GRAZ UNIVERSITY OF TECHNOLOGY

DISSERTATION

to obtain the title of

Doctor of Technical Sciences

of Graz University of Technology

Defended by

Michael Andreas Pfeiffer

Concepts and Methods from Machine Learning as Tools for the Analysis of Computations in Nervous Systems

Thesis Advisor: Univ.Prof. DI Dr. Wolfgang Maass

defended on February 4, 2010

Jury :

Advisor :	Univ.Prof. DI Dr. Wolfgang Maass	-	TU Graz
Reviewer :	Univ.Prof. Dr. Rodney J. Douglas	-	ETH Zürich
Dean of Studies:	Univ.Doz. DI Dr. Oswin Aichholzer	-	TU Graz

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbständig verfasst, andere als die angegebenen Quellen / Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommene Stellen als solche kenntlich gemacht habe.

Statutory Declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

Graz, 11 January 2010 .....

(signature)

Abstract

This thesis investigates how innovative machine learning methods, which autonomously extract information from data, can be used to gain insights into neural information processing. The brain provides a framework which has been optimized through evolution to support fast and robust adaptation to the environment, thereby increasing chances of survival. Building upon the mathematically framework of machine learning allows us to study the role of experimentally observed synaptic learning phenomena, or to use analogies from neuroscience in order to improve machine learning algorithms for difficult real-world tasks.

My dissertation is structured into several parts, which highlight the multiple possibilities where machine learning and computational neuroscience can fruitfully interact. The first part studies auditory information processing by insects in real-world scenarios. Analyzing recordings that were performed in the natural habitats of the insects in the tropical rainforest, we find that neural coding with characteristic burst firing patterns provides a reliable way of transmitting information in situations where signals are heavily distorted by environmental noise. The second part uses neural network techniques to construct models of high-level human behavior. The relevance of text that humans were reading was classified from the movements of their eyes. Our approach was so successful that it finished first in an international competition. In the third part a new algorithm for reward-based learning in continuous state- and action spaces is presented, which draws inspiration from neuroscientific concepts of motor control. In combination with sample-based models and innovative exploration policies this leads to an improvement over existing algorithms, which are applicable for robotics tasks. The final parts of my thesis links biologically plausible Hebbian learning mechanisms to mathematical concepts of learning and decision making. Neural network models are presented in which simple synaptic plasticity rules with strong convergence guarantees lead to approximately optimal decisions in a Bayesian sense. This shows how nervous systems can learn strategies for a rich variety of tasks with apparently very simple and limited basic units of computations, i.e. neurons and synapses. The presented approach makes concrete predictions for sparse, redundant neural codes for input signals, with which Hebbian learning can quickly and robustly lead to sensible decisions. We first present mechanisms for supervised learning, and extend these rules to reward-modulated learning in winner-take-all networks, where action selection policies are learned from rewards and punishments. Finally, we explore functional roles for spike-timing dependent plasticity (STDP) in soft winner-take-all circuits of spiking neurons. It is shown that spiking neurons can learn implicit internal models of high-dimensional input signals without supervision, thereby identifying hidden causes of inputs. In particular, it is shown that STDP is able to approximate a stochastic online Expectation-Maximization algorithm for modeling the input data.

Keywords: Machine learning, computational neuroscience, reinforcement learning, Hebbian plasticity, Bayesian methods

Zusammenfassung

In dieser Arbeit wird untersucht, wie man neue Einblicke in neuronale Informationsverarbeitung gewinnen kann durch den Einsatz neuartiger Methoden des Maschinellen Lernens. Das Gehirn stellt ein durch die Evolution optimiertes Gerüst für schnelle und robuste Anpassung an die Umgebung dar, um die Überlebenschancen von Organismen zu erhöhen. Aufbauend auf mathematischen Konzepten kann man die Rolle von experimentell beobachteten Lernvorgängen studieren, oder Analogien zur Hirnforschung verwenden um Algorithmen für komplexe real-world Anwendungen zu verbessern.

Die verschiedenen Teile meiner Dissertation zeigen die vielfältigen Möglichkeiten einer fruchtbaren Verbindung von Neurowissenschaft und Maschinellem Lernen auf. Zuerst wird auditorische Informationsverarbeitung von Insekten in ihrer natürlichen Umgebung untersucht. Bei der Analyse von Aufzeichnungen im natürlichen Habitat der Insekten konnte ein neuronaler Code mit charakteristischen Feuermustern in Form von Bursts identifiziert werden, der als Grundlage der Übermittlung von Information bei starken natürlichen Hintergrundgeräuschen dient. Im zweiten Teil wurden Neuronale Netzwerke verwendet um menschliches Leseverhalten aufgrund von Augenbewegungen vorherzusagen. Unser Ansatz um die Relevanz eines gelesenen Texts vorherzusagen gewann den ersten Preis bei einem internationalen Wettbewerb. Im dritten Teil wird ein Algorithmus für Reinforcement Learning in kontinuierlichen Umgebungen präsentiert, der durch neurowissenschaftliche Erkenntnisse aus dem motorischen System inspiriert wird. Zusammen mit neuartigen Methoden für Modell-Lernen und Exploration führt dieser Ansatz zu verbesserten Algorithmen z.B. für Robotik-Anwendungen. Im letzten Teil der Dissertation wird eine Verbindung von biologisch plausibler Hebb'scher Plastizität und mathematischen Modellen für Lernen und Entscheidungsfindung hergestellt. Es werden neuronale Netzwerk-Modelle präsentiert, in denen einfache synaptische Lernregeln zuverlässig zu annähernd Bayes-optimalen Entscheidungen führen. Dies zeigt, wie Nervensysteme Strategien für komplexe Aufgaben mittels Neuronen und Synapsen lernen können. Der vorgestellte Ansatz macht konkrete Vorhersagen, wie Neuronale Codes für Inputsignale aussehen könnten, mit denen Hebb'sches Lernen schnell und robust Entscheidungen findet. Mechanismen für supervised Learning werden erweitert auf reward-modulierte Lernregeln in sogenannten winner-take-all networks, wodurch Entscheidungs-Strategien anhand von Feedback gelernt werden. Abschließend wird die funktionale Aufgabe von Spike-timing Dependent Plasticity (STDP) in soft winner-take-all Netzwerken spikender Neuronen untersucht. Spikende Neuronen bilden ohne externen Supervisor implizite interne Modelle ihrer hoch-dimensionalen Inputs, und identifizieren dadurch versteckte Features des Inputs. Insbesondere wird gezeigt, dass STDP einen stochastischen online Expectation-Maximization Algorithmus zur Modellierung der Input-Verteilung approximiert.

Keywords: Maschinelles Lernen, Computational Neuroscience, Reinforcement Learning, Hebb'sche Plastizität, Bayes'sche Methoden

Acknowledgments

It is a pleasure to thank the many people who have helped making this thesis possible. First of all, I want to thank my advisor Wolfgang Maass, who has made his support available in a number of ways. He enabled me to work on very exciting topics, and attracted me to look into fields of science that were new to me. His expertise and visions have greatly inspired my research. I am also very grateful to Rodney Douglas, in particular for taking time out from his very busy schedule to review my thesis and attend my defense. I also want to thank him for the fruitful collaboration within the SECO project, which has lead to the very nice joint paper that is the basis of Chapter 6.

I would like to show my gratitude to my co-authors and colleagues Bernhard Nessler, Gerhard Neumann, Amir Saffari, and Andreas Juffinger, who have contributed substantial ideas to our joint publications. I also want to thank Heinrich Römer and Manfred Hartbauer from the Institute of Zoology at Karl Franzens University Graz for the nice and inspiring collaboration that has lead to the results presented in Chapter 2.

My deepest thanks also goes to my current and former colleagues at the Institute for Theoretical Computer Science, in particular Oliver Friedl, Stefan Häusler, Daniela Potzinger, and Ingrid Preininger. I also want to acknowledge that this thesis would not have been possible without the financial support by Graz University of Technology, the Austrian Science Fund FWF, and the research programs of the European Union.

I would like to thank my family and friends for the endless support they have provided to me. I have been really lucky to have so many good friends by my side while writing this thesis. So a big "Thank you" goes to all of you - you know who I mean! I owe my deepest gratitude to my parents, who have supported me in all possible ways throughout my whole life. I will also always remember my grandmother Rosa, who passed away too early to see me finishing my doctorate. I am more than grateful to my beloved Judith, who has always been an invaluable source of joy and encouragement to me.

Contents

Chapter 1

Introduction

> Can the brain understand the brain? Can it understand the mind? Is it a giant computer, or some other kind of giant machine, or something more?
> — *David Hubel*, (1979)

Understanding the brain has always been considered one of the last frontiers for science. There is historical evidence that the anatomy and pathology of the brain had been studied already before Christ, but our knowledge of the function of the brain has taken a giant step forward in the twentieth century with the development of new tools for neuroscience. Large credit has to be given to advances in molecular biology and electrophysiology, but no other tool has arguably influenced our understanding of the function of our nervous system more than the computer.

As Hubel suggested in the above quote, the question may be asked whether the brain is actually all that different from a computer? Since computers are among the most complex machines ever built by humans, they may represent the best analogy that we have, and that we understand well enough. We know of many significant differences in the ways that current computers and nervous systems operate, e.g. the massive parallelism in the brain, but on a more abstract level one can also find obvious analogies: The brain receives input through sensory organs, processes the incoming information as well as information stored in the memory, and produces output e.g. by activating muscles. At the same time – to stress the analogy – an 'operating system' maintains the basic functions of our brain and body to supply energy and resources. The analogy should end here because computations performed by the brain are very different from computations inside our current computers. The human brain consists of billions of small processing units (*neurons*), which are organized in more or less densely connected networks, and compute in parallel without any obvious central synchronization mechanism. In addition, the synaptic connections between neurons constantly undergo changes in their connectivity patterns or transmission properties during lifetime. In contrast, the typical PC has a fixed architecture with one, or a small number of tightly synchronized central processing units, and small alterations of the hardware may result in a crash of the whole system. As a consequence, the tasks at which brains or computers excel are very different, and extremely hard to achieve for the other system, like e.g. controlling hundreds of muscles to walk over uneven terrain, versus searching billions of database entries in a fraction of a second to answer a search query. To date it seems impossible to simulate all the computations within a brain

on a single computer, and enormous supercomputers are required to simulate tiny cortical areas with reasonable precision (Markram, 2006).

We can however greatly gain insight into particular computational processes in nervous systems by building computer models that abstract apparently irrelevant details. In this thesis my focus is on concepts and methods from *machine learning* to analyze computations in nervous systems. Machine learning is the sub-field of computer science that is concerned with the construction of computer programs that autonomously extract information from data and improve with experience. It is a particularly useful tool for the analysis of neural computations for multiple reasons: Firstly, processing large amounts of structured data is obviously one of the major strengths of silicon computers, and machine learning allows us to analyze biological data from neural recordings in order to identify the key concepts that underlie neural information processing. Depending on the type of data that is available, one can analyze mechanisms at various levels of details, e.g. by investigating processes at the synaptic or neuronal level, or more abstract, cognitive processes, e.g. the behavior of animals or humans.

Secondly, the task of learning from data also plays a crucial role for biological organisms, because the number of neurons and synapses in brains is far too large to encode all information necessary for survival in the genetic code. Instead, organisms have to learn from different forms of feedback, which may also be corrupted by large amounts of noise. Already in the early days of machine learning, mechanisms like the perceptron (Rosenblatt, 1962) tried to find analogies between plasticity observed in biology, and mathematical models for learning in computers. This has lead to a fruitful interaction of disciplines. On the one hand, mathematical theories like e.g. liquid-state machines (Maass, Natschlaeger, & Markram, 2002), winner-take-all circuits (Douglas & Martin, 2004), population codes (Pouget & Latham, 2002), or spike-timing dependent plasticity (Bi & Poo, 1998), have helped in understanding learning and computation in neural systems, and allowed the design of new experiments that specifically aimed at verifying or falsifying those theories. On the other hand, machine learning algorithms (e.g. neural networks, genetic algorithms, reward-based learning, ...) have often been inspired by biology, and translated experimental observations into algorithms with high performance also on non-biological data.

In this thesis machine learning is used for a variety of tasks that investigate computations in nervous systems at different levels of details, from learning at a single synapse to modeling human behavior at a high level. New machine learning algorithms were developed based on known biological findings, and existing machine learning algorithms were applied to improve our understanding of biological data.

1.1 Organization of the Thesis

This thesis is comprised of 6 chapters which are based on publications to which I contributed during my PhD studies, and which have been published in competitive

journals and conferences.

In Chapter 2, unsupervised machine learning is used to study neural recordings. The data originate from spike train recordings in single auditory interneurons in acoustic insects, which were conducted under natural noise conditions in their natural habitat, which is the tropical rainforest. Recently developed techniques for clustering and spike metrics that yield quantitative similarity measures for different spike trains were used to identify patterns of activity in response to auditory stimuli. We found characteristic patterns of bursts in response to particular artificial stimuli, although the stimuli were disturbed by a very high noise level in the jungle. Our analysis showed a very high firing precision, even in the presence of severe background noise. We even found very similar responses of different insect preparations to identical stimuli, although the background noise conditions were completely different. This study is a good example of how machine learning algorithms, which were not particularly designed to model neural processes, can help us analyzing information processing in the nervous system.

Chapter 3 presents a study where supervised machine learning algorithms were used to predict human reading behavior from a number of high-level features. The task was to extract from the eye movements of human test subjects information about the relevance of sentences that they were reading. The data originated from an international machine learning contest organized by the PASCAL network of excellence and the Helsinki University of Technology (Salojärvi et al., 2005), in which our approach finished in first place. This study shows that with a good choice input features, established machine learning methods can be used to learn high-level models for human behavior.

A new algorithm for reinforcement learning, i.e. learning action strategies from rewards and punishments, is presented in Chapter 4. This algorithm draws inspiration from the neuroscientific concept of motor primitives (Mussa-Ivaldi & Bizzi, 2000), which represent simple sub-policies that can be combined to solve difficult control tasks. The hierarchical approach of splitting a difficult motor control task into a combination of multiple simpler tasks with pre-programmed primitive solutions allows a more efficient way of solving such problems. In neuroscience, motor primitives are e.g. used to represent modular force fields that control limb movements, and are activated by signals from the spinal cord. Our new reinforcement learning algorithms uses this concept for a difficult continuous control and planning problem. Combining theoretical ideas like sample-based models of the continuous environment, and innovative exploration policies, in combination with motor primitives leads to an algorithm, which learns very fast and learns close to optimal solution trajectories. This is a successful example of an idea from neuroscience that has greatly contributed to the improvement of existing learning algorithms for artificial systems like e.g. robots.

The final three chapters of this thesis investigate an interesting mathematical concept for learning synaptic weights with purely Hebbian mechanisms. Hebb's theory (Hebb, 1949) that potentiation occurs at synapses where presynaptic firing repeatedly induces postsynaptic firing is conceptually simple, and is well supported

by experimental data. We provide a theoretical framework that links these simple update rules to Bayesian optimality concepts and thereby provides strong convergence guarantees. These studies show that even though the computations performed in the basic elements of nervous systems (neurons and synapses) appear to be very simple and limited, a rich variety of tasks can be learned with the right choice of connectivity patterns for neural circuits, and plasticity rules in synapses.

Chapter 5 presents a pure Hebbian learning rule for a simple neural network architecture in a supervised learning context. We show that the simple learning rule provably converges to log-probability ratios of binary variables. In combination with a suitable sparse, redundant population code for input signals, this framework can in principle learn to infer optimal Bayesian decisions. In Chapter 6 we extend the results of Chapter 5 and present Hebbian mechanisms for the learning of decisions from reward signals. Weights in a winner-take-all (WTA) circuit, a type of neuronal microcircuit which is hypothesized to be of great importance for computations in the neocortex (Douglas & Martin, 2004), are trained with our reward-modulated learning rule and lead to near-optimal action selection performance.

Finally, the theoretical model developed in Chapter 7 identifies clustering as one possible task for spiking neurons in simple network models. We were able to relate experimentally observed neural architectures like WTA, and synaptic learning rules like Spike-Timing Dependent Plasticity (STDP), to the mathematical framework of Expectation-Maximization (EM). This is another good example of how seemingly unrelated biological observations and mathematical theories can be integrated to provide a plausible model for the function of neural circuits.

Chapter 2

Probing real sensory worlds of receivers with unsupervised clustering

Contents

The task of an organism to extract information about the external environment from sensory signals is based entirely on the analysis of ongoing afferent spike activity provided by the sense organs. We investigate the processing of auditory stimuli by acoustic insects, which extract behaviorally relevant information from minute variations of stimuli, and therefore require a particularly precise neural encoding. Interneurons in the auditory system of insects encode stimulus features by bursts of spikes, which are more reliable sources of information than single spikes, but still exhibit significant spike train variability. For this study, we recorded for multiple hours from an identified acoustic interneuron directly in the natural habitat of the insect in the tropical rainforest, which yields a representative sample of neural responses to sounds that the acoustic receivers experience in the real world. In contrast to typical recordings in sound proof laboratories, strong environmental noise from multiple sound sources interferes with acoustic signals in these realistic scenarios. Still we find very precise firing patterns for bursts elicited by acoustic stimuli. This demonstrates how acoustic interneurons can communicate large amounts of information to the central nervous system under natural conditions. We explore a recently developed unsupervised machine learning algorithm based on probabilistic inference to find frequently occurring firing patterns in the response of the acoustic interneuron. Our approach learns to distinguish burst responses to environmental noise from responses to a set of artificial acoustic stimuli without the help of an external supervisor. The reliability of burst coding in the time domain is so high that spike patterns in response to identical stimuli show a high degree of similarity

for different preparations from different nights, and also for recordings that were either performed in the sound proof lab, or under realistic conditions in the rainforest. We analyze simultaneous recordings from the same identified neuron in two preparations under real world conditions and observe that temporal firing characteristics of bursts are largely preserved among individuals of the same species, but not exact spike times or average firing rates over larger time windows. We discuss our findings with respect to the reliability of signal classification and discrimination of receivers under real world conditions.

2.1 Introduction

In order to fulfill its task of shaping the behavior of organisms, the sensory system and the brain have to rely on information about the "outside" physical world, provided by the sense organs, which respond to different forms of energy. The information is transmitted via afferent nerves and encoded in trains of action potentials. The brain, by decoding this information, has to make adaptive assumptions about what had happened in the physical world. A central issue in sensory physiology deals with the coding and decoding mechanism(s) in the sense organs and CNS, respectively. Whereas early work concentrated on information provided by the average spike count over an appropriate time window (or firing rate in AP'/s), it soon became clear that codes using the precise timing of action potentials would make more efficient use of the capacity of afferent lines to the brain. Yet, the mechanisms by which stimuli are represented in the timing of spikes are still not fully understood (Rieke et al., 1997; Eggermont, 1998; Lestienne, 2001).

Irrespective of the sensory system investigated, recordings of single sensory neurons, or first-order sensory interneurons, always reveal isolated spikes and spikes grouped as bursts, i.e. short episodes of high-frequency AP firing (e.g. Krahe & Gabbiani, 2004; Eggermont & Smith, 1996; Metzner et al., 1998). These bursts - in contrast to single spikes - have been suggested to have particular importance for the function of the brain (review Lisman, 1997), and in sensory systems bursts convey the important stimulus features (Marsat & Pollack, 2006; Metzner et al., 1998). Yet, the problem of extraction of characteristic features within these bursts for identifying stimulus features and for object classification is difficult because spike trains exhibit variability. For insects and the acoustic modality, Ronacher et al. (2004) reviewed the sources for such variability, and how it affects the processing of temporal patterns of acoustic signals. For example, one important source for such variability in the auditory modality results from the fact that in real world situations individuals are exposed to multiple sound sources, originating from different locations, or that signals are degraded and attenuated on the transmission channel between sender and receiver (Wiley & Richards, 1978, 1982; Morton, 1975; Römer & Lewald, 1992; Römer, 1998). Internal noise as a result of stochastic processes within the nervous system is a further source for spike train variability.

As a result of the unavoidable noisiness of spike trains in neurons of sensory

pathways one should expect the evolution of mechanisms in the nervous system leading to a reduction of the effects caused by false stimulus feature extraction and/or classification due to noise. On the other hand, minute variations in spike trains may well reflect differences between objects or object classes which are important for the receiver, such as small differences in the size of a sender, or the loudness or frequency composition in the sound signal of a mate. Such small differences, in contrast to those caused by noise, should be preserved during sensory processing, since they represent the neuronal basis for discrimination between mates or other decisions of importance for the receiver.

If bursts of action potentials contain the information about relevant features of objects or object classes, it should be possible to unambiguously distinguish 1) bursts of spikes elicited in response to a given stimulus from those bursts which resulted from noise, and 2) from bursts elicited in response to stimuli with different features. Various attempts have been made in the past to identify algorithms for such a task.

In this paper we present a set of machine learning tools to analyze and discriminate burst data without losing information about precise firing times, which is crucial within the auditory system. Spike train data typically comes in the form of sequences of firing times of variable length, which is not compatible with the requirements of traditional machine learning methods to receive inputs in the form of fixed-size, real-valued vectors. One can circumvent this problem by discretizing the spike trains into time bins, or extracting sets of numerical features, but both methods inevitably lead to a loss of information and temporal precision. Preserving temporal precision in spike timing is a necessary prerequisite for the analysis of auditory neural codes, where fast temporal fluctuations provide important information about the nature of the incoming stimulus. One way to analyze neural data without further preprocessing, is to use non-parametric machine learning methods (see e.g. Narayan et al., 2006; L. Wang et al., 2007). These methods require similarity measures for spike train data, which work directly on the exactly measured spike times. A number of spike metrics (e.g. Victor, 2005; Christen et al., 2006; van Rossum, 2001; Schrauwen & Campenhout, 2007), and kernel functions (Shpigelman et al., 2003; Eichhorn et al., 2004) for spike data have been proposed for this purpose.

These methods were designed for supervised classification or regression, and therefore require a labeling of the spike trains by an external supervisor. The same ideas, however, can be used for unsupervised learning methods, where only statistical differences of firing patterns are considered for the formation of different classes or clusters. One of the most promising techniques for clustering is the recently developed affinity propagation algorithm (Frey & Dueck, 2007), which is based on principles from probabilistic inference, and has lead to excellent results for a number of large datasets. Our study presents the first application of affinity propagation to the discovery of groups of related bursts, by combining it with the spike metric proposed by Victor (2005). This allows us to find meaningful spike patterns also in the responses to environmental noise signals, which may carry information about the identity or location of different sound sources.

In this study we present data from a model system using an identified neuron approach in an acoustically communicating insect. This system offers several advantages for studying sensory burst coding over previous ones: 1) All recordings stem from the same identified neuron (called omega-neuron; Molina & Stumpner, 2005) in different preparations. 2) The first-order neuron in the auditory pathway integrates sensory information from a very limited number of receptor cells in the ear (20 - 40 receptors). 3) Recordings can be obtained for several hours, and most importantly, 4) portable preparations have been developed to make recordings directly in the insects' natural environment, such as the tropical rainforest (Rheinlaender & Römer, 1986; Römer & Lewald, 1992; Lang et al., 2005). This enables to study sensory coding under the most natural conditions possible. Broadcasting well defined acoustic stimuli from some distance to the preparation, while recording the response of the neuron to these stimuli and to the background noise allows us to gain new insights about the characteristics and reliability of burst coding.

Our results indicate that the omega-neuron communicates through bursts of spikes large amounts of information about acoustic stimuli to the central nervous system, even in the presence of strong environmental noise in the natural habitat. The temporal characteristics of bursts in response to identical stimuli are well preserved under natural or laboratory conditions, as well as for different preparations, although exact spike times or average firing rates over longer time windows are quite different. The presented techniques for clustering of bursts of spikes promises to be potentially very useful for the analysis of spike train data from various other neural systems.

