9), 1659-1671.

Maass, W. (1999). Paradigms for computing with spiking neurons. In L. van Hemmen (Ed.), *Models of neural networks* (Vol. 4). Springer Berlin.

Maass, W., & Bishop, C. M. (1999). *Pulsed neural networks.* Cambridge, MA: The MIT Press.

MacGregor, R., & Oliver, R. (1974). A model for repetitive firing in neurons. *Kybernetik*, *16*, 53-64.

Mainen, Z., & Sejnowski, T. (1995). Reliability of spike timing in neocortical neurons. *Science*, *268*(5216), 1503–1506.

Markram, H., Lübke, J., Frotscher, M., & Sakmann, B. (1997). Regulation of synaptic efficacy by coincidence of postsynaptic aps and epsps. *Nature*, *375*, 213–215.

Martignon, L., Hasseln, H. von, Grün, S., Aertsen, A., & Palm, G. (1995). Detecting higher-order interactions among the spiking events in a group of neurons. *Biol. Cybern.*, *73*, 69-81.

Mattia, M., & Giudice, P. D. (2000). Efficient event-driven simulation of large networks of spiking neurons and dynamical synapses. *Neural Computation*, *12*(10), 2305–2329.

McCulloch, W., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, *5*, 115–133.

Mehta, M., Lee, A., & Wilson, M. (2002). Role of experience and oscillations in transforming a rate code into a temporal code. *Nature*, *417*, 741-746.

Mel, B., & Fiser, J. (2000). Minimizing binding errors using learned conjunctive features. *Neural Computation*, *12*, 247-278.

Michie, D., Spiegelhalter, D., & Taylor, C. (Eds.). (1994). *Machine learning, neural and statistical classification.* Ellis Horwood, West Sussex.

Minsky, M., & Papert, S. (1969). *Perceptrons: An introduction to computational geometry.* MIT Press.

Murthy, V. N., & Fetz, E. E. (1994). Effects of input synchrony on the firing rate of a three-conductance cortical neuron model [Letter]. *Neural Computation*, *6*(6), 1111–1126.

Natschläger, T., & Ruf, B. (1998). Spatial and temporal pattern analysis via spiking neurons. *Network: Comp. Neural Syst.*, *9*(3), 319-338.

O'Keefe, J., & Recce, M. (1993). Phase relationship between hippocampal place units and the EEG theta rhythm. *Hippocampus*, *3*, 317-330.

Olshausen, B., & Field, D. (1996). Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, *381*, 607-609.

O'Reilly, R. C., & Busby, R. S. (2001). Generalizable relational binding from coarse-coded distributed representations. In *Advances in neural information processing systems.* The MIT Press.

Pieri, G. (2001). *Improvements in handwritten character recognition: a mixture of experts.* Unpublished master's thesis, Free University of Amsterdam.

Plate, T. A. (1995). Holographic reduced representations [Paper]. *IEEE Transactions on Neural Networks*, *6*(3), 623–641.

Pollack, J. B. (1990). Recursive distributed representations. *Artificial Intelligence*, *46*(1), 77-105.

Pouget, A., Deneve, S., Ducom, J.-C., & Latham, P. (1999). Narrow versus wide tuning curves: What's best for a population code? *Neural Computation*, *11*(1), 85-90.

Press, W., Flanney, B., Teukolsky, S., & Vetterling, W. (1986). *Numerical recipes: the art of scientific computing.* Cambridge University Press, Cambridge.

Rachkovskij, D., & Kussul, E. (2001). Binding and normalization of binary sparse distributed representations by context-dependent thinning. *Neural Computation*, *13*, 411-452.

Reich, D., Mechler, F., & Victor, J. (2001). Independent and redundant information in nearby cortical neurons. *Science*, *294*, 2566–2568.

Richmond, B. (2001). Information coding. *Science*, *294*, 2493–2494.

Riehle, A., Grun, S., Diesmann, M., & Aertsen, A. (1997). Spike synchronization and rate modulation differentially involved in motor cortical function. *Science*, *278*, 1950-1953.