2.2 Material and Methods

2.2.1 Animals and Physiological Preparation

A total of 27 adult male and female bushcrickets (*Docidocercus gigliotosi*) were used for this study. We recorded the activity of an identified auditory interneuron, the so-called omega neuron, in the field, using a technique introduced by Rheinlaender and Römer (1986) and Römer and Bailey (1986). The morphology of the cell, as revealed from intracellular dye injection, is shown in Figure 2.1 (inset). It is a local neuron in the prothoracic ganglion and receives excitatory input from almost all of the 20 - 40 receptors in the hearing organ (Römer et al., 1988). The tuning of the cell reflects the broad-band hearing sensitivity of the insect, matching both the frequencies of the conspecific calling song, and ultrasonic frequencies up to 100 kHz, thus including bat echolocation calls as well. As in other bushcricket species, the sensitivity of auditory receptors in *D. gigliotosi* differs by only a few dB from the sensitivity of the omega cell at most frequencies except below 5 kHz (Römer, 1985). Furthermore, in response to a stimulus above its threshold, the neuron fires bursts of action potentials and copies the temporal pattern of an acoustic stimulus in a tonic manner. Altogether these attributes make outdoor recordings of the activity of the omega cell very suitable for studying sensory coding under realistic,

i.e. outdoor conditions in the animals' own natural habitat.

The methods of the preparation and for obtaining extracellular action-potential recordings of the neuron have been described in detail by Römer et al. (2002). In short, the prothoracic ganglion was surgically exposed in a preparation ventral side up and the tip of an electrolytically sharpened tungsten electrode ($0.7 - 1.3M\Omega$ resistance) was inserted into the anterior part of the ganglion, slightly lateral to where the neurite of the omega-neuron crosses the ganglionic midline (see Figure 2.1). Then the opening in the cuticle was sealed with petroleum jelly to prevent dessication.

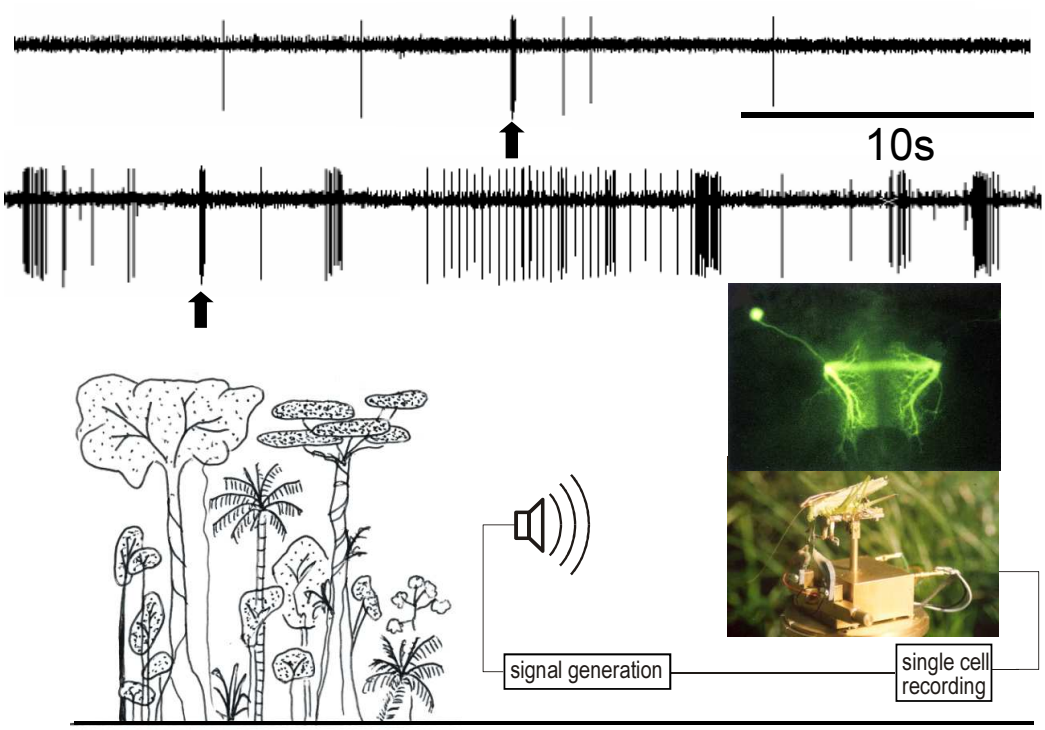


Figure 2.1: Experimental arrangement for long-term recordings of single cell activity in the tropical rainforest, and examples of recordings made at about one hour before sunset (upper line), and 45 minutes after sunset, when the background noise level had increased from 40 dB SPL to 65 dB SPL. Note that in the low noise situation only a stimulus (arrow) elicited a short burst of spikes, whereas after sunset the neuron fires many bursts also in response to the acoustic background. The inset shows the morphology of the cell within the prothoracic ganglion after intracellular dye injection (upper part), and a prototype of the portable preparation.

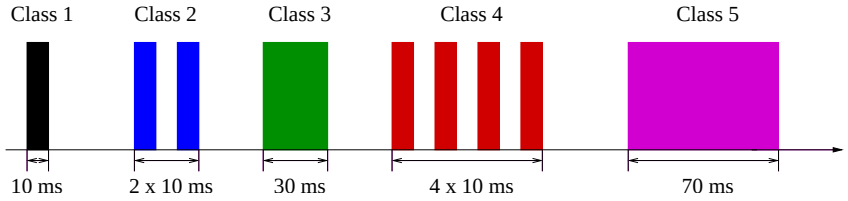


Figure 2.2: Illustration of the five stimulus classes played to bushcrickets during experiments. 1) Single pulse of 10 ms; 2) double pulse with 10 ms duration each, separated by an interval of 10 ms; 3) 30 ms pulse; 4) four repetitive pulses, 10 ms each, separated by an interval of 10 ms; 5) 70 ms pulse.

2.2.2 Study site and experimental procedure

The study was conducted on Barro Colorado Island (BCI), located in central Panama within Gatún Lake, part of the Panama Canal. Data collection took place in February/March and June/July 2002, 2003 and 2004. *D. gigliotosi* is a tropical insect living predominantly in the rainforest understorey, and all its activity, including acoustic communication, is restricted to the night. Thus, all recordings were made during times after sunset (about 6 p.m. local time) except for control measurements (see below and Figure 2.1).

First, the preparation was tested for intrinsic spontaneous activity and for its response to the stimuli without background noise in an anechoic chamber in the laboratory. Background noise level in this chamber was below 30 dB SPL and thus below the threshold of the omega-cell preparation. Five different stimuli were broadcast through a speaker (TW8 spezial) at an intensity of either 10 or 20 dB above the threshold of the preparation. They consisted of the same carrier frequency of 20 kHz, but differed in duration and the number of pulses (see Figure 2.2). The rise- and fall-time of all sound pulses was 1 ms. Stimulus intervals were 10 seconds, which is within the range of the naturally occurring intervals in the calling song of this insect (Lang et al., 2005). Action potential responses of the omega cell were digitally recorded at a rate of 20 kHz together with the trigger for a stimulus, on separate channels of a data acquisition system (PowerLab, ADInstruments Inc.).

After completion of the control experiments indoors, the preparation with the single cell recording was mechanically stable enough to be transferred to a position within the rainforest about 200 meters from the lab, and fixed at a distance of 2m from a speaker at a height of 1m from the ground. The same stimulation regime as in the laboratory was used for outdoor recordings. Since some recordings lasted for several hours (max of 9.5hrs), at the end of a stimulation regime we controlled for a change in sensitivity of the preparation. If the sensitivity was decreased by 5 dB or more (which happened in four cases), the recording was discarded.

2.2.3 Burst Detection

The goal of the burst detection mechanism is to extract from several hours of spike recordings those short segments in which the omega neuron is bursting. A burst in a spike train can be qualitatively defined as a short sequence of spikes with high firing rate, separated by time windows of low or no firing. There is no exact mathematical definition of what constitutes a burst, and many different approaches for burst detection in a sequence of spikes have been proposed (e.g. Cocatre-Zilgien & Delcomyn, 1992; Turnbull et al., 2005; Gourevitch & Eggermont, 2007; Chiappalone et al., 2005). Furthermore, every approach needs to be slightly tuned to the parameters of the neurons under investigation, because different neurons may have slightly different background firing rates or refractory periods.

For our study we defined a heuristic set of rules to extract bursts from the recordings, which is similar to the method used by Cocatre-Zilgien and Delcomyn (1992). Before a burst starts, there must be a time window of at least 60 ms in which no spike occurs. The first spike of a burst must be followed by another spike no later than 15 ms afterward. The end of the burst is detected when the first time interval of 30 ms or longer occurs, or if two consecutive intervals combined are longer than 45 ms. A burst is only accepted as such if it contains at least 5 spikes and is longer than 8 ms. Figure 2.3 illustrates the criteria that define a burst.

This set of rules could reliably extract all bursts from the recordings, which are often obvious from visible inspection. A variant of this algorithm has been used for previous studies (Lang et al., 2005). Other burst extraction methods did not lead to a (subjectively) better performance.

When artificial stimuli were broadcast to the animals, most stimuli were followed by a burst in the omega neuron with a short latency of about 10 ms. We associated stimulus and burst if the onset of the burst occurred not more than 50 ms before or after the onset of the stimulus (the burst may start before the stimulus if the burst detection algorithm includes spikes elicited by background noise immediately before the stimulus associated burst begins). Every burst was assigned one of 6 class labels: it is either associated with one of the five different stimuli (see Figure 2.2),

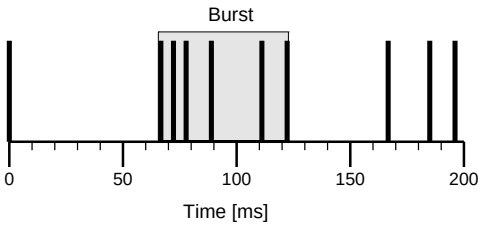


Figure 2.3: Detection of bursts in spike trains. The 6 spikes in the shaded area constitute a burst, because they are separated by time window of at least 60 ms from the first spike, the interspike-interval is never larger than 30 ms, the burst duration is longer than 8 ms and there are more than 5 spikes.

or it is a noise burst, in response to a random signal in the acoustic background.

2.2.4 Spike Metrics

In order to find re-occurring burst patterns in neural signals, one needs an objective measure of how similar two bursts are. Many different similarity measures for sequences of spikes have been proposed in the literature (e.g. Victor, 2005; Christen et al., 2006; van Rossum, 2001; Schrauwen & Campenhout, 2007). We chose to use the spike-time metric by Victor (2005), which computes the minimal costs of transforming one spike train into the other. The algorithm applies three operators with variable costs for the transformations: Insertion or deletion of a spike at an arbitrary time point constitutes a cost of 1. The third operator is a shift of a single spike, which has a cost of $q \cdot \Delta$, where Δ is the duration of the shift, and q is a parameter that needs to be defined in advance. High q will tend to prefer insertion and deletion of spikes to shifting, so small spike-time differences have a large influence on the distance. Low q, on the other hand, is more tolerant with respect to small spike-time differences, so the dominating factor is the difference in the number of spikes in the two bursts. Shifting is preferred to insertion and deletion as long as two spikes occur within an interval of $2/q$ seconds (Victor, 2005).

One problem with the automatic detection of bursts is that spikes that are results of background noise, rather than responses to the stimulus that caused the burst, may occur shortly before the beginning of a burst. These spikes cannot be separated from the rest of the burst if they fulfill all criteria for a burst spike. For the metric proposed by Victor (2005), a burst that is temporally shifted because of noise spikes before the actual burst can look very different from another burst with the same temporal pattern, but without initial noise spikes. We therefore modified the metric of Victor (2005) and introduced another operation, the *burst-shift* operator. The burst-shift operator can delete up to N_{shift} initial spikes from every burst for a cost of 1 per spike, and adds the distance between the remaining spike trains, which is computed by the metric from Victor (2005). For our purposes we set $N_{\text{shift}} = 5$. This compensates occasional unavoidable errors in the burst detection process.

For two spike trains A and B the distance $D_q^n(A, B)$ between the two spike trains is defined, using the cost factor q and $N_{\text{shift}} = n$ (for concise notation, q and n will be omitted wherever obvious). For two sets of spike trains $\Sigma = \{A_1, \ldots, A_{n_1}\}$ and $\Theta = \{B_1, \ldots, B_{n_2}\}$, we define a $n_1 \times n_2$ distance matrix $\mathbf{D}(\Sigma, \Theta) = (d_{ij})$, where $d_{ij} = D_q^n(A_i, B_j)$. For a single set of bursts Σ the matrix $\mathbf{D}(\Sigma) = \mathbf{D}(\Sigma, \Sigma)$ yields all distances between bursts within the same data set.

2.2.5 Clustering with Affinity Propagation

Clustering is an unsupervised machine learning technique that finds groups of related data objects, based on a measure of similarity or distance. Many of the standard clustering methods, like e.g. k-means (Hastie et al., 2001), are not suit-

able for clustering spike train data, because they work only in Euclidean space, and require direct manipulations of the data points, e.g. for computing means of groups of data points. For neural recordings it is not defined what the "average" spike train of a group of multiple spike trains is. Clustering a set of spike trains Σ therefore requires a clustering algorithm that works only on the matrix of spike-train distances $\mathbf{D}(\Sigma)$, or equivalently the similarity matrix $\mathbf{S}(\Sigma) = -\mathbf{D}(\Sigma)$. There are several clustering algorithms that meet this requirement, such as e.g. k-medoids, hierarchical clustering methods (Hastie et al., 2001) or spectral clustering (Ng et al., 2001). For this study the affinity propagation algorithm by Frey and Dueck (2007) was chosen, which is very fast and reliable, and which has exhibited superior performance over comparable methods on a variety of datasets.

Affinity propagation defines every cluster through one central data point, the cluster *exemplar*. The algorithm assigns all data points to clusters in order to minimize an energy function based on the similarity between data points and their assigned exemplars. The number of clusters, k, does not have to be specified in advance. In contrast, the algorithm initially considers all data points as exemplars, and then iteratively applies a message passing algorithm on a factor graph representation (Kschischang et al., 2001) of the data set, until a suitable set of clusters and exemplars emerges. The factor graph and the messages that are passed along the edges are directly derived from the similarity matrix $\mathbf{S}(\Sigma)$. Messages between nodes indicate the affinity that a data point has for choosing another point as its exemplar, and the "availability" of single data points to become exemplars.

The number of clusters can be implicitly controlled by scaling the diagonal entries $s_{i,i}$ of the similarity matrix, which define the preference of a data point to choose itself as its exemplar. Larger self-preference values lead to a larger number of clusters. While normally the distance of a spike-train to itself is 0, a successful strategy in practice is to use the median of similarities $\bar{M}_i = median(s_{i,j}, j = 1, \ldots, n)$ as the self-preference $s_{i,i}$, which leads to a moderate number of clusters. In practice we always used a multiple $\alpha \cdot \bar{M}_i$, where α is a constant between 1 and 20 to produce a smaller number of clusters. In contrast to other methods that require the exact number of desired clusters as input, this method is much easier to tune to obtain satisfactory results. The method is also insensitive to random assignments of cluster exemplars, so it is not necessary to run the clustering algorithm multiple times and choose the best clustering.

To visualize the results of the clustering algorithm, the clusters are arranged in a dendrogram, applying the group average hierarchical clustering method (Hastie et al., 2001) to the distances of the cluster exemplars. Clusters with similar exemplars are then grouped together in the same branch.

2.3 Results

2.3.1 Recordings in Natural Habitats

A typical result for the effect of background noise on sensory coding is shown in Figure 2.1. The receiver was placed within the rainforest at 17.00 hrs 2m from a speaker broadcasting a single sound pulse of 10ms, at a sound pressure level of 20 dB above the threshold of the cell. Since a female has no a priori knowledge about the presence of a male signal, her only information about a signal is encoded in afferent nervous activity such as the one shown in the upper recording. Before sunset each burst of action potential activity was caused by a stimulus. A detection criterion based on bursts of action potentials or the corresponding increase in spike rate would give "hits" in term of signal detection (Green & Swets, 1966). Indeed, in all cases when there was an acoustic signal during the experiment at 17.00 hrs, there was bursting activity in the nerve cell and there was no, or only single spike spontaneous activity when a signal was absent, therefore there were no "misses" or "false alarms" respectively.

After sunset, however, this ideal situation for signal detection changed due to the strong increase in background noise. The same preparation at exactly the same position in the rainforest now exhibited high action potential activity (Figure 2.1, lower trace), and only an a priori knowledge of the time of signal presentation (arrow) would allow correct detection of the stimulus. Using the same detection criterion as in the situation before sunset would result in many false alarms (i.e. identifying background noise as signals).

Apparently, analyzing neural signals recorded under natural conditions poses new and different challenges in comparison to laboratory experiments. The background noise mainly constitutes the communication activity of different individuals and species of insects, frogs and vertebrates, and their acoustic signals may be of different importance for the survival and reproductive success of the receiver. Some of the noise comes from individuals of the same species of bushcricket, which may be potential mates or rivals, but the majority of noise comes from heterospecifics with no relevance. A third category may be sound originating from predators, such as bats.

An analysis of a typical recording of a longer sequence of spike activity is shown in Figure 2.4, and it is obvious that there are considerable fluctuations in firing and bursting rates, mainly due to background noise. In this example, firing rates vary from 7 Hz to 17 Hz over the time period of 200 minutes of recording, and burst rates vary from about 0.2 to 0.8 Hz. The curves for firing and bursting rates are visibly correlated (correlation coefficient $\rho = 0.37$).

2.3.2 Analysis of Bursts

We analyzed the recordings from different dates to find bursts and optimize the parameter of the spike metrics. In Figure 2.5A we show as an example the analysis of one particular recording (Feburary 22 2004). The joint interspike-interval (ISI)

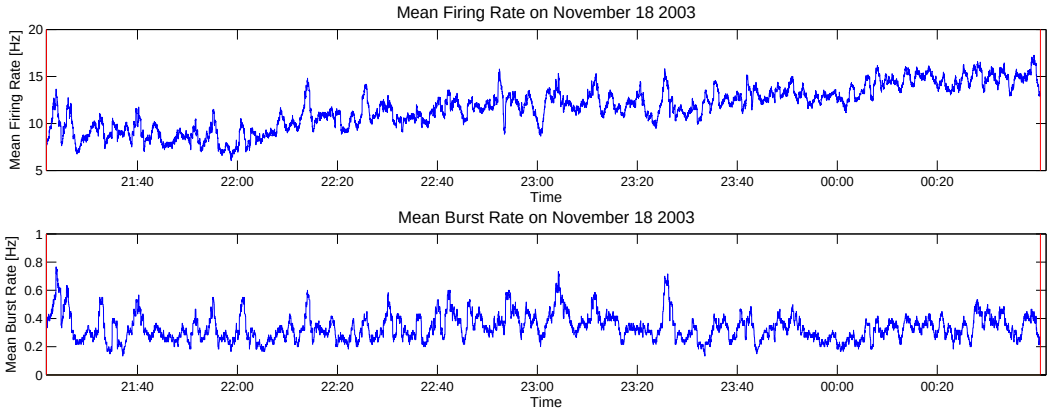


Figure 2.4: Firing (top) and bursting rates (bottom) of the spike activity of the omega-neuron from 21.20 hrs to 0.40 hrs at night (November 18; 2003) in the natural habitat. The fluctuation in both rates is high, but firing and burst rates are correlated with a correlation coefficient of $\rho = 0.37$. The mean firing rate over the entire night is 11.5 Hz, and the mean bursting rate is 0.33 Hz.

diagram, which shows the duration of the next ISI as a function of the preceding ISI, one can see the presence of bursts in the recordings. The accumulation of points in the lower left corner shows that there are numerous periods of fast firing, preceded or followed by longer ISIs. The intervals between bursts display no clear pattern, but the histogram of inter-burst intervals in Figure 2.5B shows that most intervals are short, and the frequency of longer inter-burst intervals decays. The bursts in response to the five stimulus classes can be aligned to the onset of the stimulus. Figure 2.5C and D show the bursts that follow these stimuli as PSTHs and as firing rate histograms, respectively.

A comparison of the bursts of three different preparations and recording sessions, when aligned to the onset of the stimuli (Figure 2.6) indicates qualitatively that the firing patterns are very similar, with slight variations in the latency of the burst or the variability of firing.

2.3.3 Clustering of Bursts

For the first experiment we used recordings in which artificial stimuli were broadcast to the preparations in their natural habitat after sunset. The five artificial stimuli used for playbacks differed in duration and temporal structure (see Figure 2.2). Bursts were extracted from multiple hours of spike data recordings, and bursts at the time of the onset of the artificial stimulus were labeled with the class of the associated stimulus. The labels of the bursts were only used for evaluation purposes, but were not provided to the clustering algorithm, since we were interested whether a purely unsupervised method could reconstruct the groups of bursts according to

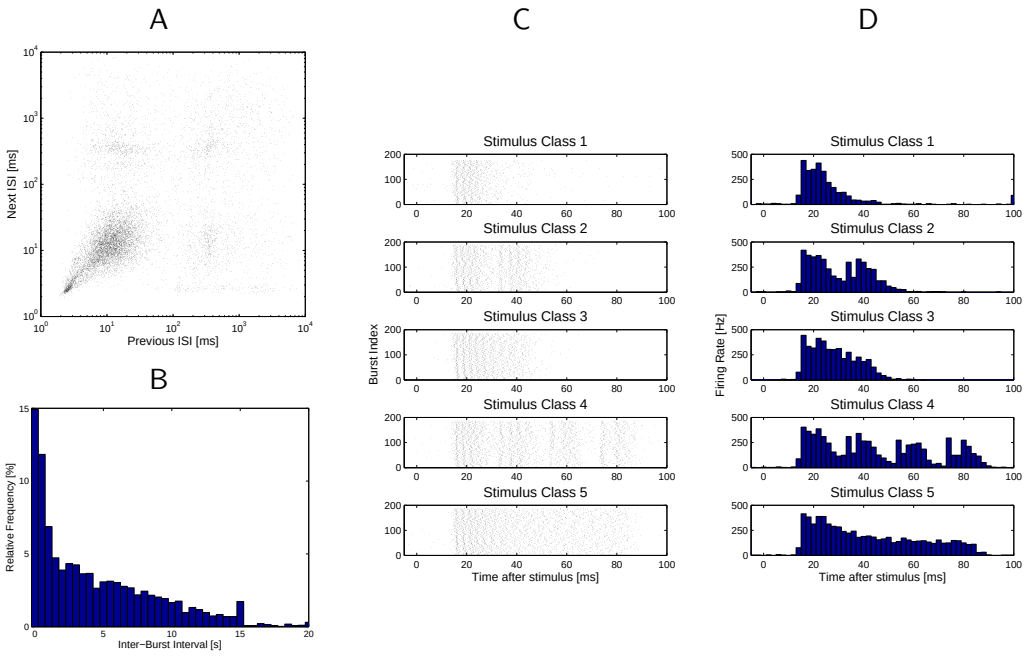


Figure 2.5: Analysis of bursts extracted from the spike data. **A)** The joint interspike-interval plot for recordings from February 22; 2004 indicates the presence of bursts by a large cluster of points in the lower left corner, which represents periods of fast firing. **B)** Histogram of inter-burst intervals (bin size: 0.5 s). **C)** Bursts in response to the five artificial stimuli, plotted aligned to the onset of the stimulus. **D)** Peri-stimulus time histograms (bin size: 2 ms) for the five classes of artificial stimuli.

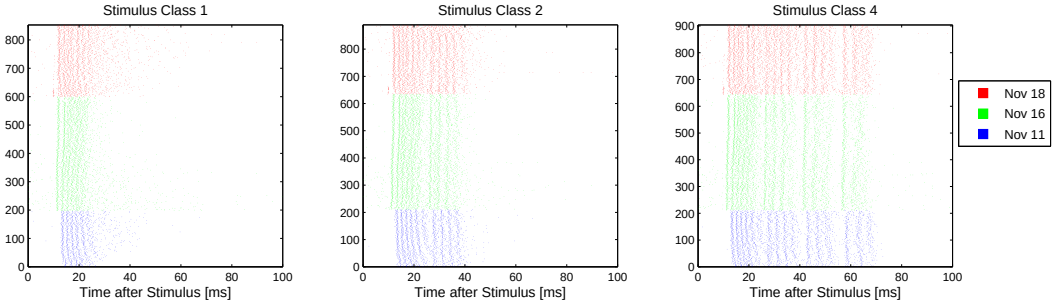


Figure 2.6: Bursts in response to three artificial stimuli, recorded at different recording dates with three different insect preparations and aligned to the stimulus onset. One can see a clear similarity of the responses, but also different latencies and variabilities of firing.

the stimulus that caused the neural response.