Ripley, B. (1996). *Pattern recognition and neural networks.* Clarendon Press, Oxford.

Roelfsema, P. (1998). Solutions for the binding problem. *Z. Naturforsch.*, *53*(7-8), 691-715.

Roelfsema, P., Engel, A., König, P., & Singer, W. (1997). Visuomotor integration is associated with zero time-lag synchronization among cortical areas. *Nature*, *385*, 157-161.

Roelfsema, P. R., Bohte, S. M., & Spekreijse, H. (1999). Algorithms for the detection of connectedness and their neural implementation. In O. Parodi (Ed.), *Proceedings of the Corsican Neuroscience Summerschool.*

Rosenblatt, F. (1961). *Principles of neurodynamics: Perception and the theory of brain mechanisms.* Washington, CD: Spartan Books.

Roy, A., Govil, S., & Miranda, R. (1995). An algorithm to generate radial basis function(rbf)-like nets for classification problems. *Neural Networks*, *8*(2), 175-201.

Ruf, B., & Schmitt, M. (1998). Self-organization of spiking neurons using action potential timing. *IEEE-Trans. Neural Networks*, *9*(3), 575–578.

Rumelhart, D., Hinton, G., & Williams, R. (1986). Learning representations by back-propagating errors. *Nature*, *325*, 533-536.

Rumelhart, D., McClelland, J., & PDP Research Group the. (1986). *Parallel distributed processing: Explorations in the microstructure of cognition* (Vol. 1). Cambridge, MA: MIT Press.

Shadlen, M., & Movshon, J. (1999). Synchrony unbound: A critical evaluation of the temporal binding hypothesis. *Neuron*, *24*, 67-77.

Shadlen, M., & Newsome, W. (1994). Noise, neural codes and cortical organization. *Curr. Opin. Neurobiol.*, *4*, 569-579.

Shadlen, M., & Newsome, W. (1995). Is there a signal in the noise? *Curr. Opin. Neurobiol.*, *5*, 248-250.

Shadlen, M., & Newsome, W. (1998). The variable discharge of cortical neurons: implications for connectivity, computation and information coding. *J. Neuronsci*, *18*, 3870-3896.

Singer, W. (1995). Development and plasticity of cortical processing architectures. *Science*, *270*, 758-764.

Singer, W. (1999). Neuronal synchrony: A versatile code for the definition of relations. *Neuron*, *24*, 49-65.

Singer, W., & Gray, C. (1995). Visual feature integration and the temporal correlation hypothesis. *Arnu. Rev. Neurosci.*, *18*, 555-589.

Smolensky, P. (1990). Tensor product variable binding and the representation of symbolic structures in connectionist networks. *Artificial Intelligence*, *46*(1), 159-216.

Snippe, H., & Koenderink, J. (1992). Information in channel-coded systems: correlated receivers. *Biol. Cybern.*, *67*(2), 183-190.

Softky, W. (1995). Simple codes versus efficient codes. *Curr. Opin. Neurobiol.*, *5*, 239-247.

Softky, W., & Koch, C. (1993). The highly irregular firing of cortical cells is inconsistent with temporal integration of random epsp's. *J. Neurosci.*, *13*, 334-350.

Steinmetz, P., Roy, A., Fitzgerald, P., Hsiao, S., Johnson, K., & Niebur, E. (2000). Attention modulates synchronized neuronal firing in primate somatosensory cortex. *Nature*, *404*, 187-190.

Stopfer, M., & Laurent, G. (1999). Short-term memory in olfactory network dynamics. *Nature*, *402*, 610-614.

Stratford, K., Tarczy-Hornoch, K., Martin, K., Bannister, N., & Jack, J. (1996). Excitatory synaptic inputs to spiny stellate cells in cat visual cortex. *Nature*, *382*, 258–261.