The result of the clustering is illustrated in Figure 2.7. Here the distance matrix is shown before and after the clustering process (dark values show high similarity). Before the clustering, the distance matrix for 1000 randomly picked bursts is displayed, where the ordering of the burst indices corresponds to the order in which they were recorded. After the clustering, the order of the bursts is rearranged, and one can clearly observe groups of bursts that cluster homogeneously together, and have larger distance to other groups of bursts.

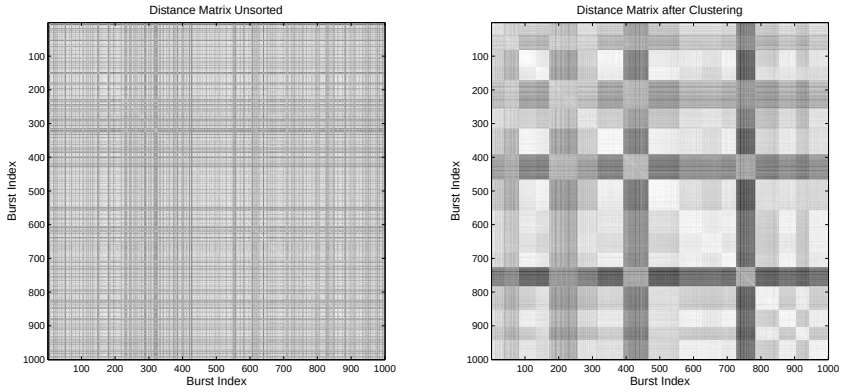


Figure 2.7: Distance matrices before and after clustering from recordings from the natural habitat on February 22 2004. Light pixels indicate high similarity of bursts, whereas dark pixels show larger distances. Clearly, groups of bursts are formed by the clustering process.

Figure 2.8 shows the resulting groups of bursts, drawing bursts following stimuli in red, and bursts as a result of background noise in black. The labels of some clusters are very homogeneous, in particular those in the upper part of the dendrogram with clusters of long and relatively unstructured bursts, which are almost exclusively bursts in response to background noise. These bursts are grouped together because they have a similar mean firing rate and a similar number of spikes, although their firing patterns do not exactly match. Two other groups of homogeneous clusters are comprised almost exclusively of bursts in response to two artificial stimuli (classes 4 (a four-pulse-stimulus) and 5 (a pulse of 70ms duration) in Figure 2.2); only rarely do we find in the same cluster bursts not elicited by these stimuli. In some cases, however, the unsupervised clustering algorithm produced inhomogeneous clusters, which include both bursts in response to stimuli as well as bursts in response to the background noise. This is true for the two bottom clusters in Figure 2.8A, but also for two clusters with bursts in response to the shorter 30 ms pulse and the two-pulse stimulus, which cluster together with background noise bursts. Figure 2.8B shows a clustering for another recording with a different preparation and night, and again it can be seen that bursts following one of the longer or more structured artificial signals (classes 4 and 5) fall into more homogeneous clusters than bursts

after stimuli with shorter pulses. One can further observe in both plots that there are some very homogeneous clusters of bursts with precisely timed firing patterns in response to unidentified background noise events.

2.3.4 Separability of Artificial Stimulus Classes

Figure 2.8 demonstrates that bursts in response to particular classes of artificial stimuli form very homogeneous clusters, e.g. bursts in response to the four pulse pattern (class 4). On the other hand, some stimulus evoked bursts are mostly mixed together with bursts in response to background noise in the habitat, or bursts in response to different stimuli. We evaluated this separability property of stimulus evoked bursts over data from six recordings sessions, of which three used all 5 stimulus classes, and three contained only stimuli of classes $1, 2$, and 4.

As a measure of homogeneity we used the mutual information between the cluster index and the class label. The mutual information is an information-theoretic measure, which computes for two random variables X and Y (here assumed discrete), the amount of information that one variable contains about the other. Formally, the mutual information $I(X;Y)$ is defined as the difference between the entropy $H(X)$ of X and the conditional entropy $H(X|Y)$, and can be calculated as

$$I(X;Y) = \sum_{y \in Y} \sum_{x \in X} p(x,y) \log \left(\frac{p(x,y)}{p(x)p(y)} \right) \quad . \tag{2.1}$$

The mutual information is high if knowing one variable reduces the uncertainty about the other. In our case this would mean that knowing the cluster label should provide high information about the classes of bursts that are found within the cluster. For every stimulus class c we measure the mutual information between the variables L, which indicates the cluster label for all clusters that contain at least one burst of class c, and C_c, which indicates whether a burst b belongs to class c (in that case $C_c(b) = 1$), or to another class ($C_c(b) = 0$). For each of the six recording sessions we compute the clustering, and measure the mutual information $I(L;C_c)$ separately for every stimulus class. Figure 2.9 shows the average results over all recordings for every class. Although these statistics are based on only six recording sessions, and the standard deviations are large, one can observe the same trend that was qualitatively visible from Figure 2.8. The mutual information is high for classes of bursts in response to long and/or temporally structured stimuli (classes $2, 4$, and 5), and much lower for the single pulses of classes 1 and 3. This indicates that classes $2, 4$, and 5 can be better discriminated from other artificial or background signals than the single pulse stimuli. Due to the limited amount of available data, these results are statistically not significant, and more measurements would be required.

2.3.5 Matching Clusters of Bursts

The two clustering results in Figures 2.8A and 2.8B indicate that similar clusters of bursts can be found in both recordings, even though the recordings stem from dif-

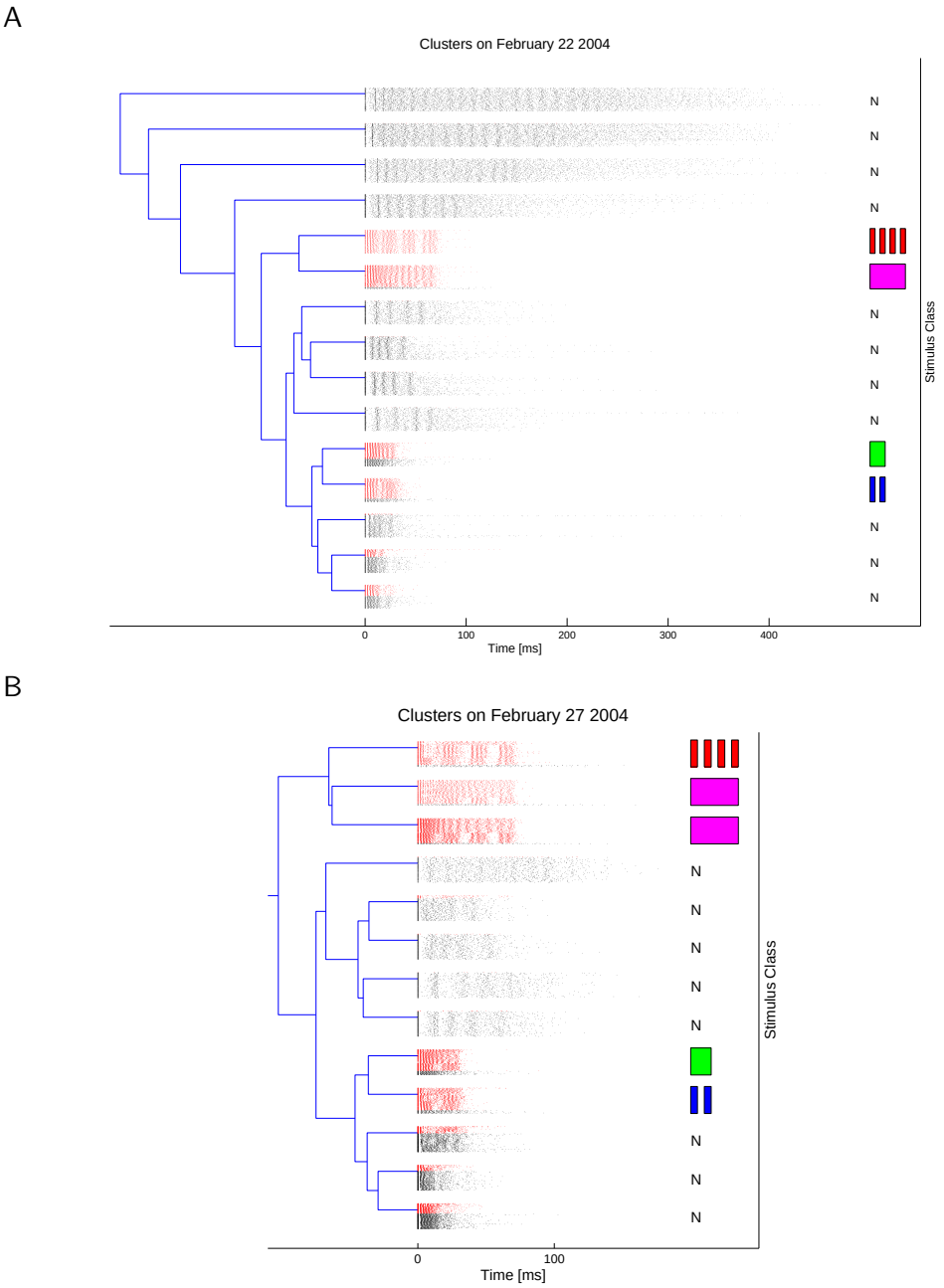


Figure 2.8: Clusters of bursts from a recording in the natural habitat (**A**) February 22 2004, **B**) February 27 2004). Bursts associated with artificial stimuli are plotted red, bursts associated with noise are plotted in black. On the right is an illustration of the stimulus that caused the bursts, or *N* if the cluster mainly consists of bursts resulting from noise signals. The clusters are arranged hierarchically, grouping clusters with similar exemplars together. Longer and more structured bursts form more homogeneous groups than bursts after short pulse signals (e.g. clusters at the bottom of diagram B). Clusters in A) contain between 140 and 501 bursts, and between 145 and 328 bursts in B).

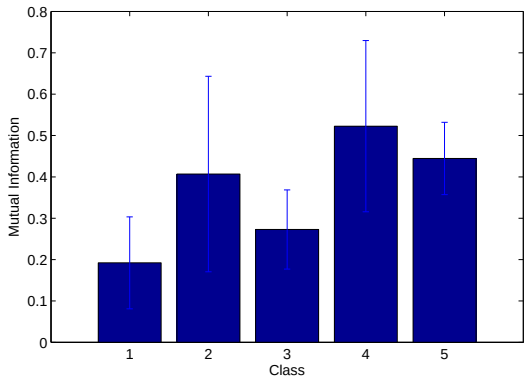


Figure 2.9: Mutual information between cluster indices and class labels for every class of artificial stimuli, averaged over six recording sessions (errorbars denote standard deviations). Classes of bursts with higher mutual information form more homogeneous clusters. Artificial stimuli that consist of temporally more structured and/or longer stimuli (classes $2, 4$, and 5) are better separable from noise or other stimuli than single pulse stimuli (classes 1 and 3).

ferent preparations and different recording sessions. Furthermore, the background noise in the natural habitat is not constant over the recording periods, since different sound sources may be present and located at different positions in comparison to other recording sessions in different nights, and even years. We therefore searched for burst clusters from one recording session that have corresponding clusters of bursts in different recording sessions with similar firing patterns. Starting from the previously computed clusterings of bursts from single recording sessions, the spike-time metric described in methods was used to calculate distances among cluster exemplars from different sessions. For every cluster in one recording the best-matching cluster in the other recording was then computed. A high similarity of two clusters in different sessions would suggest that they contain bursts in response to the same type of sound source. Furthermore, for several preparations we recorded the response to artificial stimuli in the laboratory, and for others outdoors. Hence, the similarity of clusters in response to the same stimuli under two different acoustic conditions could be analyzed. Obviously these recording conditions are very different, because the majority of bursts in outdoor recordings stems from background noise, while in the laboratory bursts occur almost exclusively in response to artificial stimuli.

In Figure 2.10 the clusters of bursts found in a laboratory experiment, in which only artificial stimuli were broadcast, were matched to clusters from outdoor recordings. As can be seen from a comparison of clusters in response to artificial stimuli type 1, 3 and 4 (compare with Figure 2.2), there are close matches of laboratory-burst clusters to clusters from outdoor recordings. On the other hand, the responses to the four-pulsed stimulus in the laboratory condition reveal a more precise timing of spikes within the burst compared to the responses recorded outdoors. This indi-

cates that the specific acoustic conditions of the noisy nocturnal rainforest caused some changes in this timing within bursts.

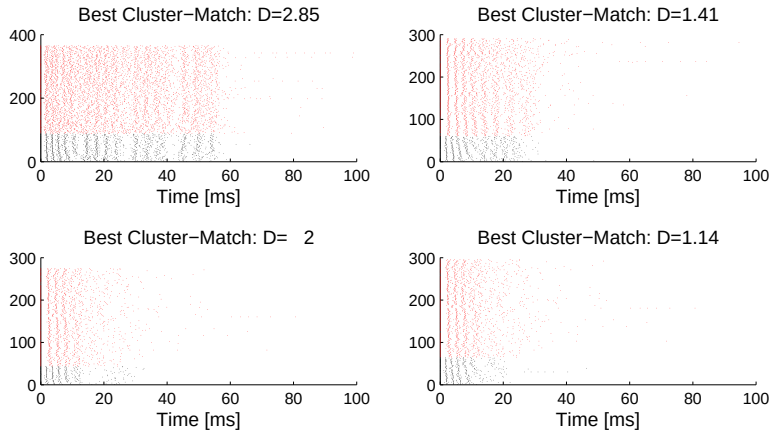


Figure 2.10: Clusters of bursts from laboratory recordings (black; November 11 2003), matched to clusters of bursts from outdoor recordings (red; November 18 2003 and February 22 2004). For these laboratory clusters, closely matching clusters can be found in the outdoor recordings. D defines the distance between the cluster exemplars under the spike metric.

In a similar way we matched clusters of bursts from different outdoor recording sessions, in which the activity of omega neuron from different animals was recorded at different nights (sometimes in different years). The results in Figure 2.11 show that also under these conditions it is possible to find close matches for some clusters of bursts. Comparing the similarity indices D in Figures 2.10 and 2.11 would indicate that some of the cluster matches between different animals in different outdoor recording conditions are closer than the cluster matches between outdoor and laboratory recording conditions.

On the other hand, the cluster matching procedure also revealed several clusters of bursts for which no good match in the other recording session was found. This holds in particular for clusters of long bursts without clear temporal structure. Figure 2.12 shows some examples of such 'matchings'. Notice the substantially higher distance value D between cluster exemplars, which is due to the fact that these bursts include more spikes, and so more shifts or insertions may have to be made in order to transform one spike train into another. Bursts within these clusters do not show the precise temporal signature that could be observed in the previous analysis.

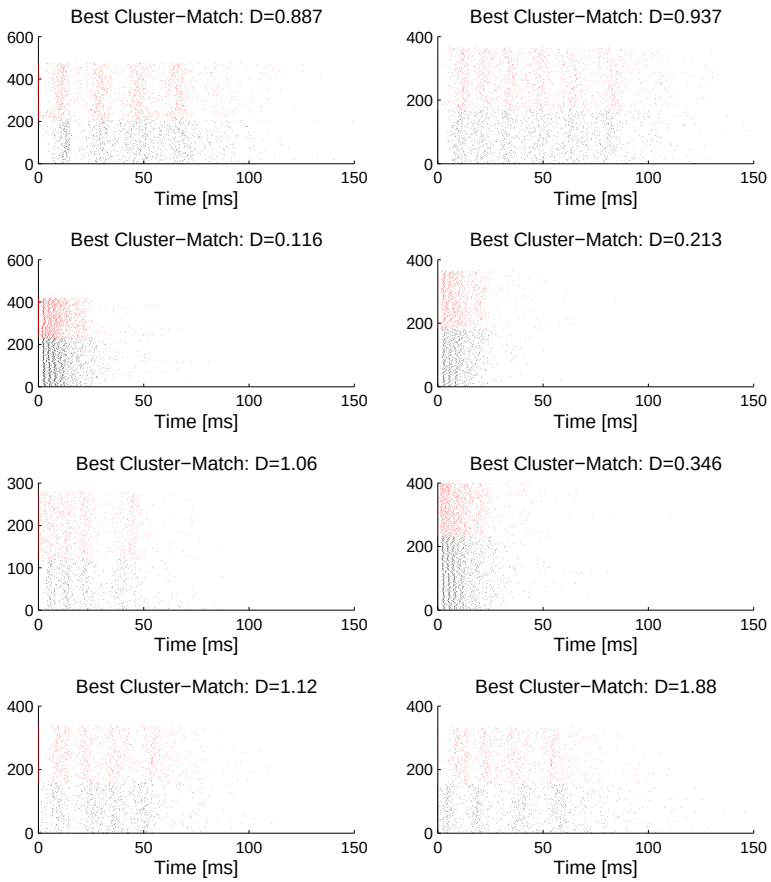


Figure 2.11: Clusters of bursts from two different outdoor recordings and their best matching cluster. D defines the distance between the cluster exemplars under the spike metric.

2.3.6 Matching Clusters of Bursts in two Simultaneous Recordings of the same Cell

In the previous section we compared the similarity of burst activity in the omega-neuron between lab and outdoor recordings, or between different cells in different nights. The "biological microphone approach" offers in addition one unique opportunity to test the power of our method, by comparing the burst responses of omega cells from two different preparations recorded simultaneously, and placed next to each other, so that they experience the same acoustic events. For the experiment presented in Figures 2.13 and 2.14 the two preparations were placed in the nocturnal rainforest, at a distance of about 10 cm from each other, so that they were exposed to the same acoustic environment. Prior to these recordings, the threshold of each omega-cell in response to a pure tone, 20 kHz stimulus was determined in the laboratory, and one preparation was 5 dB less sensitive compared to the other preparation. In this experiment, no artificial sound stimuli were broadcast to the

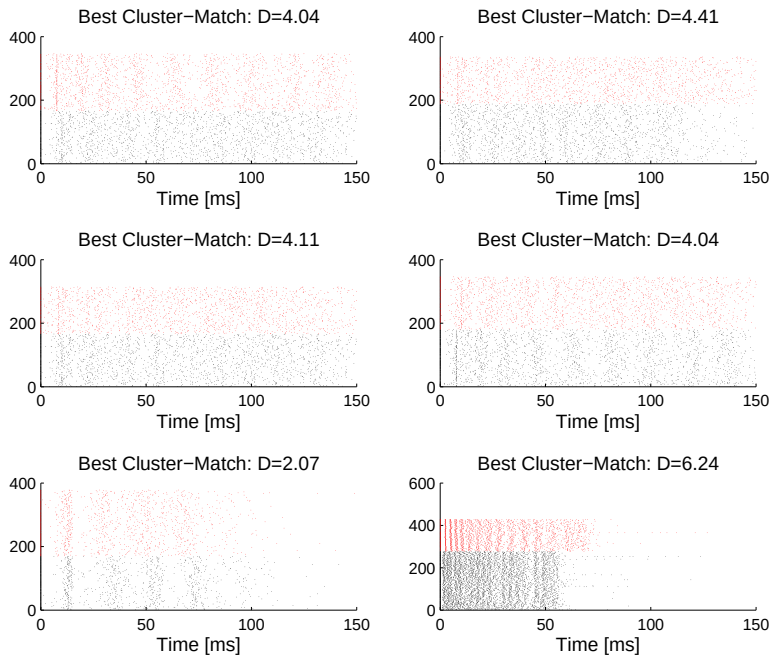


Figure 2.12: Clusters of long bursts from two different outdoor recordings and their best matching cluster. D defines the distance between the cluster exemplars under the spike metric.

preparations, so all bursts had been elicited as a result of background noise alone. Figure 2.13A shows a short sequence of the original spike recording of both cells, and in Figure 2.13B the firing and burst rates of both cells are illustrated for a sequence of continuous 20 minutes of recording. The gross firing and bursting pattern of both cells is rather similar (Figure 2.13A), although the less sensitive cell exhibits a reduced firing rate (Figure 2.13B). The firing rates are actually correlated with a correlation coefficient of $\rho = 0.34$, the bursting rates are correlated with $\rho = 0.27$.

Even though the firing behavior of the two omega-cells is somewhat different due to the threshold difference of 5 dB, one should find similar spiking patterns in the bursts, as the two preparations had been exposed to the same background noise. For both preparations the bursts were extracted, and the affinity propagation algorithm was used to find clusters in the aggregated set of bursts from both preparations. Figure 2.14 shows the resulting cluster dendrogram, where bursts from the first preparation are drawn in red, and bursts from the second preparation in black. Every cluster contains bursts from both preparations, and no cluster contains significantly more than 50% of bursts from only one preparation.

These results suggest that there are no firing patterns that are uniquely found only in one preparation, but not in the other. Instead, the bursting patterns in response to the same acoustic background are very similar for different preparations.

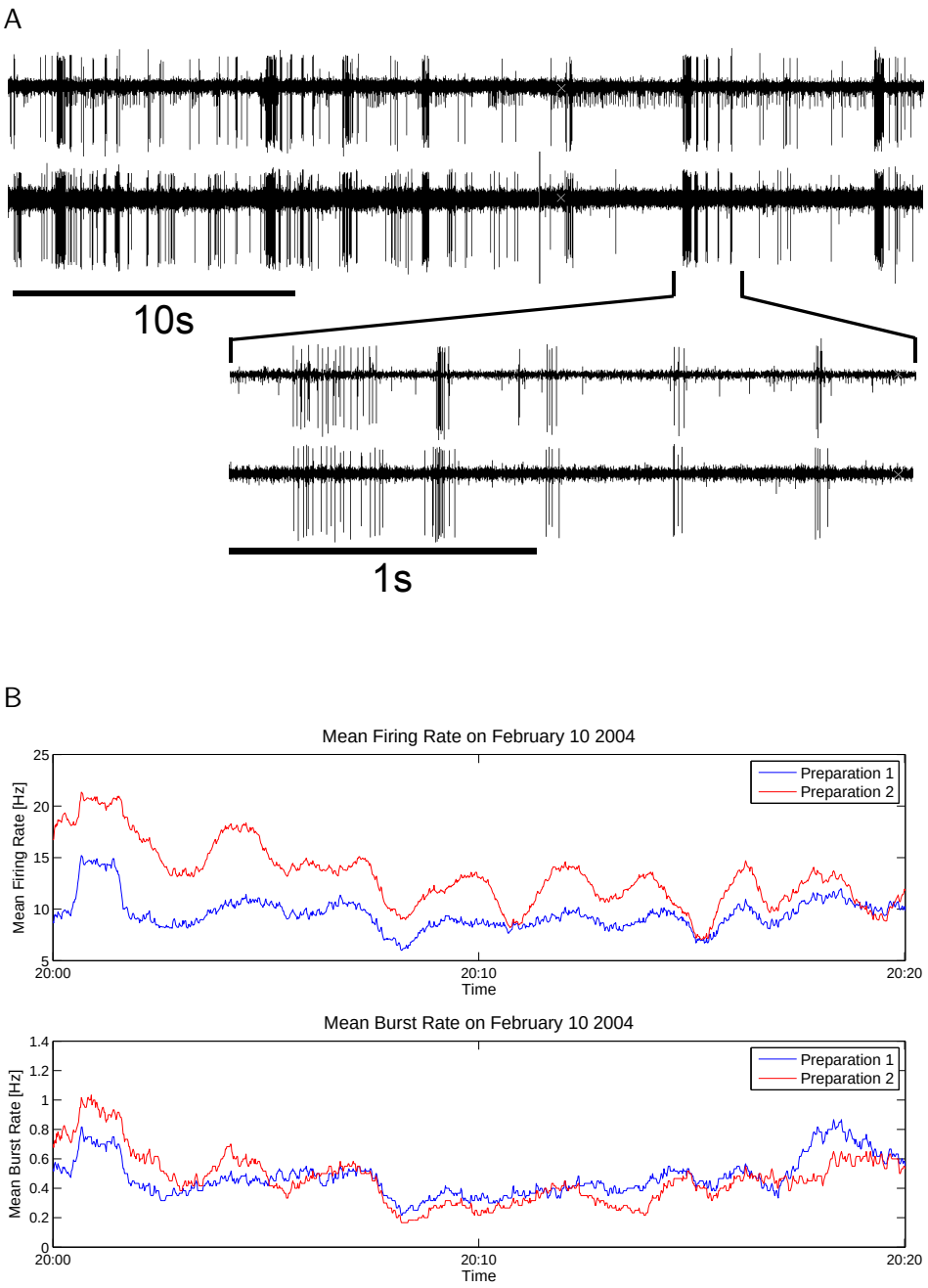


Figure 2.13: **A)** Short sequence of the original spike recording of both cells recorded simultaneously. **B)**: Firing and burst rates of both cells for a duration of 20 minutes. The firing rates of preparation 1 and 2 are correlated with a correlation coefficient of $\rho = 0.34$. The burst rates are correlated with $\rho = 0.27$. Mean firing rates over the entire 20 minute recordings are 9.38 Hz (preparation 1) and 13 Hz (preparation 2), and mean bursting rates are 0.47 Hz and 0.46 Hz respectively.

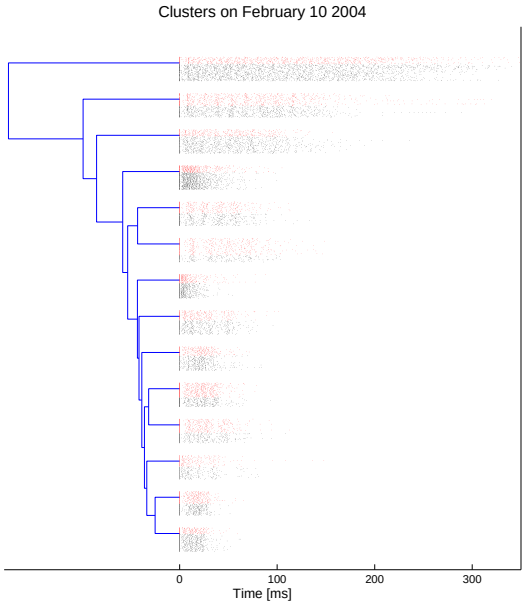


Figure 2.14: Clusters for the aggregated bursts of two omega-cell preparations recorded simultaneously in their natural habitat (February 10 2004); no broadcast of artificial stimuli. Bursts from preparation 1 are drawn in red, and bursts from preparation 2 in black. Every cluster contains about half of its bursts from one preparation.

2.4 Discussion

2.4.1 Coding problems for stimuli in the natural environment

For the two major tasks of sensory systems of object classification and discrimination the central nervous system needs to interpret the ongoing afferent spike activity. Consistent with a number of previous studies on sensory coding in different modalities we view short bursts of action potentials as the basic units for the representation of information. The importance of bursts, in contrast to single spikes, has been discussed in the context of the efficiency of synaptic transmission and thereby synaptic plasticity (Lisman, 1997), in the selective distribution of information to different target neurons (Izhikevich et al., 2003), or the dynamics of encoding behaviorally relevant stimulus features (Gabbiani et al., 1996; Oswald et al., 2004; Alitto et al., 2005; Marsat & Pollack, 2007; Krahe & Gabbiani, 2004). Bursts can be viewed as robust symbols for the neural coding alphabet; they can carry information in their duration, the number of spikes, or the exact timing of the firing pattern, and specifically tuned synapses may read out such a code easily (Kepecs & Lisman, 2003).

However, classification and discrimination are severely impaired by variation in afferent spike trains, either as a result of intrinsic noise in nervous systems,

or external noise resulting from interactions of the stimulus with the transmission channel. A further source of variability of high relevance for a receiver is introduced as a result of small differences in the features of signals from different sources, such as the signals of mates. Ronacher et al. (2004) reviewed the sources of spike train variability and the associated problems and constraints for producing adaptive behavior in grasshoppers. In the case of the auditory system, a further problem results from the background noise of many natural environments, so that relevant stimuli (and stimulus variants) have to be discriminated from irrelevant background noise. Research in the past decade demonstrated that the auditory system of many animals evolved mechanisms to cope with such noise (Brumm & Slabbekoorn, 2005).

In this paper we therefore argue that it is important to study sensory coding under the most realistic conditions, using stimuli in the natural habitat of the organism. It is now widely agreed that the encoding of stimuli by sensory neurons is adapted to the statistics of stimuli in the environment in which an organism lives (Rieke et al., 1995). Variants of the efficient coding hypothesis, for example, have been studied for over 50 years (Barlow, 1961; Attneave, 1954). The hypothesis suggests that stimuli that occur frequently in the natural environment are encoded particularly efficiently by sensory neurons. Under this hypothesis the benefit of particular coding schemes for natural stimuli can be quantified with tools from information theory. Early studies for the visual system (Barlow, 1961) have also suggested that early sensory neurons reduce redundancies in the input, in order to use available computing resources most efficiently. A similar argument was made in the "matched filter hypothesis" in that rather peripheral 'matched filters' may relax the nervous system from computational strain (Capranica & Moffat, 1983; Wehner, 1987). A more recent review of the implications of the efficient coding hypothesis for visual systems can be found in Simoncelli and Olshausen (2001). As is pointed out in this review, efficient coding of natural stimuli should not be studied in isolation, but must also take into account the robustness of neural representations to noise in the environment and stochastic processes at the neuronal level.

The efficient coding hypothesis has recently been challenged by Neuhofer et al. (2008), using a comparative study of homologous neurons in two grasshopper species. They demonstrated that stimuli of high relevance for one species were processed in the afferent auditory system of the other species in exactly the same, quantitatively indistinguishable way, although being "meaningless" in terms of any behavioral relevance (for a similar finding see Ronacher and Stumpner (1988)). This suggests that neuronal elements of the sensory system have been strongly conserved during the evolutionary convergence of the two species. Similarly, in our study we used as a model system a single interneuron, the so-called omega neuron, which has been identified in all species of crickets and bushcrickets so far studied (Molina & Stumpner, 2005). We do not argue, therefore, that the burst coding we found in our study demonstrates specific adaptive properties of the species under study. Rather, we chose to use this insect preparation because of the simple architecture of insect auditory pathways, and their remarkable precision and discrimination abilities in general (Machens et al., 2001, 2005; Rokem et al., 2006). A further reason was that

the interneuron is part of an early processing stage, directly postsynaptic to almost all 20-40 auditory receptor cells in the ear (Römer et al., 1988), so that it integrates signals from almost all receptor cells and a wide range of carrier frequencies from less than 10 kHz far into the ultrasonic range. Thus, monitoring the activity of the cell directly in the animals' own environment provides information about the complete acoustic input of the animal under study, encoded in its spike activity. Therefore, our study is among the first to investigate sensory coding under the natural environmental conditions of an organism, instead of idealized lab conditions.

2.4.2 Detecting spike patterns with unsupervised learning

We have used clustering as an unsupervised tool to detect burst patterns in raw neural recordings. Bursts are characterized by their similarity to all other bursts, measured by a spike metric (Victor, 2005). In a good clustering, bursts that are grouped into the same cluster are similar to each other, but dissimilar to bursts in other clusters. The result of a clustering therefore provides a characterization of different spike patterns that occur frequently in the recorded spike train. Our clustering is based on exact firing patterns, but it is also possible to compute clusterings based on numerical features extracted from bursts (such as firing rate, duration, ...). The disadvantage of this method is that information is lost by replacing the exact pattern with a lower-dimensional feature vector.

Spike-metrics, which we use in our approach for computing similarities of spike trains, have been used in a number of other studies for auditory discrimination. L. Wang et al. (2007) and Narayan et al. (2006) analyzed the time scales at which different conspecific songs could be discriminated by auditory cortical neurons of songbirds under idealized lab conditions. They used different spike metrics such as those proposed by Victor and Purpura (1997) or van Rossum (2001) with different temporal resolutions. Performance was highest for spike timing metrics with short time-scales, which emphasize precise firing times in contrast to firing rates. Their learning framework was based on supervised classification, i.e. template patterns for every sender were known, which is different from the unsupervised approach used in the present paper, where important spike templates are extracted from the recording stream. In a similar approach, Machens et al. (2003) used the metric by van Rossum (2001) for supervised spike train discrimination, and showed that individual calling songs of grasshoppers can be discriminated reliably at the single receptor level, if a metric with high temporal resolution (5 ms) is used.

Finding firing patterns in spike train recordings requires a clustering algorithm which is suitable for this kind of data. Spike trains are not objects in Euclidean space, which is required for basic clustering methods like k-means. We have presented one of the first applications of the affinity propagation algorithm (Frey & Dueck, 2007) to neural recordings. It is fast, reliable, and does not require a lot of parameter tuning for finding suitable firing patterns. Fellous et al. (2004) and Toups and Tiesinga (2006) were the first to use unsupervised methods for finding firing patterns. Their approach was based on the assumption that every spike

train is associated with a particular event (which could be e.g. the presence of a particular sender, or the onset of a stimulus). Spike metrics are used as similarity measure, and clustering was performed on the (Euclidean) vectors of distances from one spike train to all other spike trains with a fuzzy k-means algorithm. They discovered various spike patterns in response to artificial stimuli for recordings from rats, monkeys, and cats. One cell could produce several different spike patterns in response to the same stimulus, depending on the history of the cell. The approach of encoding one spike train by the distance vector to all other spike trains is only practical for relatively small datasets, because the feature vectors become larger with every new training example, and clustering becomes increasingly difficult in higher dimensional feature spaces. The last point is not a problem for affinity propagation, because it does not embed the data in a feature space, and instead uses only distances between data points. Memory limitations are still an issue, but in this study we could apply affinity propagation to very large datasets with more than $10,000$ bursts, and Frey and Dueck (2007) have proposed efficient approximations to handle even larger problems by using sparse distance matrices (these approximations were not used in our study).

2.4.3 Bursts and Neural Codes based on Temporal Firing Patterns

The results obtained with unsupervised clustering demonstrate that information in the omega neuron is not simply encoded by the presence or absence of a burst. The reliability of burst coding in the time domain was very high, so that bursts in response to one of the presented stimuli clustered differently from bursts in response to background noise. On the other hand, it was difficult to distinguish neural responses to short pulse signals from acoustic background noise, since they often clustered together with bursts induced by the background. This would indicate that reliable coding of short signals with little amplitude modulation is severely impaired under high background noise conditions of the nocturnal rainforest. Most remarkably, preparations from different nights show a higher degree of similarity (as quantified by the similarity index D) than bursts of a cell preparation in response to the same stimulus recorded in the sound proof room compared to outdoor conditions. This indicated a high sensitivity of the algorithm for the details of the temporal firing pattern within bursts, since the same homologous cell in different preparations placed at the same position in different nights may experience the same / similar broadcast signals of other animals, and these elicit a rather similar firing pattern. By contrast, the similarity of two bursts recorded in the lab and outdoors can be reduced, since the omega neuron is known for its gain-control mechanism (Pollack, 1988; Römer & Krusch, 2000), which may be activated by high levels of background noise, thus altering the finer details of temporal firing within a burst.

Whereas our results clearly demonstrate the importance of precise spike timing within bursts, it is an ongoing debate whether neural systems use codes that rely more on firing rates, or on exact timing of spikes (Shadlen & Newsome, 1994; Eggermont, 1998). Codes based on bursts are another alternative, but also there

it is not known whether spike timing, firing rates, burst durations, or spike count provide the most efficient encoding of stimuli. It has also been argued that different areas of primate brains may use different encoding strategies (Nicolelis et al., 1998). For single neurons, the general assumption is that firing rates are more robust to intrinsic noise, but require integration of information over longer time scale, and thus cannot encode fast stimulus modulations such as those found in auditory signals. Recent studies on grasshopper receptor neurons revealed a strong dependence of burst occurrence and burst characteristics on temporal modulations of the acoustic input stimuli (Eyherabide et al., 2008, 2009). They found that burst codes based on either the number of spikes or on spike-timing patterns, but not rate codes, could reliably transmit various features of the input. For cortical neurons it was shown that sub-millisecond variations of spike times represent information about fast fluctuations of input stimuli (Mainen & Sejnowski, 1995). Thus, precise firing times may rather encode important features of the stimulus, especially for stimuli with high behavioral importance, instead of being artifacts of noise in spike generation mechanisms.

2.4.4 Number of burst variants, the consequences for reliable acoustic communication, and outlook for necessary refinements of the algorithm

Depending on the individual preparation and the particular recording session, the clustering revealed a variable number of different clusters of bursts over the period of some hours (see e.g. Figure 2.8). If we assume that these different bursts are representations of different signalers, the data give some hints for the requirements of the discrimination task(s) of insect receivers. The number of variants to be discriminated (and thus the difficulty of the task) will affect the speed and accuracy with which it is solved, and can result in speed - accuracy trade-offs in animal decision making (Chittka et al., 2009). First, they need to distinguish conspecific mates and rivals from heterospecific (irrelevant) signalers. This task is probably the easiest, because the amplitude-modulation of most insect calling songs differ from species to species considerably (Gerhardt & Huber, 2002), as should be the case with their representation at early stages of the afferent sensory system. Nevertheless, the transmission channel for sound can impose strong fluctuations on these amplitude modulations (Richards & Wiley, 1980; Römer & Lewald, 1992; McGregor & Krebs, 1984), an effect which increases with distance, so that even this classification task is not without any problems. Furthermore, the probability of signal interference increases with the number of signalers obscuring important features necessary for species recognition. We have seen in our comparison of bursts always recorded at the same position in the rainforest, but at different nights (or years), that some bursts cluster very close together (see quantitative values of D). This would indicate that the specific amplitude-modulated signal of the same individual (or species) elicited rather similar bursting activity in the different receivers, so that these species-specific signals appear to be well represented in the sensory system.

The second task, namely the discrimination between different variants of signals produced by different signalers of conspecifics, is certainly more demanding. In their study on the representation of such variants in grasshopper receptors Machens et al. (2003) have shown that the precise timing of spikes would indeed allow such discrimination under ideal laboratory conditions. However, in the real world situations the precise timing of spikes will be modified by background noise or transmission effects (see the burst comparison in response to the same signal in the lab and outdoors in Figure 2.6), so that such signals (and their variants) need not only be classified as relevant and different from the acoustic background, but there is also the need to discriminate strongly against any burst activity as a result of predator action/vocalisation. Acoustic insects face a variety of such predators, and one of the best studied are the defense and avoidance behaviors in response to insectivorous bats (Fullard, 1998; Hoy, 1992). Behavioral studies on crickets indicate that the discrimination of "good" and "bad" (conspecific from predatory bats) is based on categorical perception (Wyttenbach et al., 1996), and is rather reliable, since it is based on input within a low-frequency and an ultrasonic frequency channel. In bushcrickets, such a discrimination based on frequency is impossible, since conspecifics and bats use similar carrier frequencies. Thus, the important information about predators must be based in the amplitude modulation as well, and should be present in afferent bursting activity recorded at night. For example, in the predator detection system of noctuid moths, Waters (1996) demonstrated that even intrinsic noise in the form of spontaneous action potentials may reduce the ability of moths to discriminate bat from non-bat signals. He proposed that a moth would only be able to recognize an approaching bat from the repetitious nature of the search calls of a bat.

This points to a need for further refinement of our approach, since the algorithm so far developed does not allow for clustering repetitive bursting activity, which should be elicited by the 7 - 20 Hz repetition rate of echolocation calls in the search phase of bats. Similarly, many acoustic insects use characteristic repetition of the same basic call elements, which could also not be detected by the presented algorithm. The classes of bursts identified by our method could, however, be used as a starting point for identifying longer burst sequences in hierarchical approaches. We expect that the identified firing patterns of bursts will serve as robust symbols in the neural coding alphabet, and their temporal alignment provides relevant information about the identity, location, and other important characteristics of an acoustic sender.

2.5 Conclusion

We have demonstrated how unsupervised machine learning techniques in combination with spike metrics may be used to find common spike patterns in neuronal responses to auditory stimuli. The spike trains were recorded from a single identified interneuron in bushcrickets under the most natural possible conditions: by

placing the insect preparation in its natural environment. Under the assumption that the sensory system of the insect is optimized for the conditions of its natural habitat, recordings under these circumstances reveal more about neural coding mechanisms than laboratory experiments without distractor sounds. Our results support this view, since the clustering algorithm produces various stable firing patterns, although there is a substantial level of background noise in the rainforest. We have also found very similar spike patterns in different preparations and under different recording conditions. This may point to a common neural coding mechanism in the omega neuron that is responsible for noise suppression, in order to facilitate the processing for higher areas in the insect brain.

2.6 Acknowledgments

This chapter is based on the paper *Probing real sensory worlds of receivers with unsupervised clustering*, which was written by myself (MP), Manfred Hartbauer (MH), Heinrich Römer (HR), and Wolfgang Maass (WM). The data was recorded by MH and HR, the data analysis and the experiments were designed and performed by MP, and the paper was written jointly by all four authors.

Chapter 3

Predicting Text Relevance from Sequential Reading Behavior

Contents

In this study we show that it is possible to make good predictions of text relevance, from only features of conscious eye movements during reading. We pay special attention to the order in which the lines of text are read, and compute simple features of this sequence. Artificial neural networks are applied to classify the relevance of the read lines. The use of ensemble techniques creates stable predictions and good generalization abilities. Using these methods we won the first competition of the PASCAL Inferring Relevance from Eye Movement Challenge (Salojärvi et al., 2005).

3.1 Introduction

The objective of the PASCAL Inferring Relevance from Eye Movement Challenge (Salojärvi et al., 2005) was to predict relevance of lines of text from eye movements of readers. The subjects were first shown a question and then a list of ten possible answers on a computer screen. The subject had to find the correct answer, then press 'Enter' and finally type in the chosen line number. Among the non-correct lines, there were four sentences which were relevant for the question, five were irrelevant. The task in the challenge is to identify not only the correct, but also the relevant lines of text, being provided only measurements of the eye movements of the subject. The gaze location on the screen and the pupil diameter were tracked during each assignment. For the first competition, the organizers had already segmented the trajectory data from the eye tracker and assigned fixations and saccades to the corresponding words and lines. 22 features that are commonly used in psychological eye movement studies were computed for every word. This made competition one a pure classification problem. In the second competition only the raw eye movement

data and word coordinates were available. Thus preprocessing by segmentation and feature extraction was a main part of the problem. Our work focuses on competition one.

Eye movements during reading have been extensively studied in the psychological literature (Rayner, 1998). Different models pay more attention to either higher-level processes governing the reading behavior, or unconscious eye movements. In our approach, however we more or less neglected unconscious information, and computed features only from the sequence, in which the lines were read. Statistical analysis showed that for this task our simple features contained enough information for standard classifiers to obtain good results.

3.2 Feature Extraction

3.2.1 Features for Competition One

The organizers already provided a dataset with pre-computed features which have shown to be useful in other eye movement studies. Our approach, however, was quite different, since we wanted to find out as much as possible about the relevance of the sentences, only from the order in which the lines were read. This would imply that a much smaller fraction of data, namely only the line number, would have to be measured in order to distinguish between correct, relevant and irrelevant lines. Our features were computed for every seen line in an assignment, and classification was then done on this reduced feature set.

We found out from analyzing individual assignments, that there are some reoccurring patterns of sequential reading behavior. Due to the nature of the task, it was obvious that most readers usually look at the line containing the correct answer before pressing 'Enter' and thereby finishing his task. A heuristic rule which assigns the label *correct* to the last read line in each assignment, identifies 93.75% of all correct answers in the training set. This rule is facilitated by the fact that all assignments with incorrect answers have been filtered out by the organizers.

Additionally we noticed that the subjects spent significantly more time reading correct or relevant sentences than irrelevant ones. So we simply counted the number of measurements for each sentence, which turned out to be a very useful feature for classification. Secondly, the subjects jumped between lines of text while they are thinking about the correct answer. When they are deciding between different possibilities for a correct answer, it seems natural that their focus jumps between the lines within their consideration. Another assumption is that relevant sentences would rather be considered as a possible answer than an irrelevant sentence. We therefore counted, how often the subjects returned to a sentence they had already read before. This gave us a good estimation of the relevance that was attributed to this sentence by the subject, since the number of returns is significantly higher for correct or relevant lines. We also calculated jumps between the presumably correct line and all other sentences as another feature, but this did not improve the performance.

Most subjects showed a stereotypical reading behavior, i.e. they started with line number 1 and continued to read subsequent sentences, eventually jumping back to known answers. Therefore the position of a sentence within the list also plays a role, since some lines are almost always seen (like line 1), and some are more often skipped.

Table 3.1: Definition of features used for classification for every sentence in an assignment. Features marked with * are constant for all sentences in an assignment.

#	Feature	Description
1	`lineNr`	Line number within the 10 possible answers
2	`isLastLine`	1 if this sentence was the last one read during this assignment
3	`count`	Total number of measurements for this sentence
4	`returns`	Number of jumps to this sentence after first reading
5	`nrReadLines`*	Number of lines read during this assignment
6	`readAllLines`*	1 if all possible lines were read
7	`newLinesBefore`	Number of different sentences that the subject started to read *before* reading this line
8	`newLinesAfter`	Number of different sentences that the subject started to read *after* reading this line

A problem we had to deal with was that not all of the possible answers in an assignment were read. Actually only in 40.8% of all assignments in the training set there was at least one measurement for every sentence. The subject was only instructed to look for the correct answer, but he was not told that the remaining lines contain either relevant or irrelevant sentences. Therefore it happened that the subject found the correct answer very early and did not even read a single relevant sentence. In this case it is of course almost impossible to make a distinction between relevant and irrelevant sentences. So we included the number of read lines as valuable context information, and also included a binary flag if all possible answers were read.

Since subjects tended to stop reading after they found the correct answer, we calculated the number of different sentences that the subjects read after or before having read that line. It turned out that the subjects read significantly fewer lines after reading the correct answer. In combination with the line number this feature thus provides evidence whether a sentence is the correct answer or not. Table 3.1 summarizes the 8 features that were used:

3.2.2 Statistical Analysis

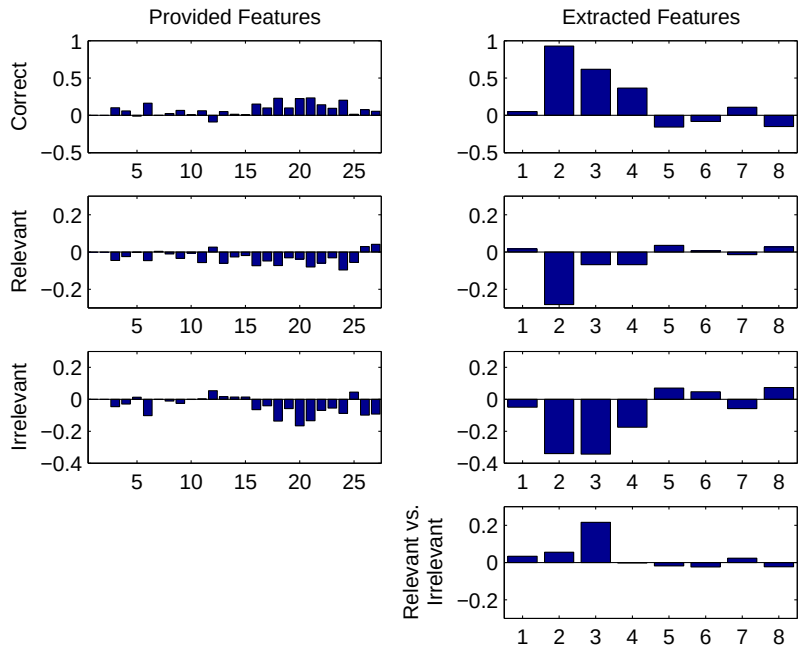
In this section we analyze the statistical properties of our extracted features on the training set. We use mutual information (MI), a measure for arbitrary dependency between random variables, and linear correlation (LC) to compare the calculated features against the provided features for competition one. Mutual information has been used extensively in the literature for feature selection (Blum & Langley, 1997) because mutual information is invariant under linear transformations and takes into account the entire dependency structure of the random variables. On the other hand linear correlation is a natural measure for variable dependency. These measures are therefore good estimators for the relevance of features, although they do not take into account correlations among different features.