't Hoen, P. J., Bohte, S. M., Gerding, E., & La Poutré, H. (2002a). Implementation of a competitive market-based allocation of consumer attention space. In W. Walsh (Ed.), *Lecture Notes in Computer Science: Proceedings of the 4th Workshop on Agent Mediated Electronic Commerce.* (Vol. 2531). Springer.

't Hoen, P. J., Bohte, S. M., Gerding, E., & La Poutré, H. (2002b). *Implementation of a competitive market-based allocation of consumer attention space* (Tech. Rep.). CWI Technical Report SEN-R0217, www.cwi.nl/∼sbohte.

Thorpe, S., Fize, F., & Marlot, C. (1996). Speed of processing in the human visual system. *Nature*, *381*, 520-522.

Thorpe, S., & Gautrais, J. (1997). Rapid visual processing using spike asynchrony. In M. C. Mozer, M. I. Jordan, & T. Petsche (Eds.), *Advances in neural information processing systems* (Vol. 9, p. 901). The MIT Press.

Tsodyks, M., Uziel, A., & Markram, H. (2000). Synchrony generation in recurrent networks with frequency-dependent synapses. *J. Neuroscience*, *20*(RC50), 1-5.

Tsodyks, M. V., & Sejnowski, T. (1995). Rapid state switching in balanced cortical network models. *Network: Computation in Neural Systems*, *6*, 111-124.

Vaadia, E., & Aertsen, A. (1992). Coding and computation in the cortex: single neuron activity and cooperative phenomena. In A. Aertsen & V. Braitenberg (Eds.), *Information processing in the cortex.* Berlin: Springer Verlag.

Volgushev, M., & Eysel, U. (2000). Noise makes sense in neuronal computing. *Science*, *290*, 1908–1909.

von der Malsburg, C. (1981). The correlation theory of brain function. *Internal Report 81-2, Max-Planck-Institute for Biophysical Chemistry, Göttingen, Germany.*

von der Malsburg, C. (1999). The what and why of binding: The modeler's perspective. *Neuron*, *24*, 95-104.

von der Malsburg, C., & Schneider, W. (1986). A neural cocktail-party processor. *Biol. Jybern.*, *54*, 29-40.

Werbos, P. (1974). *Beyond regression: new tools for prediction and analysis in the behavioural sciences.* Unpublished doctoral dissertation, Harvard University.

Williams, S., & Stuart, G. (2002). Dependence of epsp efficacy on synapse location in neocortical pyramidal neurons. *Science*, *295*, 1907–1910.

Wolberg, W. (1991). *Cancer dataset obtained from Williams H. Wolberg from the University of Wisconsin Hosqitals, Madison.*

Xin, J., & Embrechts, M. (2001). Supervised learning with spiking neural networks. In *Proceedings of International Joint Conference on Neural Networks (IJCNN)* (pp. 1772–1777).

Zhang, K., & Sejnowski, T. (1999). Neuronal tuning: To sharpen or broaden? *Neural Computation*, *11*(1), 75-84.

Zhang, K.-C., Ginzburg, I., McNaughton, B., & Sejnowski, T. (1998). Interpreting neuronal population activity by reconstruction: Unified framework with application to hippocampal place cells. *J. Neurophysiol.*, *79*, 1017-1044.

Zhang, L., Tao, H., Holt, C., Harris, W., & Poo, M. (1998). A critical window for cooperation and competition among developing retinotectal synapses. *Nature*, *395*(3), 37–44.

Zurada, J. (1992). *Introduction to artifical neural systems.* St. Paul, MN: West.

SAMENVATTING

Tussen de netwerken van neuronen in de zenuwstelsels van dieren, en de computer programma's die bekend staan als kunstmatige neurale netwerken zit een belangrijke vertaalslag: het model van hoe echte neuronen informatie verwerken. Echte neuronen communiceren via electrische pulsjes – spikes – die andere neuronen bereiken via een netwerk van verbindingen, alsmede ook via chemische stoffen. Samen leidt dit ertoe dat een "gebruiker" van deze netwerken informatie uit de buitenwereld (inclusief het lichaam zelf) verwerkt en deze omzet in acties (bv spierbewegingen) naar de buitenwereld toe. De functie van echte neurale netwerken is met andere woorden redelijk duidelijk. Wat aanzienlijk minder duidelijk is, is hoe deze neurale netwerken dit precies voor elkaar krijgen.

Inzichten in hoe biologische neurale netwerken werken wordt vooral getest door de kennis die er is in een (computer)model te stoppen, en dan te onderzoeken of dit computermodel ook de functies kan uitvoeren die echte neurale netwerken aankunnen. Tot vrij recent was het idee dat de informatie die neuronen naar elkaar sturen vooral beschreven wordt door de *frequentie* waarmee een neuron spikes naar andere neuronen stuurt. Door deze frequentie te variëren kan een neuron een analoog getal, bijvoorbeeld tussen 0 en 1, doorgeven aan andere neuronen. Deze andere neuronen kunnen dan intern een functie uitrekenen van alle analoge getallen die ze van andere neuronen krijgen, en daar weer hun eigen analoge getal mee bepalen, die vervolgens weer doorgestuurd wordt naar andere neuronen. De mate waarin ontvangende neuronen getallen van andere neuronen meenemen in hun berekening wordt bepaald door aan elke binnenkomende verbinding een *gewicht* toe te kennen.

Traditioneel bestaan kunstmatige neurale netwerken uit netwerken van modellen van neuronen die op deze manier werken. Door de gewichten van de verbindingen tussen de neuronen slim te kiezen, kunnen dit soort netwerken verrassend veel functies implementeren, zoals patroon-herkenning (handschriftherkenning, gezichtsherkenning) en functie-benadering (bijv het leren van de sinus-functie gegeven een aantal (x,y) punten). De toepassingen van kunstmatige neurale netwerken zijn duidelijk velerlei, maar ze komen nog lang niet in de buurt van de flexibiliteit en kracht van de biologische neurale netwerken waar ze model voor staan.

Juist op dit punt doet een belangrijk inzicht vanuit de biologie op dit moment opgeld: het model van neuronen die communiceren via een frequentie van pulsjes is duidelijk niet voldoende om te verklaren wat er gebeurt. Specifiek blijkt steeds duidelijker dat een belangrijk deel van de informatie die neuronen communiceren verpakt zit in het precieze *moment* waarop ze pulsjes versturen. Theoretisch is inmiddels bekend dat deze toevoeging aan het model neurale netwerken in principe aanzienlijk krachtiger maakt. Wat in dit proefschrift aan bod komt, zijn methoden om daadwerkelijk met dit soort nieuwe kunstmatige *spiking neurale netwerken* functies te implementeren.

In Hoofdstuk 2 wordt een spiking neuraal netwerk gesimuleerd dat uit zichzelf strukturen ontdekt in informatie die het wordt aangereikt uit de buitenwereld. Met behulp van regels voor het aanpassen van de gewichten tussen *spiking neuronen*, de leerregel – sterk geinspireerd door wat er daadwerkelijk in echte neuronen wordt gemeten – blijken dit soort netwerken uitstekend dergelijke strukturen te kunnen herkennen, ook wanneer er ingewikkelde, hierarchische verbanden tussen de stukjes informatie bestaat.

In Hoofdstuk 3 wordt een leerregel afgeleid die de gewichten van de verbindingen tussen spiking neuronen verandert zodanig dat het netwerk van neuronen geleerd kan worden wat de juiste uitkomst is, gegeven informatie van de buitenwereld (supervised learning). De leerregel is een variant van de error-backpropagation methode zoals die voor traditionele neurale netwerken is afgeleid, maar dan specifiek voor netwerken van spiking neuronen. In de praktijk blijken spiking neurale netwerken met deze leerregel net zo goed te werken als traditionele neurale netwerken. Een belangrijk verschil is echter dat wanneer we het model terugvertalen naar de biologie, het spiking neurale netwerk *of* aanzienlijk minder echte neuronen nodig heeft, *of* aanzienlijk sneller werkt (er is een directe trade-off tussen deze twee eigenschappen). Het literatuuronderzoek naar de biologie van spiking neurale netwerken, gepresenteerd in Hoofdstuk 7, concludeert ook dat de relevante publicaties vooral laten zien dat preciese timing van spikes vooral gevonden wordt in omstandigheden waar een dier zeer *snel* moet reageren op de buitenwereld.