Firstly, we calculated the linear correlation and mutual information for each feature provided in the challenge with the class labels. We found that the original features are poorly correlated, as can be seen in the left columns of Figures 3.1 (a) and (b) respectively. In the first row of Figure 3.1(a) the LC and in 3.1(b) the MI, is shown for the correct vs. non-correct labels. The second rows show relevant vs. non-relevant and the third rows irrelevant vs. non-irrelevant. The last rows display LC and MI for the relevant vs. irrelevant problem, where correct lines have been removed. The provided features in general exhibit small MI values, except for features 12 (`lastSaccLen`), 21 (`regressDur`), and 25 (`nWordsInTitle`). The higher information of feature 12 about the correct labels confirmed the earlier statement about jumps between the correct answer and other titles. The high MI value for feature 21 is based on the same concept of going back for regression and re-reading. The duration of a regression is therefore more significant for classification than all other features.

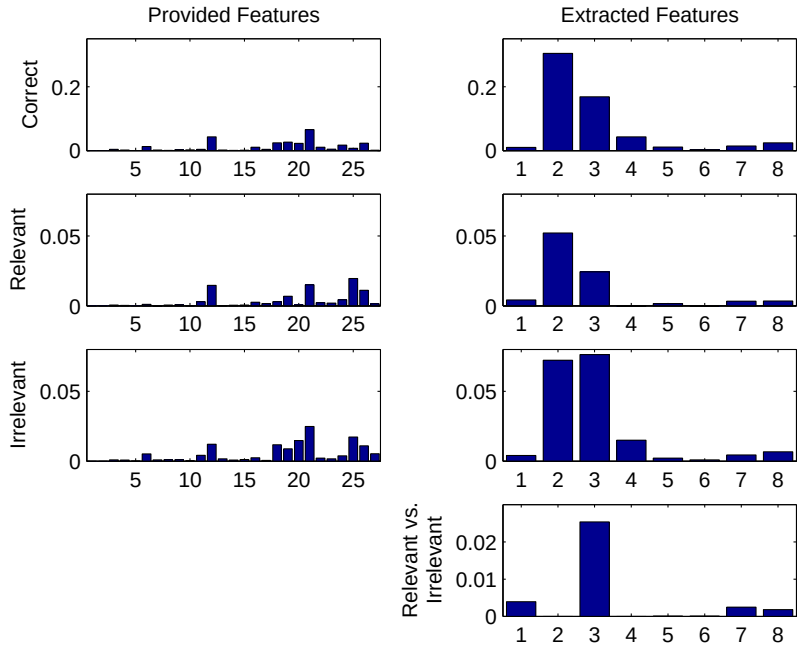
Secondly, the mutual information and linear correlation for each of our extracted features were computed (see Figure 3.1 - right columns). The results confirm that most of these features are significantly more correlated with the class labels than the original features. Note that feature 21 has the highest MI value with 0.072, which is four times smaller than the maximum value for the new features (feature 2, with a MI value of 0.3042).

Some of the features exhibit neither high correlation nor high mutual information with the class labels. Nevertheless their inclusion boosted the performance of the classifiers. To explain this effect we calculated the MI of pairs of features with the class labels. We discovered that all features except number 2 (`isLastLine`, which is a good predictor on its own) show higher MI values in combination with each other than the sum of their individual MI values. This means that even though some features are weak individual predictors, together they form a strong feature set.

We also calculated the MI and LC for our feature set after removing the correct titles to identify the features which are most significant for the relevant vs. irrelevant problem (see Figure 3.1 - last row). Feature 2 was almost constant for this dataset and therefore was not used for identifying relevant or irrelevant lines. Including



(a) Linear Correlation



(b) Mutual Information

Figure 3.1: Correlation and mutual information analysis of different feature sets.

all remaining features, even if their correlation and mutual information values were low, resulted in the best performance. Combined MI analysis again explained this effect.

3.3 Classification and Results

In this section we present our strategies and methods for building proper classifiers for the challenge datasets. Since the task is a 3-class classification problem, there exist two main strategies to solve the problem: the first one is to build a multi-class classifier, which gets the input pattern and assigns it to one of 3 classes. The second method, that we used in our systems, is to use a hierarchical classifier that first checks if the input pattern is a member of one of preselected classes (correct lines in this challenge) or not. If the result is negative, it passes the pattern to another classifier that assigns it to one of the other two remaining classes (relevant vs. irrelevant). In other words this technique decomposes a multi-class problem to a series of two-class pattern recognition problems.

Table 3.2: Correct detection rate (in %) of different 3-class classifiers.

	C4.5	**AdaBoost**	**SVM**	**MLP**
Train	68.17	68.62	63.85	66.71
Validation	69.19	65.24	66.23	71.09

We used the WEKA 3.4 package (Witten & Frank, 2005), which allowed us to quickly compare a variety of pattern recognition algorithm for this task. After manual tuning of the hyperparameters we calculated detection rates for different learning algorithms for training and validation sets averaged over 10 runs. The results are summarized in Table 3.2, showing an advantage for Multi-Layer Perceptrons (MLP) on the validation set. Note that these values are for 3-class classifiers, but we tried the same experiments for the two stage classification strategy mentioned above and the results were very similar.

3.3.1 Correct Line Identification

As can be seen in Figure 3.1, linear correlation and mutual information analysis shows that the features `isLastLine`, `count`, and `returns` are the most informative ones for identification of correct lines. Because of this we used only these features as input to our classifiers. We also changed the target labels to +1 for correct lines and -1 for the rest (relevant and irrelevant).

For the rest of the project we switched to a more efficient and flexible tool for training neural networks, namely the MATLAB Neural Networks Toolbox. A MLP with 3 hidden neurons and hyperbolic tangent activation functions was trained with scaled conjugate gradient backpropagation. We tried different numbers of hidden

neurons, but since the performance did not change significantly with more neurons, we chose the simplest network to avoid overfitting. In addition the error on the validation set was used as stopping criterion for training. The overall performance on training, validation, and test sets are shown in Table 3.3. Ensemble methods, as discussed in the next section, were also tried out for this task, but since the recognition rate was almost constant for different MLPs, we decided to use a single classifier instead.

3.3.2 Relevant vs. Irrelevant Lines Identification

For this task we first removed the predicted correct lines from the previous classifier for training, validation, and test sets. Then the feature `isLastLine` was removed, since correlation and mutual information analysis showed that it had no major contribution to this task. We also changed the target labels to +1 for relevant and -1 for irrelevant lines. As another preprocessing step, we normalized the feature values to have zero mean and unit variance according to the combined training, validation and test sets.

Different MLPs were trained, and for each network the numbers of hidden neurons was randomly selected between 4 and 10. The activation function was hyperbolic tangent for all neurons and we trained our networks using the Levenberg-Marquardt backpropagation algorithm of MATLAB's Neural Networks Toolbox. As before we used the error on the validation set as stopping criterion.

Table 3.3: Correct detection rate (in %) for two stage classifiers. Row 2 shows the average performance of single MLPs, rows 3 and 4 correspond to the two different ensemble methods for relevant vs. irrelevant line classification. Overall accuracy for the 3-class predictions is shown in parentheses.

	Train	**Validation**	**Test**
Correct vs. Rest	98.54	98.52	98.72
Single MLP	63.01 (67.26)	66.57 (69.81)	67.91 (71.03)
Best-Of Ensemble	64.21 (68.02)	68.01 (71.25)	69.26 (72.31)
Outlier-Filtered Ensemble	64.25 (68.06)	67.73 (71.17)	69.09 (72.16)

The main difference compared to the previous setup for correct line identification was that we used an ensemble averaging method to improve and stabilize our recognition rate and avoid possible overfitting. It has been shown that in most cases ensemble averaging methods improve the generalization properties of classifiers (Opitz & Maclin, 1999; Bauer & Kohavi, 1999). So we averaged the confidence values (outputs of the networks) over all ensemble members and then used it as a decision criterion. We used two different methods to select ensemble members: one method, named *Best-Of* ensemble, selected the best 10 networks out of 15 different trained networks. The other approach, named *Outlier-Filtered* ensemble, filtered

out networks that showed relatively high error rates. 5 networks were selected, and the selection threshold was set at an error rate of 38%. For both methods we used the error on the validation set as our selection criterion.

The overall results are given in rows 2-4 of Table 3.3 for training, validation and test sets. We first show the average performance of single MLPs, and then the accuracy for both ensemble selection methods in the last two rows. The benefits of using ensemble methods can be seen by comparing row 2 with rows 3 and 4, since the error on all sets was on average reduced by more than 1% (for both methods). In addition the variance of the classification error was also significantly reduced. In competition one, *Best-Of* ensemble finished first, *Outlier-Filtered* ensemble finished second. The next best result was 0.9% lower than our best performance.

Furthermore, we tried a post-processing step in which the main goal was to correct inconsistent decisions such as having more than 4 relevant or more than 5 irrelevant detections. The confidence values of the ensemble were used as the basis for the post-processing decision. We tried to change the labels of less probable excessive detections to the other class. The major problem was that in most cases the confidence value was not a good representative of being a member of a class when there were excessive detections. So we decided not to use this post-processing method for our classifiers.

3.4 Conclusion

Our feature extraction and classification approaches were highly successful in this challenge. The ensemble methods proved to be very stable and exhibited very good generalization performance. In addition we showed that a lot of information about the relevance of read lines can be extracted from features about sequential reading behavior. We do not claim, however that unconscious eye movements during reading are not informative for this task, but our results show that reasonable accuracy can be obtained without them.

3.5 Acknowledgments

This chapter is based on the paper *Predicting Text Relevance from Sequential Reading Behavior*, which was written in collaboration with Amir Saffari and Andreas Juffinger, who contributed ideas for the design of features and helped in preparing the manuscript.

Chapter 4

Efficient Continuous-Time Reinforcement Learning with Adaptive State Graphs

Contents

We present a new reinforcement learning approach for deterministic continuous control problems in environments with unknown, arbitrary reward functions. The difficulty of finding solution trajectories for such problems can be reduced by incorporating limited prior knowledge of the approximative local system dynamics. The presented algorithm builds an adaptive state graph of sample points within the continuous state space. The nodes of the graph are generated by an efficient principled exploration scheme that directs the agent towards promising regions, while maintaining good online performance. Global solution trajectories are formed as combinations of local controllers that connect nodes of the graph, thereby naturally allowing continuous actions and continuous time steps. We demonstrate our approach on various movement planning tasks in continuous domains.

4.1 Introduction

Finding near-optimal solutions for continuous control problems is of great importance for many research fields. In the weighted region path-planning problem, for example, one needs to find the shortest path to a goal state through regions of varying movement costs. In robotics specific reward functions can be used to enforce obstacle avoidance or stable and energy-efficient movements. Most existing approaches to these problems require either complete knowledge of the underlying

system, or are restricted to simple reward functions. In this study we address the problem of learning high quality continuous-time policies for tasks with arbitrary reward functions and environments that are initially unknown, except for minimal prior knowledge of the local system dynamics.

Reinforcement learning (RL) (Sutton & Barto, 1998) is an attractive framework for the addressed problems, because it can learn optimal policies through interaction with an unknown environment. For continuous tasks, typical approaches that use parametric value-function approximation suffer from various problems concerning the learning speed, quality, and robustness of the solutions (Boyan & Moore, 1995). Several authors have therefore advocated non-parametric techniques (Ormoneit & Sen, 2002; Jong & Stone, 2006), where the value function for the continuous problem is only computed on a finite set of sample states. In this case stronger theoretical convergence and performance guarantees apply (Ormoneit & Sen, 2002). Still, few RL algorithms can cope with continuous actions and time steps.

Sampling-based planning methods (Kavraki et al., 1996; Guestrin & Ormoneit, 2001), on the other hand, can efficiently construct continuous policies as combinations of simple *local controllers*, which navigate between sampled points. Local controllers for small regions of the state space are often easily available, and can be seen as minimal prior information about the task's underlying system dynamics. Local controllers do not assume complete knowledge of the environment (e.g. location of obstacles), and are therefore not sufficient to find globally optimal solutions. Instead, a graph is built, consisting of random sample points that are connected by local controllers. A global solution path to the goal is constructed by combining the paths of several local controllers.

Planning techniques are very efficient, but their application is limited to completely known environments. Guestrin and Ormoneit (2001), e.g., have used combinations of local controllers for path planning tasks in stochastic environments. Their graph is built from uniform samples over the whole state space, rejecting those that result in collisions. They also assume that a detailed simulation of the environment is available to obtain the costs and success probabilities of every transition. In this study we address problems in which the exact reward function is unknown, and the agent has no knowledge of the position of obstacles.

We propose an algorithm for efficiently exploring such unknown continuous environments in order to construct sample-based models. The algorithm builds an *adaptive state graph* of sample points that are connected by given local controllers. Feedback from the environment, like reward signals or unexpected transitions, is incorporated online. Efficiently creating adaptive state graphs can be seen as an optimal exploration problem (Simsek & Barto, 2006). The objective is to quickly find good paths from the start to the goal region, not necessarily optimizing the online performance. Initial goal-directed exploration creates a sparse set of nodes, which yields solution trajectories that are later improved by refining the sampling in critical regions. Planning with adaptive models combines the advantages of reinforcement learning and planning. We regard our algorithm more as a RL method, in the spirit of model-based RL (Sutton & Barto, 1998; Moore & Atkeson, 1993),

since the agent learns both its policy and its world model from actual experience.

The adaptive state graph transforms the continuous control problem into a discrete MDP, which can be exactly solved e.g. by dynamic programming (Sutton & Barto, 1998). This results in more accurate policies and reduced running time in comparison to parametric function approximation. The obtained policy still uses continuous actions and continuous time steps, leading to smoother and more natural trajectories than in discretized state spaces. Here we address primarily deterministic and episodic tasks with known goal regions, but with small modifications these restrictions can be relaxed. Prior knowledge of the goal position, for example, speeds up the learning process, otherwise the agent will uniformly explore the state space. We demonstrate in comparisons of our algorithm to standard RL and planning techniques that fast convergence and accurate solution trajectories can be achieved at the same time.

In the next section we introduce the basic setup of the problem. We show the structure of the algorithm in Section 4.3 and present the details of the adaptive state graph construction in Section 4.4. In Section 6.5 we evaluate our algorithm on a continuous path finding task and a planar 3-link arm reaching task, before concluding in Section 6.7.3.

4.2 Graph Based Reinforcement Learning

We consider episodic, deterministic control tasks in continuous space and time. The agent's goal is to move from an arbitrary starting state to a fixed goal region, maximizing a reward function, which evaluates the goodness of every action. In the beginning, the agent only knows the locations of the start state and the goal region, and can use local controllers to navigate to a desired target state in its neighborhood.

Let $\mathcal{X}$ define the state space of all possible inputs $x \in \mathcal{X}$ to a controller. We require $\mathcal{X}$ to be a metric space with given metric $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_0^+$. Control outputs $u \in \mathcal{U}$ change the current state x according to the system dynamics $\dot{x} = f(x, u)$. In this study we assume that only an approximative local model $\hat{f}(x, u)$ is known, which does not capture possible nonlinearities due to obstacles. The objective is to find a control policy $\mu : \mathcal{X} \to \mathcal{U}$ for the actual system dynamics $f(x, u)$ that returns for every state x a control output $u = \mu(x)$ such that the agent moves from a starting state $x^S \in \mathcal{X}$ to a goal region $X^G \subset \mathcal{X}$ with maximum reward.

Our algorithm builds an adaptive state graph $\mathcal{G} = \langle \mathcal{V}, E \rangle$, where the nodes in $\mathcal{V} = \{x_1, \ldots, x_N\} \subset \mathcal{X}$ form a finite subset of sample points from $\mathcal{X}$. We start with $\mathcal{V}_0 = \{x^S\}, E_0 = \emptyset$ and let the graph grow in subsequent exploration phases. The edges in $E \subseteq \mathcal{V} \times \mathcal{V}$ correspond to connections between points in $\mathcal{V}$ that can be achieved by a given *local controller*. The local controller $a(e)$ for an edge $e = (x_i, x_j)$ tries to steer the system from x_i to x_j, assuming that the system dynamics along the path corresponds to $\hat{f}(x, u)$. If an edge can be traversed with a local controller, it is inserted into E, and $r(e)$, the total reward obtained on the edge is stored. The

combination of multiple edges yields globally valid trajectories.

For a given graph $\mathcal{G}$ the actual task is to find an optimal *task policy* π from the starting state x^S to the goal region X^G. We therefore have to find the optimal sequence of edges $\langle e_i \rangle$ in the graph from x^S to X^G such that the sum of rewards $R^\pi := \sum_{i=0}^{n} r(e_i)$ is maximized. The problem is solved by calculating the optimal value function V^π through dynamic programming. This method is guaranteed to converge to an optimal policy (Sutton & Barto, 1998), based on the knowledge contained in the adaptive state graph.

The quality of the resulting policy depends on the available edges and nodes of the graph, but also on the quality of the local controllers. We assume here that local controllers can compute near-optimal solutions to connect two states in the absence of unforeseen events. Feedback controllers can compensate small stochastic effects, but the presented algorithm in general assumes deterministic dynamics. We restrict ourselves here to rather simple system dynamics, for which controllers are easily available, e.g. straight-line connections in Euclidean spaces.

While the agent is constructing the graph it is following an *exploration policy* π^{exp}, which can be different from the task policy π. π^{exp} does not always take the best known path to the goal, but also traverses to nodes where the creation of additional nodes and edges may lead to better solutions for the actual task. Virtual *exploration edges* to unvisited regions with heuristic *exploration rewards* are therefore inserted into the graph. This creates incentives for the algorithm to explore new regions. Whenever such virtual edges are chosen by the exploration policy, the graph is expanded to include new nodes and edges.

4.3 Structure of the Algorithm

The adaptive state graph $\mathcal{G}$ is grown from the start state towards the goal region. We use the approximative model $\hat{f}(x, u)$ to generate new potential successor states from existing nodes, and rank them by a heuristic exploration score. An exploration queue $\mathcal{Q}$ stores the most promising candidates for exploration, and the exploration policy, defined via the value function V^{exp} directs the agent towards one of these targets. Since our goal is finding a good policy from the start, not necessarily maximizing the online performance, the selection of the exploration target involves an exploration-exploitation trade-off inherent to all RL methods. Our method uses the information in the graph to efficiently concentrate on relevant regions of the state space. Whenever a new state is visited, it is added as a node into the graph. We also add all possible edges to and from neighboring nodes that can be achieved by local controllers. Initial optimistic estimates for the reward come from the local controller, but are updated when actual experience becomes available.

Algorithm 1 shows a pseudo-code implementation of the basic algorithm. Details of the subroutines are explained in Section 4.4. Roughly the algorithm can be structured into 3 parts: the first part in lines 5-11 deals with the generation of new exploration nodes and is described in Sections 4.4.1-4.4.3. The second part in

lines 12-15 first updates the value functions, and then executes the local controller to move to a different node (see Sections 4.4.4-4.4.5). In the remaining part (lines 16-26) we incorporate the feedback received from the environment to update the graph (see Section 4.4.6).

Algorithm 1 Graph-based RL

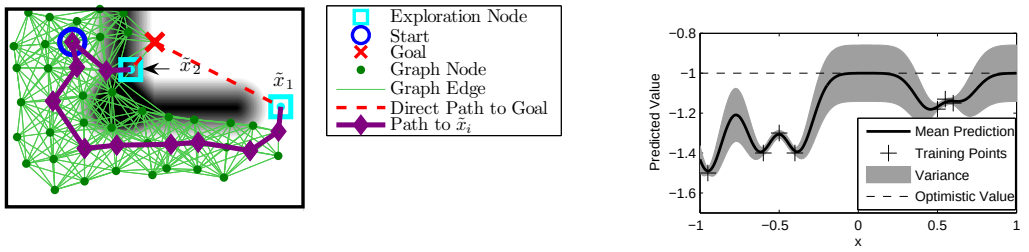
Input: Start x^S, goal region X^G, local controller a

1: **Initialize** $\mathcal{V} = \{x^S\}$, $E = \emptyset$, $\mathcal{G} = \langle \mathcal{V}, E \rangle$, $\mathcal{Q} = \emptyset$
2: **repeat** (for each episode):
3: **Initialize** $x = x^S$
4: **repeat** (for each step of the episode):
5: **for** i = 1 **to** N_t **do**
6: $\tilde{x}_i =$ `sample_new_node`()
7: $[\sigma(\tilde{x}_i), \mathrm{var}_\sigma(\tilde{x}_i)] =$ `exploration_score`$(\tilde{x}_i, V)$
8: **if** $\mathrm{var}_\sigma(\tilde{x}_i) > \theta^{\mathrm{exp}}_{\mathrm{min}}$ **then**
9: `insert_exploration_node`$(\tilde{x}_i)$
10: $[V, V^{\mathrm{exp}}, \mathcal{Q}] =$`replan`$(\mathcal{G})$
11: Select next edge $e = (x, x')$ stochastically (e.g. ε-greedy) from V^{exp}
12: Execute local controller $a(x, x')$
13: Receive actual state $\hat{x}'$ and reward r of transition
14: **if** $d(x', \hat{x}') > \delta$ **then** {*different state than predicted was reached* }
15: Delete edge (x, x') from $\mathcal{G}$ and insert edge $(x, \hat{x}')$
16: Set $x' = \hat{x}'$
17: **if** x' was previously unvisited **then**
18: `insert_new_node`(x')
19: `update_edge`(x, x', r)
20: $[V, V^{\mathrm{exp}}, \mathcal{Q}] =$`replan`$(\mathcal{G})$
21: **else**
22: `update_edge`(x, x', r)
23: **until** x is terminal

Output: Task policy π, derived from $\mathcal{G}$ and V

4.4 Building the Adaptive State Graph

A key for efficient exploration of the state space is the generation of sample states. Previous approaches for sampling-based planning, e.g. (Guestrin & Ormoneit, 2001; Kavraki et al., 1996), have used uniform random sampling of nodes over the whole state space. This requires a large number of nodes, of which many will lie in irrelevant or even unreachable regions of the state space. On the other hand, a high density of nodes in critical regions is needed for fine-tuning of trajectories. The presented algorithm iteratively builds a graph by adding states that are visited during online exploration. It thereby fulfills two objectives: Firstly, the exploration



(a) (b)

Figure 4.1: (a) Illustration of the exploration process. Exploration node $\tilde{x}_1$ is preferred over $\tilde{x}_2$, because the reward to reach $\tilde{x}_2$ is strongly negative. (b) Illustration of value prediction with Gaussian processes on an artificial 1-D dataset. The prediction approaches the optimistic value and has larger variance for points that are farther away from training points.

is directed to search towards a goal state, and secondly, it optimizes the current policy in regions where the number of nodes is insufficient.

4.4.1 Generating Samples: `sample_new_node`

Whenever a node x in the graph is visited the algorithm stochastically creates a number of potential *exploration nodes* for that state. New exploration nodes are created uniformly in the neighborhood of the current node. We therefore first uniformly sample an execution time $t_i \in [t_{\min}, t_{\max}]$, and a constant control action u_i in U. Then we simulate the local dynamics $\hat{f}(x, u)$ from x with action u_i for time t_i, and reach a new node $\tilde{x}_i$. For efficiency reasons the number of generated samples N_t should be reduced over time. Similarly the minimum and maximum execution time is reduced over time to create finer sampling and achieve fine-tuning of the policy.