In Hoofdstuk 4 en 5 wordt een spiking neuraal netwerk gepresenteerd dat laat zien hoe de specifieke eigenschappen van spiking neurons kunnen worden ingezet om een oud probleem uit de neurale netwerk literatuur aan te pakken. De winst van de gepresenteerde neurale netwerk architectuur is dat voor het herkennen van meerdere (visuele) objecten tegelijk met een neural netwerk, er relatief weinig neuronen nodig zijn.

In Hoofdstuk 6 wordt onderzocht hoe biologische metingen over de precieze timing van spikes het meest waarschijnlijk geïnterpreteerd moeten worden: of neuronen de neiging hebben precies tegelijkertijd een puls te vuren wordt in onderzoeken gemeten door de *correlatie* tussen paren van neuronen te meten. Voor bijvoorbeeld een correlatie-waarde van 0.5 wordt hieruit dan impliciet afgeleid dat gemiddeld in een populatie van neuronen (zeg 100) zo'n 50 (0.5 x 100) tegelijkertijd zullen vuren. In dit hoofdstuk wordt aangetoond dat, onder een aantal aannames, dit een tamelijk onwaarschijnlijke situatie is: het is veel aannemelijker dat af en toe alle 100 neuronen tegelijkertijd vuren, en verder meestal slechts enkelen. In experimenten met een simpel neural netwerkje blijkt dit ook het geval te zijn.

In Hoofdstuk 7 wordt eerst een kort overzicht gegeven van hoe echte spiking neuronen nou precies werken, en vervolgens een literatuurstudie over waar en hoe precies getimede spikes gevonden worden in levende dieren. Binnen de neurowetenschappen is het onderwerp van preciese spikes relatief nieuw, en nog steeds controversieel. Toch wordt steeds meer duidelijk dat de preciese timing van enkele spikes wel degelijk belangrijk is, vooral in situaties wanneer een dier (zeer) snel op zijn omgeving moet reageren. Dit suggereert dat het model van kunstmatige neurale netwerken dat in dit proefschrift is onderzocht een zinnige beschrijving is van neural netwerken zoals ze in de natuur bestaan.

CURRICULUM VITAE

Sander Marcel Bohte werd geboren op 28 december 1974 te Hoorn, Noord-Holland. Van 1989 tot 1992 doorliep hij het Atheneum op het Montessori Lyceum, te Amsterdam. Van 1992 tot 1997 studeerde hij natuurkunde aan de Universiteit van Amsterdam, alwaar hij afstudeerde in de experimentele natuurkunde, bij de groep Visuele Systeem Analyse van de vakgroep medische fysica van het AMC. Van 1998 tot 2002 was hij als Onderzoeker in Opleiding (OiO) aangesteld bij het CWI in de groep van professor Han La Poutré, alwaar hij zijn promotieonderzoek verrichtte. Op dit moment is hij aangesteld als Onderzoeker in deze groep.

Titles in the IPA Dissertation Series

J.O. Blanco. *The State Operator in Process Algebra*. Faculty of Mathematics and Computing Science, TUE. 1996-01

A.M. Geerling. *Transformational Development of Data-Parallel Algorithms*. Faculty of Mathematics and Computer Science, KUN. 1996-02

P.M. Achten. *Interactive Functional Programs: Models, Methods, and Implementation*. Faculty of Mathematics and Computer Science, KUN. 1996-03

M.G.A. Verhoeven. *Parallel Local Search*. Faculty of Mathematics and Computing Science, TUE. 1996-04

M.H.G.K. Kesseler. *The Implementation of Functional Languages on Parallel Machines with Distrib. Memory*. Faculty of Mathematics and Computer Science, KUN. 1996-05

D. Alstein. *Distributed Algorithms for Hard Real-Time Systems*. Faculty of Mathematics and Computing Science, TUE. 1996-06

J.H. Hoepman. *Communication, Synchronization, and Fault-Tolerance*. Faculty of Mathematics and Computer Science, UvA. 1996-07

H. Doornbos. *Reductivity Arguments and Program Construction*. Faculty of Mathematics and Computing Science, TUE. 1996-08

D. Turi. *Functorial Operational Semantics and its Denotational Dual*. Faculty of Mathematics and Computer Science, VUA. 1996-09

A.M.G. Peeters. *Single-Rail Handshake Circuits*. Faculty of Mathematics and Computing Science, TUE. 1996-10

N.W.A. Arends. *A Systems Engineering Specification Formalism*. Faculty of Mechanical Engineering, TUE. 1996-11

P. Severi de Santiago. *Normalisation in Lambda Calculus and its Relation to Type Inference*. Faculty of Mathematics and Computing Science, TUE. 1996-12

D.R. Dams. *Abstract Interpretation and Partition Refinement for Model Checking*. Faculty of Mathematics and Computing Science, TUE. 1996-13

M.M. Bonsangue. *Topological Dualities in Semantics*. Faculty of Mathematics and Computer Science, VUA. 1996-14

B.L.E. de Fluiter. *Algorithms for Graphs of Small Treewidth*. Faculty of Mathematics and Computer Science, UU. 1997-01

W.T.M. Kars. *Process-algebraic Transformations in Context*. Faculty of Computer Science, UT. 1997-02

P.F. Hoogendijk. *A Generic Theory of Data Types*. Faculty of Mathematics and Computing Science, TUE. 1997-03

T.D.L. Laan. *The Evolution of Type Theory in Logic and Mathematics*. Faculty of Mathematics and Computing Science, TUE. 1997-04

C.J. Bloo. *Preservation of Termination for Explicit Substitution*. Faculty of Mathematics and Computing Science, TUE. 1997-05

J.J. Vereijken. *Discrete-Time Process Algebra*. Faculty of Mathematics and Computing Science, TUE. 1997-06

F.A.M. van den Beuken. *A Functional Approach to Syntax and Typing*. Faculty of Mathematics and Informatics, KUN. 1997-07

A.W. Heerink. *Ins and Outs in Refusal Testing*. Faculty of Computer Science, UT. 1998-01

G. Naumoski and W. Alberts. *A Discrete-Event Simulator for Systems Engineering*. Faculty of Mechanical Engineering, TUE. 1998-02

J. Verriet. *Scheduling with Communication for Multiprocessor Computation*. Faculty of Mathematics and Computer Science, UU. 1998-03

J.S.H. van Gageldonk. *An Asynchronous Low-Power 80C51 Microcontroller*. Faculty of Mathematics and Computing Science, TUE. 1998-04

A.A. Basten. *In Terms of Nets: System Design with Petri Nets and Process Algebra*. Faculty of Mathematics and Computing Science, TUE. 1998-05

E. Voermans. *Inductive Datatypes with Laws and Subtyping – A Relational Model*. Faculty of Mathematics and Computing Science, TUE. 1999-01

H. ter Doest. *Towards Probabilistic Unification-based Parsing*. Faculty of Computer Science, UT. 1999-02

J.P.L. Segers. *Algorithms for the Simulation of Surface Processes*. Faculty of Mathematics and Computing Science, TUE. 1999-03

C.H.M. van Kemenade. *Recombinative Evolutionary Search*. Faculty of Mathematics and Natural Sciences, UL. 1999-04

E.I. Barakova. *Learning Reliability: a Study on Indecisiveness in Sample Selection*. Faculty of Mathematics and Natural Sciences, RUG. 1999-05