4.4.2 Evaluating Exploration Nodes: `exploration_score`

Efficient exploration preferentially visits regions where an improvement of the task policy is possible, but avoids creating unnecessary nodes in already densely sampled regions. We estimate the utility of every potential exploration target $\tilde{x}$ by an *exploration score* $\sigma(\tilde{x})$, and direct the agent towards the most promising such nodes. Informed search methods like A* (Hart et al., 1968) estimate the utility of $\tilde{x}$ as the expected return of a path from the start x^S to the goal region X^G via $\tilde{x}$. This can be decomposed into the path costs $c(x^S, \tilde{x})$ from x^S to $\tilde{x}$ plus the estimated value $\hat{V}(\tilde{x})$, i.e. the estimated rewards-to-goal. Therefore $\sigma(\tilde{x}) = c(x^S, \tilde{x}) + \hat{V}(\tilde{x})$.
For calculating the path costs $c(x^S, \tilde{x})$ we use only visited edges of the state graph. Otherwise the optimistic initialization of edge rewards will almost always lead to an underestimation of the path costs, and therefore all exploration nodes will ap-

pear similarly attractive. $\hat{V}(\tilde{x})$ must be an optimistic estimate of the value, e.g. the estimated costs of the direct path to the goal in the absence of obstacles. If the goal is not known, a constant value must be used. This prior estimate for the value of a new node $\tilde{x}$ can be improved by considering also the task-policy values $V(x')$ of nearby existing nodes x'. Gaussian process regression (Rasmussen & Williams, 2006) is a suitable method to update predictions of a prior function by taking information from a finite set of training examples into account, thereby creating a more-exact posterior. The contributions of individual training examples are weighted by a kernel $k(x, x')$, which measures the similarity between a training point x' and test point x. Typical kernels monotonically decrease with growing distance to a training point. Therefore the prediction for a new node that is far away from existing points approaches the optimistic prior estimation, whereas a point close to existing nodes will receive a prediction similar to the weighted mean of values from neighboring nodes. The range in which training points contribute to predictions can be controlled by a *bandwith* parameter β of the kernel, which is task dependent and needs to be chosen in advance. In our experiments we use a standard squared exponential kernel $k(x, x') = \exp\left(-\frac{d(x,x')^2}{2\beta^2}\right)$.

Since the prior estimate is an optimistic estimation of the true value, the predictions for an exploration node will usually increase the further the new node is away from existing nodes (see Figure 4.1(b)). Therefore this approach enforces exploration into unvisited areas. Additionally to the value estimate $\hat{V}(\tilde{x})$ the Gaussian process returns the variance $\mathrm{var}_\sigma(\tilde{x})$ of the prediction. Since the variance increases with distance to training points, we can use $\mathrm{var}_\sigma(\tilde{x})$ as a measure for the sampling density around $\tilde{x}$. To control the number of nodes we reject exploration nodes with variance lower than $\theta^{\mathrm{exp}}_{\mathrm{min}}$. This threshold may be lowered over time, to ensure refinement of the adaptive state graph in later episodes.

4.4.3 Integrating New Exploration Nodes: `insert_exploration_node`

Newly generated exploration nodes $\tilde{x}_i$ are placed on the *exploration queue* $\mathcal{Q}$, which is a priority queue ranked by the exploration scores $\sigma(\tilde{x}_i)$. The highest scored exploration targets in $\mathcal{Q}$ are the most promising candidates for exploration. If σ_{max} is the best score of a node on the queue, we consider all exploration nodes with a score not worse than $\sigma_{\mathrm{max}} - \theta_\sigma$, with $\theta_\sigma \geq 0$ as targets for the exploration policy π^{exp}. Virtual and terminal *exploration edges* are added to the graph for each such node $\tilde{x}$, originating from the node from which $\tilde{x}$ was created. The rewards of these edges are the estimated rewards-to-goal, given by $\hat{V}$. The exploration policy may then either choose an exploration edge, thereby adding a new node to the adaptive state graph, or move to an already visited node. The latter indicates that exploring from other nodes seems more promising than continuing the exploration at the current node.

The threshold parameter θ_σ has an interesting interpretation in the context of the exploration-exploitation dilemma. If $\theta_\sigma = 0$ then the agent will always choose

the most promising exploration target, similar to A* search (Hart et al., 1968) on a partially unknown graph. This will however yield a bad online performance, because the agent may have to travel all the way through the state space if it discovers that another node promises better solutions. $\theta_\sigma = \infty$ will lead to greedy search, and ultimately to inefficient uniform sampling of the whole state space. By adjusting $\theta_\sigma > 0$ one can balance the trade-off between online performance and finding near-optimal start-to-goal paths as soon as possible.

4.4.4 Re-planning within the Graph: `replan`

The adaptive state graph yields a complete model of the reduced MDP, which can be solved by dynamic programming methods. In practice we use efficient re-planning techniques like Prioritized Sweeping (Moore & Atkeson, 1993) to minimize the number of updates in every iteration. In most steps this requires only a very small number of iterations on a small set of nodes. Only when important connections are found, and the value of many states changes, we need to compute more iterations.

Re-planning is run twice: once on the graph that includes only exploration targets in $\mathcal{Q}$ with score larger than $\sigma_{\max} - \theta_\sigma$ as terminal states. This yields the value function $V^{\exp}$ for the *exploration policy* $\pi^{\exp}$. We also compute the value function V for the task policy π, using all available targets from $\mathcal{Q}$ as terminal nodes. This policy attempts to reach the goal optimally, without performing exploratory actions. It is therefore used in the computation of exploration scores, because there we are only interested in the optimistic rewards-to-goal.

4.4.5 Action Selection and Incorporation of Actual Experience

At the current node x the agent selects an outgoing edge $e = (x, x')$ through its exploration policy $\pi^{\exp}$, which is derived stochastically (e.g. ε-greedy) from $V^{\exp}$. The local controller $a(x, x')$ then moves towards x'. If the agent reaches a small neighborhood around x' the controller is deactivated, and the reward of the traversed edge in $\mathcal{G}$ is updated. If the local controller does not reach the vicinity of x' within a given maximum time, the controller stops at a state $\hat{x}'$. We then delete the edge $e = (x, x')$ from the graph $\mathcal{G}$, since it cannot be completed by a local controller, and insert an edge from x to $\hat{x}'$ instead.

4.4.6 Inserting New Nodes: `insert_new_node`

When a node x' is visited for the first time, it is inserted as a new node into the graph. Local controllers to and from all nodes in a certain neighborhood around x' are simulated to create incoming and outgoing edges. If a connection seems possible we insert the edge into $\mathcal{G}$ and store an optimistic estimate of the reward, e.g. the negative estimated transition time of the local controller in absence of obstacles. Inserting a new node x' also invalidates existing exploration nodes in the neighborhood, if their exploration score variance would fall below the threshold $\theta^{\exp}_{\min}$ (see Section 4.4.2).

If a newly inserted edge $e = (x', x'')$ with estimated reward $\hat{r}(e)$ reduces the path costs from x^S to x'', the edge becomes an attractive target for exploration. We then insert e as an exploration edge into the queue $\mathcal{Q}$. The exploration score is $\sigma(e) = c(x^S, x') + \hat{r}(e) + V(x'')$, which is the estimated return of a path from x^S to X^G that uses e. For the exploration policy the agent may then equally select exploration nodes or edges as its best exploration targets.

4.4.7 Practical Implementation Issues

Efficient data structures like *kd*-trees reduce the search time for neighbors during the training phase. The CPU time is still higher than for model-based RL methods with fixed discretizations, e.g. Prioritized Sweeping (Moore & Atkeson, 1993). The construction of an adaptive state graph is an overhead, but on the other hand, it permits better solutions and faster learning.

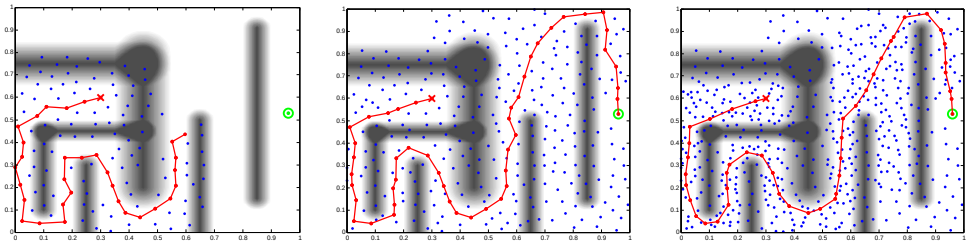
4.5 Experiments

In this section we show that our algorithm can solve several continuous control problems that are challenging for standard reinforcement learning techniques. We show that the algorithm requires less actual experience than existing methods and finds more accurate trajectories.

4.5.1 Static Puddle World

The puddle world task is a well-known benchmark for reinforcement learning algorithms in continuous domains. The objective is to navigate from a given starting state to a goal state in a 2-dimensional environment which contains *puddles*, representing regions of negative reward. Every transition inflicts a reward equal to the negative required time, plus additional penalties for entering a puddle area. The puddles are oval shapes, and the negative reward for entering a puddle is proportional to the distance inside the puddle. The 2-dimensional control action $u = (v_x, v_y)$ corresponds to setting velocities in x and y directions, leading to the simple linear system dynamics $(\dot{x}, \dot{y}) = (v_x, v_y)$. We can safely assume to know this dynamics, but planning a path to the goal state and avoiding the unknown puddles remains a difficult task.

Figure 4.2 shows various stages of the exploration process in a maze-like puddle world with multiple puddles. As optimistic value estimate $\hat{V}(\tilde{x})$ we use the negative time needed for the direct path to the goal (ignoring any puddles). In Figure 4.2(a) it can be observed that the agent directs its initial exploration towards the goal, while avoiding paths through regions of negative reward. Less promising regions like the upper left part are avoided. When the agent has reached the goal the first time (Figure 4.2(b)) the agent knows a coarse path to the goal. With continuing learning time, the agent refines the graph and adds more nodes in relevant regions,



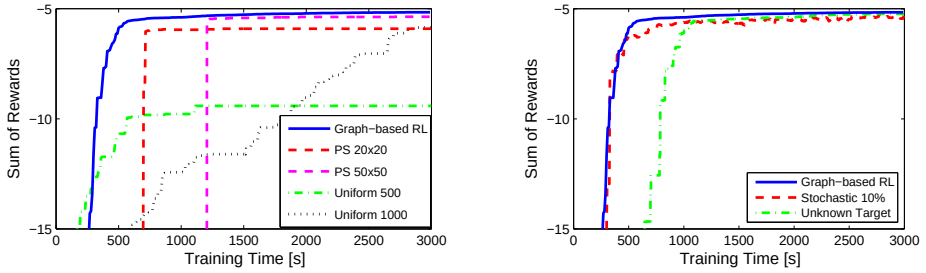
(a) (b) (c)

Figure 4.2: Static Puddle World: (a) and (b) shows the graph at the beginning of learning and when the agent has found the goal for the first time. (c) results from further optimization of the graph. The red line indicates the best known policy to the target.

which is illustrated in Figure 4.2(c). The path is almost optimal and avoids all puddles on the way to the goal.

Standard TD-learning (Sutton & Barto, 1998) with CMAC or RBF function approximation needs several thousands of episodes to converge on this task, because a rather fine discretization is required. It is therefore not considered for comparison. Better results were achieved by Prioritized Sweeping (Moore & Atkeson, 1993), a model-based RL algorithm which discretizes the environment and learns the transition and reward model from experience. In Figure 4.3 we compare the performance of RL with adaptive state graphs to prioritized sweeping with different discretization densities. We also compare the performance of a greedy search method on a graph with 500 and 1000 uniformly sampled nodes, which updates its reward estimates after every step. We evaluate the performance of the agent by measuring the sum of rewards obtained by its greedy policy at different training times. The training time is the total amount of time spent by the agent for actually moving within the state space during the training process. Figure 4.3(a) shows that the graph-based RL algorithm achieves reasonable performance faster than prioritized sweeping (even with coarse discretization), and the best found policy slightly outperforms all other methods. Our refined graph in the end contains about 730 nodes, which is approximately a fourth of the number of states used by prioritized sweeping on the fine grid. Greedy search on estimated edges initially finds the goal faster, but it either converges to a suboptimal policy, which is due to the uniform sampling, or needs longer to optimize its policy.

In Figure 4.3(b) we added small Gaussian movement noise (variance is 10% of movement velocity), and used local feedback-controllers. Our algorithm still converges quickly, but due to the stochasticity it cannot reach the same performance as in the deterministic case. We also investigated the (deterministic) problem in which the goal state is unknown. Since the agent has to explore uniformly in the beginning, it needs longer to converge, but ultimately reaches the same performance level.



(a) (b)

Figure 4.3: Learning performance on static puddle world from Figure 4.2. (a) Comparison of RL with adaptive state graphs to prioritized sweeping (PS), and greedy search on uniformly sampled nodes (Uniform) with different discretization densities. (b) Influence of stochasticity (10% movement noise) and unknown target states on performance of graph-based RL. (Average over 10 trials.)

4.5.2 3-Link Arm Reaching Task

The joints of a simulated planar 3-link robot arm are steered under static stability constraints in an environment with several obstacles (see Figure 4.4). The objective is to reach a goal area with the tip. The robot consist of a body (point mass with 1 kg), around which the arm - modeled as upper arm (length 0.5 m / weight 0.2 kg), fore arm (0.5 m / 0.1 kg) and hand (0.2 m / 0.05 kg) - can rotate. The center of mass (CoM) of the robot needs to be kept inside a finite support polygon. If the CoM leaves a neutral zone of guaranteed stability ($[-0.2, 0.2]$ in x and $[-0.1, 0.1]$ in y), the agent receives negative reward that grows quadratically as the CoM approaches the boundary of the support polygon. Under these constraints the trivial solution of rotating the arm around the top left obstacle achieves lower reward than the trajectory that maneuvers the arm through the narrow passage between the obstacles.

The 3-dimensional state space consists of the three joint angles, and the control actions correspond to setting the angular velocities. The approximative model $\hat{f}$ is a simple linear model, but the true system dynamics f contains nonlinearities due to obstacles, which are not captured by $\hat{f}$. The optimistic value estimate $\hat{V}(x)$ is the negative time needed by a local controller to reach a target configuration, calculated by simple inverse kinematics. Figure 4.5 shows that graph-based RL converges much faster to more accurate trajectories than prioritized sweeping with different levels of discretization.

4.6 Conclusion

We introduced a new efficient combination of reinforcement learning and sampling-based planning for continuous control problems in unknown environments. We use minimal prior knowledge in the form of approximative models and local con-

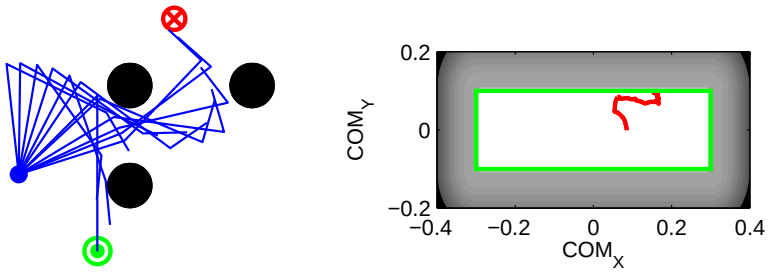


Figure 4.4: Arm reaching task with stability constraints. Left: Solution trajectory found by our algorithm. The agent must reach the goal region (red) from the starting position (green), avoiding the obstacles. Right: Trajectory of the CoM of the robot (red) inside the neutral zone (green).

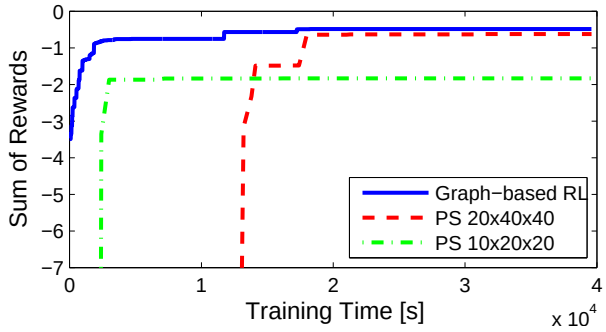


Figure 4.5: Learning performance on the 3-link arm reaching task for RL with adaptive state graphs and prioritized sweeping (PS) with different discretization densities. (Average over 10 trials)

trollers to increase the learning speed. Our algorithm builds an adaptive state graph through goal-directed exploration. We demonstrated on various movement planning tasks with difficult reward functions that RL with adaptive state graphs requires less actual experience than existing methods to obtain high quality solutions. The approach is particularly promising for complicated tasks that can be projected to low dimensional representations, such as balancing humanoid robots using motion primitives (Hauser et al., 2007).

4.7 Acknowledgments

This chapter is based on the paper *Efficient Continuous-Time Reinforcement Learning with Adaptive State Graphs*, which was written by Gerhard Neumann (GN),

myself (MP), and Wolfgang Maass (WM). The presented algorithm was jointly developed by GN and MP, the experiments were performed by MP and GN, and the manuscript was prepared by MP and GN with additional input from WM.

Chapter 5

Hebbian Learning of Bayes Optimal Decisions

Contents

Uncertainty is omnipresent when we perceive or interact with our environment, and the Bayesian framework provides computational methods for dealing with it. Mathematical models for Bayesian decision making typically require data-structures that are hard to implement in neural networks. This study shows that even the simplest and experimentally best supported type of synaptic plasticity, Hebbian learning, in combination with a sparse, redundant neural code, can in principle learn to infer optimal Bayesian decisions. We present a concrete Hebbian learning rule operating on log-probability ratios, which has strong convergence guarantees. Empirically the Hebbian rule learns very fast for a variety of tasks, and in combination with appropriate population codes for the inputs obtains optimal performance. In the subsequent Chapter 6 we show that a similar Hebbian plasticity rule, modulated by reward-signals, also provides a new perspective for understanding how Bayesian inference could support fast reinforcement learning in the brain.

5.1 Introduction

Evolution is likely to favor those biological organisms which are able to maximize the chance of achieving correct decisions in response to multiple unreliable sources of evidence. Hence one may argue that probabilistic inference, rather than logical inference, is the "mathematics of the mind", and that this perspective may help us to understand the principles of computation and learning in the brain (Rao, 2007). Bayesian inference, or equivalently inference in Bayesian networks (Bishop, 2006) is the most commonly considered framework for probabilistic inference, and a mathematical theory for learning in Bayesian networks has been developed.

Various attempts to relate these theoretically optimal models to experimentally supported models for computation and plasticity in networks of neurons in the brain have been made. Rao (2007) models Bayesian inference through an approximate implementation of the Belief Propagation algorithm (see (Bishop, 2006)) in a network of spiking neurons. For reduced classes of probability distributions, Deneve (2008a) proposed a method for spiking network models to learn Bayesian inference with an online approximation to an EM algorithm. The approach of Sandberg et al. (2002) interprets the weight w_{ji} of a synaptic connection between neurons representing the random variables x_i and x_j as $\log \frac{p(x_i,x_j)}{p(x_i)\cdot p(x_j)}$, and presents algorithms for learning these weights.

Neural correlates of variables that are important for decision making under uncertainty had been presented e.g. in the recent experimental study by Yang and Shadlen (2007). In their study they found that firing rates of neurons in area LIP of macaque monkeys reflect the log-likelihood ratio (or log-odd) of the outcome of a binary decision, given visual evidence. The learning of such log-odds for Bayesian decision making can be reduced to learning weights for a linear classifier, given an appropriate but fixed transformation from the input to possibly nonlinear features (Roth, 1999a). We show that the optimal weights for the linear decision function are actually log-odds themselves, and the definition of the features determines the assumptions of the learner about statistical dependencies among inputs.

In this work we show that simple Hebbian learning (Hebb, 1949) is sufficient to implement learning of Bayes optimal decisions for arbitrarily complex probability distributions. We present and analyze a concrete learning rule, which we call the *Bayesian Hebb rule*, and show that it provably converges towards correct log-odds. In combination with appropriate preprocessing networks this implements learning of different probabilistic decision making processes like e.g. Naive Bayesian classification. In Chapter 6 a reward-modulated version of this this Hebbian learning rule is presented, which can solve simple reinforcement learning tasks, and also provides a model for the experimental results of Yang and Shadlen (2007).

5.2 A Hebbian rule for learning log-odds

We consider the model of a linear threshold neuron with output y_0, where $y_0 = 1$ means that the neuron is firing and $y_0 = 0$ means non-firing. The neuron's current decision $\hat{y_0}$ whether to fire or not is given by a linear decision function $\hat{y_0} = \text{sign}(w_0 \cdot constant + \sum_{i=1}^{n} w_i y_i)$, where the y_i are the current firing states of all presynaptic neurons and w_i are the weights of the corresponding synapses.

We propose the following learning rule, which we call the Bayesian Hebb rule:

$$\boxed{\Delta w_i = \begin{cases} \eta\,(1+e^{-w_i}), & \text{if } y_0 = 1 \text{ and } y_i = 1 \\ -\eta\,(1+e^{w_i}), & \text{if } y_0 = 0 \text{ and } y_i = 1 \\ 0, & \text{if } y_i = 0. \end{cases}} \tag{5.1}$$

This learning rule is purely local, i.e. it depends only on the binary firing state of

the pre- and postsynaptic neuron y_i and y_0, the current weight w_i and a learning rate η. Under the assumption of a stationary joint probability distribution of the pre- and postsynaptic firing states $y_0, y_1, \ldots, y_n$ the Bayesian Hebb rule learns log-probability ratios of the postsynaptic firing state y_0, conditioned on a corresponding presynaptic firing state y_i. We consider here the use of the rule in a supervised, teacher forced mode (see Section 5.3). A reward-modulated version of this rule for reinforcement learning is presented in Chapter 6. We will prove that the rule converges globally to the target weight value w_i^*, given by

$$w_i^* = \log \frac{p(y_0 = 1 | y_i = 1)}{p(y_0 = 0 | y_i = 1)} \quad . \tag{5.2}$$

We first show that the expected update $\mathrm{E}[\Delta w_i]$ under (5.1) vanishes at the target value w_i^*:

$$\begin{aligned} \mathrm{E}[\Delta w_i^*] = 0 \;&\Leftrightarrow\; p(y_0{=}1, y_i{=}1)\eta(1 + e^{-w_i^*}) - p(y_0{=}0, y_i{=}1)\eta(1 + e^{w_i^*}) = 0 \\ &\Leftrightarrow\quad \frac{1 + e^{w_i^*}}{1 + e^{-w_i^*}} = \frac{p(y_0{=}1, y_i{=}1)}{p(y_0{=}0, y_i{=}1)} \\ &\Leftrightarrow\quad w_i^* = \log \frac{p(y_0{=}1 | y_i{=}1)}{p(y_0{=}0 | y_i{=}1)} \quad . \end{aligned} \tag{5.3}$$

Since the above is a chain of equivalence transformations, this proves that w_i^* is the only equilibrium value of the rule. The weight vector $\mathbf{w}^*$ is thus a global point-attractor with regard to expected weight changes of the Bayesian Hebb rule (5.1) in the n-dimensional weight-space $\mathbb{R}^n$.

Furthermore we show, using the result from (5.3), that the expected weight change at any current value of w_i points in the direction of w_i^*. Consider some arbitrary intermediate weight value $w_i = w_i^* + 2\varepsilon$:

$$\begin{aligned} \mathrm{E}[\Delta w_i]|_{w_i^*+2\varepsilon} &= \mathrm{E}[\Delta w_i]|_{w_i^*+2\varepsilon} - E[\Delta w_i]|_{w_i^*} \\ &\propto p(y_0{=}1, y_i{=}1)e^{-w_i^*}(e^{-2\varepsilon} - 1) - p(y_0{=}0, y_i{=}1)e^{w_i^*}(e^{2\varepsilon} - 1) \\ &= (p(y_0{=}0, y_i{=}1)e^{-\varepsilon} + p(y_0{=}1, y_i{=}1)e^{\varepsilon})(e^{-\varepsilon} - e^{\varepsilon}) \quad . \end{aligned} \tag{5.4}$$

The first factor in (5.4) is always non-negative, hence $\varepsilon < 0$ implies $E[\Delta w_i] > 0$, and $\varepsilon > 0$ implies $E[\Delta w_i] < 0$. The Bayesian Hebb rule is therefore always expected to perform updates in the right direction, and the initial weight values or perturbations of the weights decay exponentially fast.