M.P. Bodlaender. *Schedulere Optimization in Real-Time Distributed Databases*. Faculty of Mathematics and Computing Science, TUE. 1999-06

M.A. Reniers. *Message Sequence Chart: Syntax and Semantics*. Faculty of Mathematics and Computing Science, TUE. 1999-07

J.P. Warners. *Nonlinear approaches to satisfiability problems*. Faculty of Mathematics and Computing Science, TUE. 1999-08

J.M.T. Romijn. *Analysing Industrial Protocols with Formal Methods*. Faculty of Computer Science, UT. 1999-09

P.R. D'Argenio. *Algebras and Automata for Timed and Stochastic Systems*. Faculty of Computer Science, UT. 1999-10

G. Fábián. *A Language and Simulator for Hybrid Systems*. Faculty of Mechanical Engineering, TUE. 1999-11

J. Zwanenburg. *Object-Oriented Concepts and Proof Rules*. Faculty of Mathematics and Computing Science, TUE. 1999-12

R.S. Venema. *Aspects of an Integrated Neural Prediction System*. Faculty of Mathematics and Natural Sciences, RUG. 1999-13

J. Saraiva. *A Purely Functional Implementation of Attribute Grammars*. Faculty of Mathematics and Computer Science, UU. 1999-14

R. Schiefer. *Viper, A Visualisation Tool for Parallel Progam Construction*. Faculty

of Mathematics and Computing Science, TUE. 1999-15

K.M.M. de Leeuw. *Cryptology and Statecraft in the Dutch Republic*. Faculty of Mathematics and Computer Science, UvA. 2000-01

T.E.J. Vos. *UNITY in Diversity. A stratified approach to the verification of distributed algorithms*. Faculty of Mathematics and Computer Science, UU. 2000-02

W. Mallon. *Theories and Tools for the Design of Delay-Insensitive Communicating Processes*. Faculty of Mathematics and Natural Sciences, RUG. 2000-03

W.O.D. Griffioen. *Studies in Computer Aided Verification of Protocols*. Faculty of Science, KUN. 2000-04

P.H.F.M. Verhoeven. *The Design of the MathSpad Editor*. Faculty of Mathematics and Computing Science, TUE. 2000-05

J. Fey. *Design of a Fruit Juice Blending and Packaging Plant*. Faculty of Mechanical Engineering, TUE. 2000-06

M. Franssen. *Cocktail: A Tool for Deriving Correct Programs*. Faculty of Mathematics and Computing Science, TUE. 2000-07

P.A. Olivier. *A Framework for Debugging Heterogeneous Applications*. Faculty of Natural Sciences, Mathematics and Computer Science, UvA. 2000-08

E. Saaman. *Another Formal Specification Language*. Faculty of Mathematics and Natural Sciences, RUG. 2000-10

M. Jelasity. *The Shape of Evolutionary Search Discovering and Representing Search Space Structure*. Faculty of Mathematics and Natural Sciences, UL. 2001-01

R. Ahn. *Agents, Objects and Events a computational approach to knowledge, observation and communication*. Faculty of Mathematics and Computing Science, TU/e. 2001-02

M. Huisman. *Reasoning about Java programs in higher order logic using PVS and Isabelle*. Faculty of Science, KUN. 2001-03

I.M.M.J. Reymen. *Improving Design Processes through Structured Reflection*. Faculty of Mathematics and Computing Science, TU/e. 2001-04

S.C.C. Blom. *Term Graph Rewriting: syntax and semantics*. Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2001-05

R. van Liere. *Studies in Interactive Visualization*. Faculty of Natural Sciences, Mathematics and Computer Science, UvA. 2001-06

A.G. Engels. *Languages for Analysis and Testing of Event Sequences*. Faculty of Mathematics and Computing Science, TU/e. 2001-07

J. Hage. *Structural Aspects of Switching Classes*. Faculty of Mathematics and Natural Sciences, UL. 2001-08

M.H. Lamers. *Neural Networks for Analysis of Data in Environmental Epidemiology: A Case-study into Acute Effects of Air Pollution Episodes*. Faculty of Mathematics and Natural Sciences, UL. 2001-09