Already after having seen a finite set of examples $\langle y_0, \ldots, y_n \rangle \in \{0, 1\}^{n+1}$, the Bayesian Hebb rule closely approximates the optimal weight vector $\hat{\mathbf{w}}$ that can be inferred from the data. A traditional frequentist's approach would use counters $a_i = \#[y_0{=}1 \wedge y_i{=}1]$ and $b_i = \#[y_0{=}0 \wedge y_i{=}1]$ to estimate every w_i^* by

$$\hat{w}_i = \log \frac{a_i}{b_i} \; . \tag{5.5}$$

A Bayesian approach would model $p(y_0|y_i)$ with an (initially flat) *Beta*-distribution, and use the counters a_i and b_i to update this belief (Bishop, 2006), leading to the

same MAP estimate $\hat{w}_i$. Consequently, in both approaches a new example with $y_0 = 1$ and $y_i = 1$ leads to the update

$$\hat{w}_i^{new} = \log \frac{a_i + 1}{b_i} = \log \frac{a_i}{b_i} \left(1 + \frac{1}{a_i}\right) = \hat{w}_i + \log(1 + \frac{1}{N_i}(1 + e^{-\hat{w}_i})) \ , \qquad (5.6)$$

where $N_i := a_i + b_i$ is the number of previously processed examples with $y_i = 1$, thus $\frac{1}{a_i} = \frac{1}{N_i}(1 + \frac{b_i}{a_i})$. Analogously, a new example with $y_0 = 0$ and $y_i = 1$ gives rise to the update

$$\hat{w}_i^{new} = \log \frac{a_i}{b_i + 1} = \log \frac{a_i}{b_i} \left(\frac{1}{1 + \frac{1}{b_i}}\right) = \hat{w}_i - \log(1 + \frac{1}{N_i}(1 + e^{\hat{w}_i})). \qquad (5.7)$$

Furthermore, $\hat{w}_i^{new} = \hat{w}_i$ for a new example with $y_i = 0$. Using the approximation $\log(1 + \alpha) \approx \alpha$ the update rules (5.6) and (5.7) yield the Bayesian Hebb rule (5.1) with an adaptive learning rate $\eta_i = \frac{1}{N_i}$ for each synapse.

In fact, a result of Robbins-Monro (see (Bertsekas & Tsitsiklis, 1996) for a review) implies that the updating of weight estimates $\hat{w}_i$ according to (5.6) and (5.7) converges to the target values w_i^* not only for the particular choice $\eta_i^{(N_i)} = \frac{1}{N_i}$, but for any sequence $\eta_i^{(N_i)}$ that satisfies $\sum_{N_i=1}^{\infty} \eta_i^{(N_i)} = \infty$ and $\sum_{N_i=1}^{\infty} (\eta_i^{(N_i)})^2 < \infty$. More than that the Supermartingale Convergence Theorem (see (Bertsekas & Tsitsiklis, 1996)) guarantees convergence in distribution even for a sufficiently small constant learning rate.

5.2.1 Learning rate adaptation

One can see from the above considerations that the Bayesian Hebb rule with a constant learning rate η converges globally to the desired log-odds. A too small constant learning rate, however, tends to slow down the initial convergence of the weight vector, and a too large constant learning rate produces larger fluctuations once the steady state is reached.

(5.6) and (5.7) suggest a decaying learning rate $\eta_i^{(N_i)} = \frac{1}{N_i}$, where N_i is the number of preceding examples with $y_i = 1$. We will present a learning rate adaptation mechanism that avoids biologically implausible counters, and is robust enough to deal even with non-stationary distributions.

Since the Bayesian Hebb rule and the Bayesian approach of updating *Beta*-distributions for conditional probabilities are closely related, it is reasonable to expect that the distribution of weights w_i over longer time periods with a non-vanishing learning rate will resemble a $Beta(a_i, b_i)$-distribution transformed to the log-odd domain. The parameters a_i and b_i in this case are not exact counters anymore but correspond to virtual sample sizes, depending on the current learning rate. We formalize this statistical model of w_i by

$$\sigma(w_i) = \frac{1}{1 + e^{-w_i}} \sim Beta(a_i, b_i) \iff w_i \sim \frac{\Gamma(a_i + b_i)}{\Gamma(a_i)\Gamma(b_i)} \sigma(w_i)^{a_i} \sigma(-w_i)^{b_i},$$

In practice this model turned out to capture quite well the actually observed quasi-stationary distribution of w_i. Analytically it can be proven that $\mathrm{E}[w_i] \approx \log \frac{a_i}{b_i}$ and $\mathrm{Var}[w_i] \approx \frac{1}{a_i} + \frac{1}{b_i}$. A learning rate adaptation mechanism at the synapse that keeps track of the observed mean and variance of the synaptic weight can therefore recover estimates of the virtual sample sizes a_i and b_i. The following mechanism, which we call *variance tracking* implements this by computing running averages of the weights and the squares of weights in $\bar{w}_i$ and $\bar{q}_i$:

$$\boxed{\begin{array}{lcll} \eta_i^{new} & \leftarrow & \frac{\bar{q}_i - \bar{w}_i^2}{1+\cosh \bar{w}_i} & \\ \bar{w}_i^{new} & \leftarrow & (1-\eta_i)\,\bar{w}_i + \eta_i\, w_i & \\ \bar{q}_i^{new} & \leftarrow & (1-\eta_i)\,\bar{q}_i + \eta_i\, w_i^2 & . \end{array}} \tag{5.8}$$

In practice this mechanism decays like $\frac{1}{N_i}$ under stationary conditions, but is also able to handle changing input distributions. It was used in all presented experiments for the Bayesian Hebb rule.

5.3 Hebbian learning of Bayesian decisions

We now show how the Bayesian Hebb rule can be used to learn Bayes optimal decisions. The first application is the Naive Bayesian classifier, where a binary target variable x_0 should be inferred from a vector of multinomial variables $\mathbf{x} = \langle x_1, \ldots, x_m \rangle$, under the assumption that the x_i's are conditionally independent given x_0, thus $p(x_0, \mathbf{x}) = p(x_0) \prod_1^m p(x_k|x_0)$ (see Figure 5.1 B for an example graphical representation of this distribution). Using basic rules of probability theory the posterior probability ratio for $x_0 = 1$ and $x_0 = 0$ can be derived:

$$\frac{p(x_0{=}1|\mathbf{x})}{p(x_0{=}0|\mathbf{x})} = \frac{p(x_0{=}1)}{p(x_0{=}0)} \prod_{k=1}^{m} \frac{p(x_k|x_0{=}1)}{p(x_k|x_0{=}0)} = \left(\frac{p(x_0{=}1)}{p(x_0{=}0)}\right)^{(1-m)} \prod_{k=1}^{m} \frac{p(x_0{=}1|x_k)}{p(x_0{=}0|x_k)} = \tag{5.9}$$

$$= \left(\frac{p(x_0{=}1)}{p(x_0{=}0)}\right)^{(1-m)} \prod_{k=1}^{m} \prod_{j=1}^{m_k} \left(\frac{p(x_0{=}1|x_k{=}j)}{p(x_0{=}0|x_k{=}j)}\right)^{I(x_k=j)} ,$$

where m_k is the number of different possible values of the input variable x_k, and the indicator function I is defined as $I(true) = 1$ and $I(false) = 0$.

Let the m input variables $x_1, \ldots, x_m$ be represented through the binary firing states $y_1, \ldots, y_n \in \{0, 1\}$ of the n presynaptic neurons in a population coding manner. More precisely, let each input variable $x_k \in \{1, \ldots, m_k\}$ be represented by m_k neurons, where each neuron fires only for one of the m_k possible values of x_k. Formally we define the simple preprocessing (*SP*)

$$\mathbf{y}^\mathsf{T} = \left[\boldsymbol{\phi}(x_1)^\mathsf{T}, \ldots, \boldsymbol{\phi}(x_m)^\mathsf{T}\right] \quad \text{with} \quad \boldsymbol{\phi}(x_k)^\mathsf{T} = [I(x_k = 1), \ldots, I(x_k = m_k)] \ . \tag{5.10}$$

The binary target variable x_0 is represented directly by the binary state y_0 of the postsynaptic neuron. Substituting the state variables $y_0, y_1, \ldots, y_n$ in (5.9) and taking the logarithm leads to

$$\log \frac{p(y_0 = 1|\mathbf{y})}{p(y_0 = 0|\mathbf{y})} = (1-m) \log \frac{p(y_0 = 1)}{p(y_0 = 0)} + \sum_{i=1}^{n} y_i \log \frac{p(y_i = 1|y_0 = 1)}{p(y_i = 1|y_0 = 0)} \; .$$

Hence the optimal decision under the Naive Bayes assumption is

$$\hat{y_0} = \text{sign}((1-m)w_0^* + \sum_{i=1}^{n} w_i^* \; y_i) \quad .$$

The optimal weights w_0^* and w_i^*

$$w_0^* = \log \frac{p(y_0 = 1)}{p(y_0 = 0)} \qquad \text{and} \qquad w_i^* = \log \frac{p(y_0 = 1|y_i = 1)}{p(y_0 = 0|y_i = 1)} \qquad \text{for} \qquad i = 1, \ldots, n.$$

are obviously log-odds which can be learned by the Bayesian Hebb rule (the bias weight w_0 is simply learned as an unconditional log-odd).

5.3.1 Learning Bayesian decisions for arbitrary distributions

We now address the more general case, where conditional independence of the input variables $x_1, \ldots, x_m$ cannot be assumed. In this case the dependency structure of the underlying distribution is given in terms of an arbitrary Bayesian network BN for discrete variables (see e.g. Figure 5.1 A). Without loss of generality we choose a numbering scheme of the nodes of the BN such that the node to be learned is x_0 and its direct children are $x_1, \ldots, x_{m'}$. This implies that the BN can be described by $m+1$ (possibly empty) parent sets defined by

$$\mathtt{P}_k \;\; = \;\; \{i \mid \text{a directed edge } x_i \to x_k \text{ exists in BN and } \; i \geq 1\} \quad .$$

The joint probability distribution on the variables $x_0, \ldots, x_m$ in BN can then be factored and evaluated for $x_0 = 1$ and $x_0 = 0$ in order to obtain the probability ratio

$$\frac{p(x_0 = 1, \mathbf{x})}{p(x_0 = 0, \mathbf{x})} = \frac{p(x_0 = 1|\mathbf{x})}{p(x_0 = 0|\mathbf{x})} = \frac{p(x_0 = 1|\mathbf{x}_{\mathtt{P}_0})}{p(x_0 = 0|\mathbf{x}_{\mathtt{P}_0})} \prod_{k=1}^{m'} \frac{p(x_k|\mathbf{x}_{\mathtt{P}_k}, x_0 = 1)}{p(x_k|\mathbf{x}_{\mathtt{P}_k}, x_0 = 0)} \prod_{k=m'+1}^{m} \frac{p(x_k|\mathbf{x}_{\mathtt{P}_k})}{p(x_k|\mathbf{x}_{\mathtt{P}_k})} \quad .$$

Obviously, the last term cancels out, and by applying Bayes' rule and taking the logarithm the target log-odd can be expressed as a sum of conditional log-odds only:

$$\log \frac{p(x_0{=}1|\mathbf{x})}{p(x_0{=}0|\mathbf{x})} = \log \frac{p(x_0{=}1|\mathbf{x}_{\mathtt{P}_0})}{p(x_0{=}0|\mathbf{x}_{\mathtt{P}_0})} + \sum_{k=1}^{m'} \left(\log \frac{p(x_0{=}1|x_k, \mathbf{x}_{\mathtt{P}_k})}{p(x_0{=}0|x_k, \mathbf{x}_{\mathtt{P}_k})} - \log \frac{p(x_0{=}1|\mathbf{x}_{\mathtt{P}_k})}{p(x_0{=}0|\mathbf{x}_{\mathtt{P}_k})} \right). \tag{5.11}$$

We now develop a suitable sparse encoding of of $x_1, \ldots, x_m$ into binary variables $y_1, \ldots, y_n$ (with $n \gg m$) such that the decision function (5.11) can be written

as a weighted sum, and the weights correspond to conditional log-odds of y_i's. Figure 5.1 C illustrates such a sparse code: One binary variable is created for every possible value assignment to a variable and all its parents, and one additional binary variable is created for every possible value assignment to the parent nodes only. Formally, the previously introduced population coding operator ϕ is generalized such that $\phi(x_{i_1}, x_{i_2}, \ldots, x_{i_l})$ creates a vector of length $\prod_{j=1}^{l} m_{i_j}$ that equals zero in all entries except for one 1-entry which identifies by its position in the vector the present assignment of the input variables $x_{i_1}, \ldots, x_{i_l}$. The concatenation of all these population coded groups is collected in the vector $\mathbf{y}$ of length n

$$\mathbf{y}^\mathsf{T} = \left[\phi(\mathbf{x}_{\mathbf{P}_0})^\mathsf{T}, \phi(x_1, \mathbf{x}_{\mathbf{P}_1})^\mathsf{T}, -\phi(\mathbf{x}_{\mathbf{P}_1})^\mathsf{T}, \ldots, \phi(x_m, \mathbf{x}_{\mathbf{P}_m})^\mathsf{T}, -\phi(\mathbf{x}_{\mathbf{P}_m})^\mathsf{T}\right] \quad . \tag{5.12}$$

The negated vector parts in (5.12) correspond to the negative coefficients in the sum in (5.11). Inserting the sparse coding (5.12) into (5.11) allows writing the Bayes optimal decision function (5.11) as a pure sum of log-odds of the target variable:

$$\hat{x_0} = \hat{y_0} = \text{sign}(\sum_{i=1}^{n} w_i^* y_i), \qquad \text{with} \qquad w_i^* = \log \frac{p(y_0{=}1|y_i{\neq}0)}{p(y_0{=}0|y_i{\neq}0)} \quad .$$

Every synaptic weight w_i can be learned efficiently by the Bayesian Hebb rule (5.1) with the formal modification that the update is not only triggered by $y_i{=}1$ but in general whenever $y_i{\neq}0$ (which obviously does not change the behavior of the learning process). A neuron that learns with the Bayesian Hebb rule on inputs that are generated by the generalized preprocessing (GP) defined in (5.12) therefore approximates the Bayes optimal decision function (5.11), and converges quite fast to the best performance that any probabilistic inference could possibly achieve (see Figure 5.2B).

5.4 Experimental Results

We have tested the Bayesian Hebb rule on 400 different prediction tasks, each of them defined by a general (non-Naive) Bayesian network of 7 binary variables. The networks were randomly generated by the algorithm of Ide and Cozman (2002). From each network we sampled 2000 training and 5000 test examples, and measured the percentage of correct predictions after every update step.
The performance of the predictor was compared to the Bayes optimal predictor, and to online logistic regression, which fits a linear model by gradient descent on the cross-entropy error function. This non-Hebbian learning approach is in general the best performing online learning approach for linear discriminators (Bishop, 2006). Figure 5.2A shows that the Bayesian Hebb rule with the simple preprocessing (5.10) generalizes better from a few training examples, but is outperformed by logistic regression in the long run, since the Naive Bayes assumption is not met. With the generalized preprocessing (5.12), the Bayesian Hebb rule learns fast and converges to the Bayes optimum (see Figure 5.2B). In Figure 5.2C we show that the Bayesian Hebb rule is robust to noisy updates - a condition very likely to occur in biological

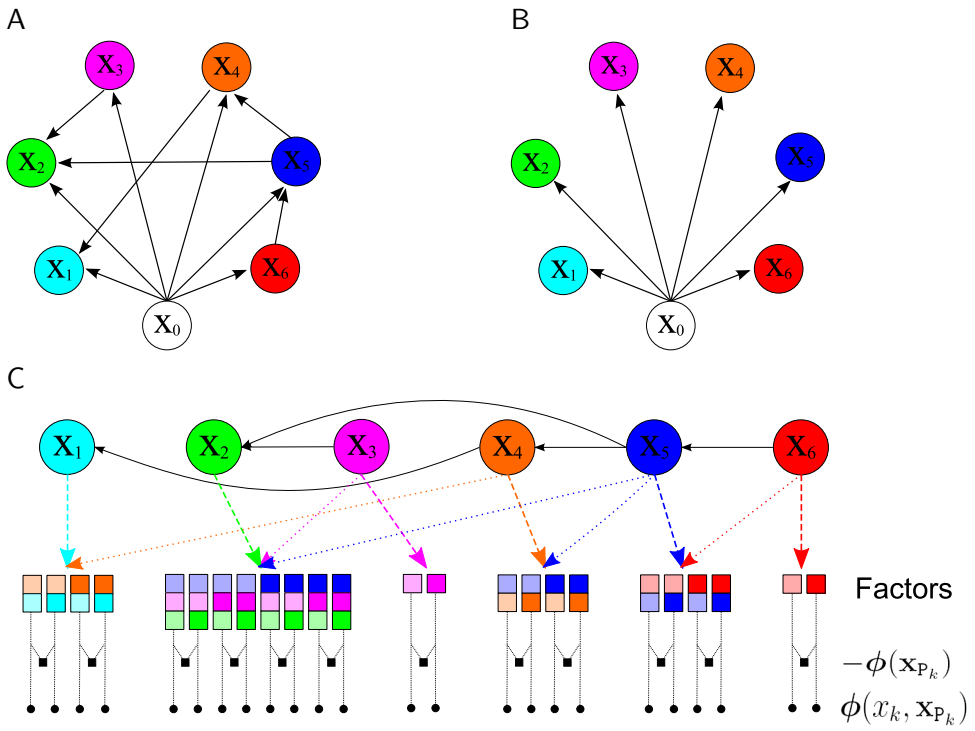


Figure 5.1: **A)** An example Bayesian network with general connectivity. **B)** Representation of the same Bayesian network under the Naive Bayes assumption. **C)** Population coding applied to the Bayesian network shown in panel A. For each combination of values of the variables $\{x_k, \mathbf{x}_{\mathrm{P}_k}\}$ of a factor there is exactly one neuron (indicated by a black circle) associated with the factor that outputs the value 1. In addition OR's of these values are computed (black squares). We refer to the resulting preprocessing circuit as generalized preprocessing (GP).

systems. We modified the weight update Δw_i such that it was uniformly distributed in the interval $\Delta w_i \pm \gamma\%$. Even such imprecise implementations of the Bayesian Hebb rule perform very well. Similar results can be obtained if the exp-function in (5.1) is replaced by a low-order Taylor approximation.

5.5 Discussion

We have shown that the simplest and experimentally best supported local learning mechanism, Hebbian learning, is sufficient to learn Bayes optimal decisions. We have introduced and analyzed the Bayesian Hebb rule, a training method for synaptic weights, which converges fast and robustly to optimal log-probability ratios, without requiring any communication between plasticity mechanisms for different synapses. We have shown how the same plasticity mechanism can learn Bayes optimal decisions under different statistical independence assumptions, if it is provided with an appropriately preprocessed input. We have demonstrated on a

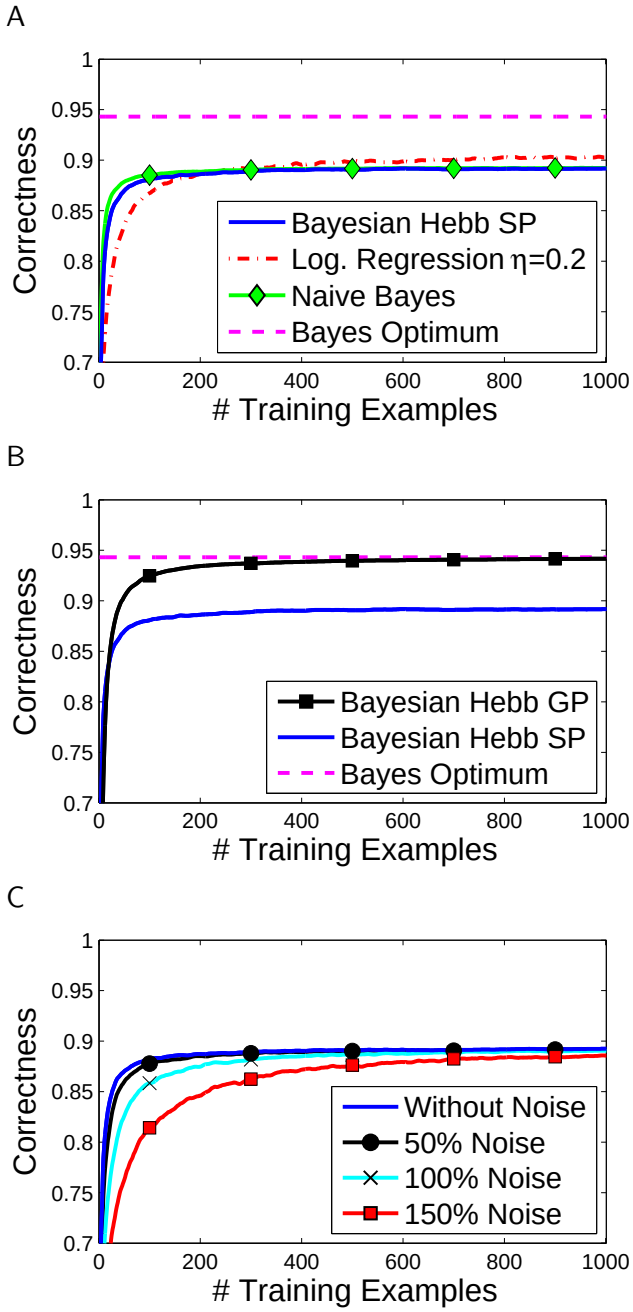


Figure 5.2: Performance comparison for prediction tasks. **A)** The Bayesian Hebb rule with simple preprocessing (*SP*) learns as fast as Naive Bayes, and faster than logistic regression (with optimized constant learning rate). **B)** The Bayesian Hebb rule with generalized preprocessing (*GP*) learns fast and converges to the Bayes optimal prediction performance. **C)** Even a very imprecise implementation of the Bayesian Hebb rule (noisy updates, uniformly distributed in $\Delta w_i \pm \gamma\%$) yields almost the same learning performance.

variety of prediction tasks that the Bayesian Hebb rule learns very fast, and with an appropriate sparse preprocessing mechanism for groups of statistically dependent features its performance converges to the Bayes optimum. Our approach therefore suggests that sparse, redundant codes of input features may simplify synaptic learning processes in spite of strong statistical dependencies. In the subsequent Chapter 6 we show that Hebbian learning also suffices for simple instances of reinforcement learning. The Bayesian Hebb rule, modulated by a signal related to rewards, enables fast learning of optimal action selection. Experimental results of Yang and Shadlen (2007) on reinforcement learning of probabilistic inference in primates can be partially modeled in this way with regard to resulting behaviors.

An attractive feature of the Bayesian Hebb rule is its ability to deal with the addition or removal of input features through the creation or deletion of synaptic connections, since no relearning of weights is required for the other synapses. In contrast to discriminative neural learning rules, our approach is generative, which according to Ng and Jordan (2002) leads to faster generalization. Therefore the learning rule may be viewed as a potential building block for models of the brain as a self-organizing and fast adapting probabilistic inference machine.

5.6 Acknowledgments

This chapter is based on the paper *Hebbian learning of Bayes optimal decisions*, which was written by Bernhard Nessler (BN), myself (MP), and Wolfgang Maass (WM). The theory of the Bayesian Hebb rule was developed by BN with input from WM and MP, the experiments were performed by MP, and the paper was written by MP and BN with additional input from WM.