T.C. Ruys. *Towards Effective Model Checking*. Faculty of Computer Science, UT. 2001-10

D. Chkliaev. *Mechanical verification of concurrency control and recovery protocols*. Faculty of Mathematics and Computing Science, TU/e. 2001-11

M.D. Oostdijk. *Generation and presentation of formal mathematical documents*. Faculty of Mathematics and Computing Science, TU/e. 2001-12

A.T. Hofkamp. *Reactive machine control: A simulation approach using χ*. Faculty of Mechanical Engineering, TU/e. 2001-13

D. Bošnački. *Enhancing state space reduction techniques for model checking*. Faculty of Mathematics and Computing Science, TU/e. 2001-14

M.C. van Wezel. *Neural Networks for Intelligent Data Analysis: theoretical and experimental aspects*. Faculty of Mathematics and Natural Sciences, UL. 2002-01

V. Bos and J.J.T. Kleijn. *Formal Specification and Analysis of Industrial Systems*. Faculty of Mathematics and Computer Science and Faculty of Mechanical Engineering, TU/e. 2002-02

T. Kuipers. *Techniques for Understanding Legacy Software Systems*. Faculty of Natural Sciences, Mathematics and Computer Science, UvA. 2002-03

S.P. Luttik. *Choice Quantification in Process Algebra*. Faculty of Natural Sciences, Mathematics, and Computer Science, UvA. 2002-04

R.J. Willemen. *School Timetable Construction: Algorithms and Complexity*. Faculty of Mathematics and Computer Science, TU/e. 2002-05

M.I.A. Stoelinga. *Alea Jacta Est: Verification of Probabilistic, Real-time and Parametric Systems*. Faculty of Science, Mathematics and Computer Science, KUN. 2002-06

N. van Vugt. *Models of Molecular Computing*. Faculty of Mathematics and Natural Sciences, UL. 2002-07

A. Fehnker. *Citius, Vilius, Melius: Guiding and Cost-Optimality in Model Checking of Timed and Hybrid Systems*. Faculty of Science, Mathematics and Computer Science, KUN. 2002-08

R. van Stee. *On-line Scheduling and Bin Packing*. Faculty of Mathematics and Natural Sciences, UL. 2002-09

D. Tauritz. *Adaptive Information Filtering: Concepts and Algorithms*. Faculty of Mathematics and Natural Sciences, UL. 2002-10

M.B. van der Zwaag. *Models and Logics for Process Algebra*. Faculty of Natural Sciences, Mathematics, and Computer Science, UvA. 2002-11

J.I. den Hartog. *Probabilistic Extensions of Semantical Models*. Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2002-12

L. Moonen. *Exploring Software Systems*. Faculty of Natural Sciences, Mathematics, and Computer Science, UvA. 2002-13

J.I. van Hemert. *Applying Evolutionary Computation to Constraint Satisfaction and Data Mining*. Faculty of Mathematics and Natural Sciences, UL. 2002-14

S. Andova. *Probabilistic Process Algebra*. Faculty of Mathematics and Computer Science, TU/e. 2002-15

Y.S. Usenko. *Linearization in μCRL*. Faculty of Mathematics and Computer Science, TU/e. 2002-16

J.J.D. Aerts. *Random Redundant Storage for Video on Demand*. Faculty of Mathematics and Computer Science, TU/e. 2003-01

M. de Jonge. *To Reuse or To Be Reused: Techniques for component composition and construction*. Faculty of Natural Sciences, Mathematics, and Computer Science, UvA. 2003-02

J.M.W. Visser. *Generic Traversal over Typed Source Code Representations*. Faculty of Natural Sciences, Mathematics, and Computer Science, UvA. 2003-03

S.M. Bohte. *Spiking Neural Networks*. Faculty of Mathematics and Natural Sciences, UL. 2003-04