Chapter 6

Reward-modulated Hebbian Learning of Decision Making

Contents

We introduce a framework for decision making in which the learning of decision making is reduced to its simplest and biologically most plausible form: Hebbian learning on a linear neuron. Extending the supervised framework introduced in Chapter 5, we cast our Bayesian-Hebb learning rule as reinforcement learning in which certain decisions are rewarded, and prove that each synaptic weight will on average converge exponentially fast to the log-odd of receiving a reward when its pre- and post-synaptic neurons are active. In our simple architecture, a particular action is selected from the set of candidate actions by a winner-take-all operation. The global reward assigned to this action then modulates the update of each synapse. Apart from this global reward signal our reward-modulated Bayesian Hebb rule is a pure Hebb update that depends only on the co-activation of the pre- and postsynaptic neurons, and not on the weighted sum of all presynaptic inputs to the post-synaptic neuron as in the perceptron learning rule or the Rescorla-Wagner rule. This simple approach to action-selection learning requires that information about sensory inputs be presented to the Bayesian decision stage in a suitably pre-processed form resulting from other adaptive processes (acting on a larger time scale) that detect salient dependencies among input features. Hence our proposed framework for fast learning of decisions also provides interesting new hypotheses regarding neural nodes and computational goals of cortical areas that provide input to the final decision stage.

6.1 Introduction

A typical decision making task of an organism requires the evaluation of multiple alternative actions, with the goal of maximizing the probability of obtaining positive reward. If input signals provide only uncertain cues, and reward is obtained stochastically in response to actions, then Bayesian statistics provides a mathematical framework for the optimal integration of all available information. Bayes' theorem can be used to calculate the probability that an action yields a reward, given the current sensory input and the current internal state of an organism. The goal of this article is to present the simplest possible neural network model that can make such an evaluation, where simplicity is assessed both in terms of computational operations, and the complexity of the learning method.

A large number of experimental results suggest that animals do indeed make decisions based on Bayesian integration of information about stimulus-action-reward contingencies. For example, Sugrue et al. (2004) (see (Sugrue et al., 2005) for a review) e.g. have shown that monkeys use the *matching behavior strategy*, in which the frequency with which a particular action is chosen matches the expected reward for that action. Yang and Shadlen (2007) have shown that the previous experience of macaque monkeys in probabilistic decision tasks is represented by the firing rates of neurons in area LIP in the form of the log-likelihood ratio (or log-odd) of receiving a reward for a particular action a in response to a stimulus $\mathbf{x}$ (in an experiment where the monkey received in each trial either no reward, or a reward of unit size, depending on the choice of the monkey among two possible actions).

We show that an optimal action selection policy can be reduced to a Winner-Take-All (WTA) operation applied to linear gates, which receive suitably preprocessed inputs (see Figure 6.1). Furthermore, we show that the updating of the WTA circuit in the face of new evidence can be reduced to the application of a local reward-modulated Hebbian learning rule to each linear gate. We call this rule the Bayesian Hebb Rule. Despite the simplicity of this model, one can prove that it enables fast learning of near optimal decision making, which is remarkable because rigorous insight into convergence properties of Hebbian learning rules is often lacking.

WTA (see (Yuille & Geiger, 2003) for a review) is a very simple computational operation that selects the largest among l values $L_1, \ldots, L_l$. This selection is usually encoded through l binary outputs $z_1, \ldots, z_l$, where $z_a = 1$ if L_a is selected as the largest input (ties can be broken arbitrarily), else $z_a = 0$ (see Figure 6.1). In an action selection framework this output then triggers the selection of the a^{th} among l possible actions. Each value L_a is just a weighted sum

$$L_a = \sum_{i=0}^{n} w_{a,i}\ y_i$$

of variables $y_1, \ldots, y_n$ (and a dummy variable $y_0 \equiv 1$ that allows to use $w_{a,0}$ as

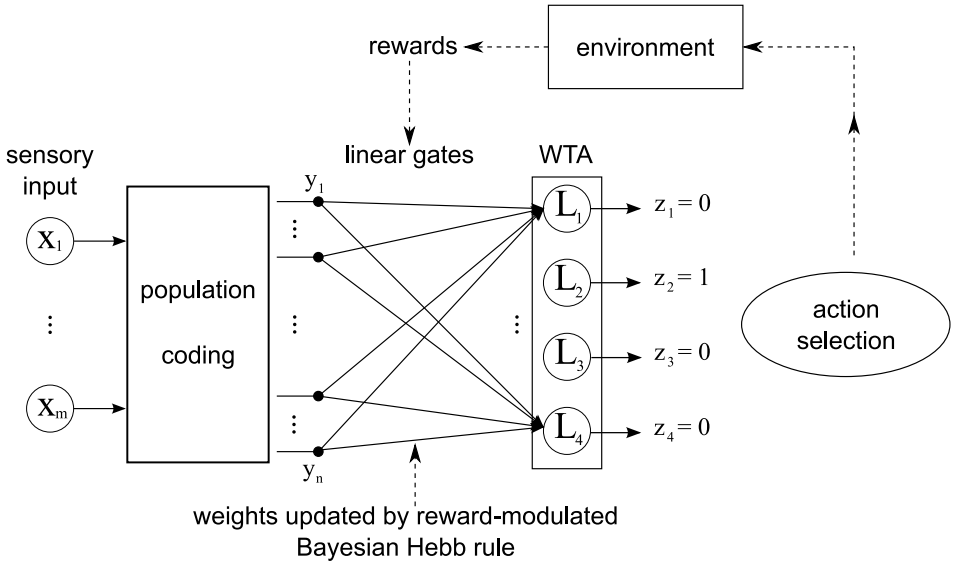


Figure 6.1: Winner-Take-All (WTA) architecture for learning of decision making. First, the multinomial input variables $x_1, \ldots, x_m$ are preprocessed by a fixed circuit (which implements some type of population coding) to yield binary variables $y_1, \ldots, y_n$. For every possible action a there is an associated linear neuron L_a which computes a weighted sum $\sum_{i=0}^{n} w_{a,i} y_i$ of the variables $y_1, \ldots, y_n$. The neuron L_a with the largest weighted sum "wins", i.e. $z_a = 1$, and action a is selected.

a bias). Despite its simplicity, the resulting WTA-circuit is computationally quite powerful (Maass, 2000).

The main contribution of this article is a novel learning algorithm for the weights $w_{a,i}$ of the linear gates L_a. We show that for a suitable fixed preprocessing (that transfers the original input variables x_k into binary variables y_i) the optimal value $w^*_{a,i}$ for the weight $w_{a,i}$ in Figure 6.1 is the log-likelihood ratio (or log-odd) of receiving a reward for a particular action a, provided that the binary feature y_i is activated by the preprocessing function, i.e.

$$w^*_{a,i} = \log \frac{p(r = 1 | y_i = 1, a)}{p(r = 0 | y_i = 1, a)} \quad . \tag{6.1}$$

In the asymptotic case, where all weights $w_{a,i}$ have converged to their respective target values $w^*_{a_i}$, the policy of the WTA-circuit in Figure 6.1 is optimal in the sense that for any input signal the action with the highest chance to deliver reward is chosen. We also show that after finitely many training trials steps the weights closely approximate the optimal weights that can be inferred from the previously observed data.

Our algorithm for reward-modulated learning of optimal weights uses only Hebbian learning, a form of learning for which there is strong experimental evidence (Abbott & Nelson, 2000; Fregnac, 2003; Caporale & Dan, 2008). Hebb (1949)

proposed (see (Fregnac, 2003) for a recent review) that a synapse from neuron A to neuron B is strengthened if A and B often fire together. But several studies have shown that Hebbian synaptic plasticity requires a third signal (often in the form of neuromodulators) in order to consolidate weight changes (Bailey et al., 2000; Reynolds et al., 2001; Farries & Fairhall, 2007; Legenstein et al., 2008). It is often assumed that the third signal provides information about reward or reward expectations. Hence learning rules involving these signals are referred to as reward-modulated learning rules.

Hebbian learning, such as in the proposed Bayesian Hebb rule should be contrasted with non-Hebbian learning rules such as the perceptron learning rule (also referred to as Delta-rule), or the Rescorla-Wagner rule (Rescorla & Wagner, 1972), which are harder to support on the basis of experimental data for synaptic plasticity. In these latter learning rules the change Δw_i of a synaptic weight w_i at a single synapse depends not only on the current activation values of the pre- and postsynaptic neuron and the current value of w_i (and possibly a reward-related third signal), but also on the current values of the other weights and the activation values of all other neurons that provide synaptic input to the same postsynaptic neuron (more precisely: on the value of the weighted sum of all presynaptic inputs).

We present a mechanism for reward-modulated local learning of the weights $w_{a,i}$ that permits them to converge (on average) to the ideal value (6.1). Learning from rewards is conceptionally different from learning with a supervisor that informs the learner about the correct choice. In reward-based learning, the learner must explore different actions multiple times, even if he assumes that other actions would be better in the given situation. This strategy is necessary to avoid premature convergence to suboptimal policies.

We want to make clear that in this article we do not study the learning of sequences of actions as in general reinforcement learning (Sutton & Barto, 1998), but investigate scenarios like in operant conditioning, where decisions have to be made based on learned immediate reward probabilities for single actions. We follow however the terminology proposed for example in (Dayan & Abbott, 2001), and subsume the latter also under the term reinforcement learning.

We will provide in this article a rigorous theoretical analysis of the convergence properties of the Bayesian Hebb rule. Because our learning rule makes online updates after every training trial, rather than performing a batch update after collecting a set of data, we are interested in the asymptotic behavior of the rule, as well as its online performance. Non-Hebbian learning rules usually perform gradient descent optimization along an error surface. If local minima exist on the error surface, this approach always carries the risk of becoming trapped in suboptimal solutions, from which it cannot escape. In contrast, the optimal values of the weights to be learned by the Bayesian Hebb rule act as global fixed point attractors in weight-space with regard to expected weight updates of the Bayesian Hebb rule. Our analysis shows that the weights learned during training are very close to the optimal values that can be inferred from finitely many training trials, and they converge exponentially fast to the optimal values. We will also demonstrate that an

extremely simple linear approximation to the Bayesian Hebb rule performs almost equally well.

Bayesian decision making combines information from many variables, and therefore must consider statistical dependencies amongst them. An influential paper by Roth (1999b) noted that decision making can be reduced to the computation of weighted sums, provided that the input signals are properly pre-processed (see also (Domingos & Pazzani, 1997)). This observation motivates our use of the neural network model shown in Figure 6.1. Roth (1999b) proved his results in the context of linear statistical queries for probabilistic classification. We now extend this approach to the case of policy learning by incorporating a WTA gate for action selection. Roth (1999b) noted that the set of features produced by the preprocessing function must be related to independence assumptions among input variables. We show that these features correspond to the factors in a factor graph (Kschischang et al., 2001) of the input- and reward distribution.

One particularly simple case is *Naive Bayes*, which assumes that all input variables are conditionally independent given one particular target variable, e.g. the occurrence of reward. In this case it is sufficient to know the reward-prediction probabilities for every input variable and every action separately, since then the reward probability given the complete input is the product of all individual predictors. We provide a simple preprocessing function for this case, which does not use any information about statistical dependencies of input variables, but leads to satisfactory policies.

The general case, in which there are statistical dependencies among input variables, requires more complex algorithms for Bayesian inference. Graphical models like Bayesian networks (Bishop, 2006) and factor graphs (Kschischang et al., 2001) are used to model conditional dependencies among variables, and inference algorithms operate by passing messages along edges of the graphs. Factor graphs are particularly useful tools. They consider groups of dependent variables as *factor nodes*, in which functions of all connected variable nodes are computed. Inference in these models is performed using the sum-product algorithm (Bishop, 2006; Kschischang et al., 2001), which is conceptually simpler than the belief propagation algorithms used for inference in general Bayesian networks. Recent work (Steimer, Maass, & Douglas, 2009) has shown that these factor nodes can be implemented in networks of spiking neurons. In this chapter we define an optimal generalized preprocessing function based on the factor graph representation of the reward distribution. This provides a concrete processing goal for multimodal integration in sensory areas, and links the theory of factor graphs to experimentally observed neural population codes. These codes, as all other components of our framework, are easily implemented in neural networks, and allow fast and robust learning with the Hebbian learning algorithms presented in this chapter.

We assume here that the graph structure of the underlying Bayesian network is known, but not the parameters of it (i.e., the probability distribution). We do not address the problem of structure learning, which is a very different task, and thus requires different algorithms. Whereas the parameters that define decision

strategies require very fast adaptation, statistical dependencies between inputs reflect invariances in the environment, which could be learned by separate learning processes on much longer time scales.

This chapter is organized as follows: We present the Bayesian Hebb rule for reinforcement learning tasks in section 6.2, and analyze its convergence behavior for learning reward log-odds. In section 6.3 we present a linear approximation to the Bayesian Hebb rule that is much simpler to implement, but exhibits similar convergence behavior. In section 6.4 we show that after a suitable preprocessing of sensory variables $\mathbf{x}$ one arrives at a population code $\mathbf{y}$ for which optimal decisions can be represented by WTA applied to weighted sums of the variables y_i. The required weights can be learnt quite fast with the Bayesian Hebb rule, even if there exist conditional dependencies among the input variables $\mathbf{x}$. Section 6.5 gives experimental results on the performance of the Bayesian Hebb rule in various action selection tasks. Section 6.5.2 addresses the case of non-stationary reward distributions. In section 6.6 the learning rule is generalized to handle tasks in environments with continuous input signals $\mathbf{x}$. We discuss in section 7.4 salient aspects of the presented results, an application of the Bayesian Hebb rule to model the experimental data of (Yang & Shadlen, 2007), related work, and open problems.

6.2 The Bayesian Hebb rule

In this section we introduce a simple local learning rule, the reward-modulated Bayesian Hebb rule, which learns log-odds of reward probabilities conditioned on binary input variables. Analyzing the convergence behavior of the rule one sees that the true reward log-odds are fixed point attractors for expected weight changes under the reward-modulated Bayesian Hebb rule. The Bayesian Hebb rule also learns fast, since the online learned weights are close to what an optimal Bayesian learning approach, using (biologically unrealistic) counters and auxiliary variables, would achieve. It is further shown that an even simpler rule - which approximates the Bayesian Hebb rule - learns weights which are close to the optimum, and is sufficient for reliable decision making.

6.2.1 Action selection strategies and goals for learning

We consider the standard operant conditioning scenario, where the learner receives at each trial an input $\mathbf{x} = \langle x_1, \ldots, x_m \rangle$ (e.g. a sensory stimulus or internal state signals of the organism) with multinomial variables x_j, chooses an action a out of a set of l possible actions $A = \{a_1, \ldots, a_l\}$, and receives a reward $r \in \{0, 1\}$ with probability $p(r|\mathbf{x}, a)$. The learner's goal is to learn (as fast as possible) a policy $\pi(\mathbf{x}, a) = p(a|\mathbf{x})$ (or $\pi(\mathbf{x})$ in the case of a deterministic policy) so that action selection according to this policy maximizes the average reward. A structural difference to supervised prediction problems is that it does not suffice that the learner passively observes the outcomes of trials, since the reward received for action a in response to stimulus $\mathbf{x}$ provides no information about the probability of rewards

for alternative actions a' in response to the same stimulus $\mathbf{x}$. He therefore needs to try out different actions for the same input through an exploration process, in order to learn the reward-probabilities for all actions.

In this study the goal of the learner is fast learning of a policy that approximates the optimal policy. The learner does not necessarily maximize the online performance during learning, and does not specifically try to reduce uncertainty about the outcome of unexplored action. The strategies employed during learning are therefore not Bayes-optimal in the sense of decision theory and sequential analysis (Dayan & Daw, 2008). Optimal solutions to the exploration problem for a restricted subclass of tasks can be computed (Gittins, 1979; Lai & Robbins, 1985; Auer et al., 2002), but neural network implementations of these mechanisms are beyond the scope of this study. During learning we follow heuristic strategies that are commonly used in reinforcement learning (Sutton & Barto, 1998). The actions are chosen based on the currently learned weights, which approximate the Bayes optimal estimates for the reward log-odds. In order to maintain a rather high level of rewards during exploration, the agent might for example choose actions stochastically with $p(a|\mathbf{x}) = p(r{=}1|\mathbf{x}, a)$. This corresponds to the *matching behavior* phenomenon observed in biology, where the fraction of choices for one action exactly matches the fraction of total rewards from that action (Sugrue et al., 2004). This policy was used during training in all our computer experiments.

If the goal of the agent is to accumulate as many rewards as possible, and rewards are binary, the agent will choose the action with the highest probability $p(r = 1|\mathbf{x}, a)$ to yield reward. Since the function which maps a probability p onto $\log \frac{p}{1-p}$ is strictly monotonically increasing, the agent can choose instead the action a which has the highest log-odd

$$\log \frac{p(r = 1|\mathbf{x}, a)}{p(r = 0|\mathbf{x}, a)}. \tag{6.2}$$

Hence the optimal policy for maximizing the probability of reward can be written in the form

$$\pi(\mathbf{x}) = \arg\max_{a \in A} \ \log \frac{p(r = 1|\mathbf{x}, a)}{p(r = 0|\mathbf{x}, a)}. \tag{6.3}$$

We assume for now that the input $\mathbf{x} = \langle x_1, \ldots, x_m \rangle$ consists of m input variables which are arbitrary multinomial discrete random variables with unknown joint distribution (in section 6.6 we will consider the case of continuous inputs $\mathbf{x}$). We assume that these m variables are represented through binary states (firing / non-firing) $\mathbf{y} = \langle y_1, \ldots, y_n \rangle$ of n neurons in a population coding manner. We will define the encoding scheme later in section 6.4 and show that different encodings allow different representations of statistical dependencies. For every possible action a there exists in our simple model (see Figure 6.1) a linear neuron which receives as inputs the components $y_1, \ldots, y_n$ of $\mathbf{y}$. The activation L_a of this linear neuron is defined by the weighted sum

$$L_a = w_{a,0} + \sum_{i=1}^{n} w_{a,i} \, y_i. \tag{6.4}$$

Our approach aims at learning weights $w_{a,i}$ for every action a such that L_a corresponds to the reward log-odd (6.2), which indicates how desirable it is to execute action a in the current situation defined by $\mathbf{x}$ and its neural encoding $\mathbf{y}$. The action with the highest assumed probability of yielding reward is then selected by a Winner-Take-All (WTA) operation that is formally defined through the binary outputs $z_1, \ldots, z_l$ as follows:

$$z_a = \begin{cases} 1, & \text{if } L_a \geq L_b \text{ for } b \neq a \\ 0, & \text{else} \end{cases} . \tag{6.5}$$

This action selection strategy is commonly referred to as the *greedy* strategy.

If the goal is not only to exploit preceding experience in order to choose an action that maximizes the probability of reward for the current stimulus $\mathbf{x}$, but to simultaneously keep on learning and exploring reward probabilities for other actions, the previously mentioned matching behavior strategy (Sugrue et al., 2005) offers an attractive compromise. It can be implemented with the help of the learned parameters $w_{a,i}$ in the following way: The linear gate L_a in Figure 6.1 is replaced by a sigmoidal gate (i.e., the weighted sum L_a according to (6.4) is replaced by $\sigma(L_a) = \frac{1}{1+\exp(-L_a)}$, and the deterministic WTA gate is replaced by a stochastic soft-WTA gate (which selects a as winner with probability $\frac{\sigma(L_a)}{\sum_b \sigma(L_b)}$).

6.2.2 A local rule for learning reward log-odds

We will now present a learning rule and an appropriate input encoding for learning weights, which asymptotically approach target values such that the architecture in Figure 6.1 selects actions optimally. Consider first the case where for a single binary input y_i and action a the reward log-odd $\log \frac{p(r=1|y_i=1,a)}{p(r=0|y_i=1,a)}$ should be learned in the weight $w_{a,i}$. A traditional frequentist's approach would use counter variables

$$\begin{aligned} \alpha_{a,i} &= \#[r = 1 \wedge y_i = 1 \wedge \text{action } a \text{ selected}], \\ \beta_{a,i} &= \#[r = 0 \wedge y_i = 1 \wedge \text{action } a \text{ selected}] \end{aligned}$$

to estimate the reward log-odds $w^*_{a,i}$ after finitely many steps by

$$\hat{w}_{a,i} = \log \frac{\alpha_{a,i}}{\beta_{a,i}} \quad \text{for } i = 1, \ldots, n.$$

In a rewarded trial (i.e. $r = 1$) where $y_i = 1$ and action a is selected this leads to the update

$$\hat{w}^{new}_{a,i} = \log \frac{\alpha_{a,i} + 1}{\beta_{a,i}} = \log \frac{\alpha_{a,i}}{\beta_{a,i}} \left(1 + \frac{1}{\alpha_{a,i}}\right) = \hat{w}_{a,i} + \log(1 + \frac{1}{N_{a,i}}(1 + e^{-\hat{w}_{a,i}})) \ , \tag{6.6}$$

where $N_{a,i} := \alpha_{a,i} + \beta_{a,i}$ is the total number of previous updates, thus $\frac{1}{\alpha_{a,i}} = \frac{1}{N_{a,i}}(1 + \frac{\beta_{a,i}}{\alpha_{a,i}})$. Analogously, an update after a new unrewarded trial ($r = 0$) gives rise to the update

$$\hat{w}^{new}_{a,i} = \hat{w}_{a,i} - \log(1 + \frac{1}{N_{a,i}}(1 + e^{\hat{w}_{a,i}})). \tag{6.7}$$

Using the approximation $\log(1+x) \approx x$, and using a constant learning rate η instead of the factor $\frac{1}{N_{a,i}}$, the update rules (6.6) and (6.7) can be combined to yield a new local learning rule, which does not use any counters. [1] We call this rule the *reward-modulated Bayesian Hebb rule*. The update for weight $w_{a,i}$, whenever action a is selected and $y_i = 1$ is:

$$\boxed{\Delta w_{a,i} = \begin{cases} \eta \cdot (1 + e^{-w_{a,i}}), & \text{if } r = 1 \\ -\eta \cdot (1 + e^{w_{a,i}}), & \text{if } r = 0. \end{cases}} \tag{6.8}$$

This rule increases the weight whenever reward is encountered, and decreases the strength of the synapse otherwise. This learning rule (6.8) is purely local, i.e. it depends only on quantities that are available at the trained synapse, but not on the activity of other presynaptic neurons.

The approximation of the reward-modulated Bayesian Hebb rule to the exact counting model, which computes for every parameter the Bayes-optimal estimate that can be inferred from a fixed finite set of data, is illustrated in Figure 6.2A. In order to estimate a single parameter $q_{a,i} = p(r = 1|y_i = 1, a)$, a uniform prior on $[0, 1]$ was initially imposed on $q_{a,i}$. The counters $\alpha_{a,i}$ and $\beta_{a,i}$, as defined above, were incremented as training samples became available, and the posterior distribution for $q_{a,i}$ was given by the Beta$(\alpha_{a,i} + 1, \beta_{a,i} + 1)$ distribution (Neapolitan, 2004). The same samples were simultaneously used to update the weight $w_{a,i}$ by rule (6.8). The weights $w_{a,i}$, which represent log-odds $\log \frac{p(r=1|y_i=1,a)}{p(r=0|y_i=1,a)}$ were transformed into probabilities via the transformation

$$\hat{q}_{a,i} = \frac{1}{1 + \exp(-w_{a,i})} \quad .$$

Figure 6.2A shows the optimal posterior for a single $q_{a,i}$ after every update, and the approximation obtained by (6.8). The probability estimated by the Bayesian Hebb rule is always close to the Bayes-optimal estimate.

6.2.3 Convergence properties of the Bayesian Hebb rule in reinforcement learning

The Bayesian Hebb rule is an online learning rule which has no prior knowledge of its target values. However, one can prove that the weights learned with (6.8) converge (in expectation) to their optimal values $w^*_{a,i} = \log \frac{p(r=1|y_i=1,a)}{p(r=0|y_i=1,a)}$, just on the basis of the statistics of pre- and postsynaptic values they encounter. This is in fact very easy to prove, since the equilibrium of the rule is reached when the expected update $E[\Delta w_{a,i}]$ under the rule (6.8) vanishes, and this can be written as

$$\begin{aligned} E[\Delta w_{a,i}] = 0 \quad \Leftrightarrow \quad & p(r = 1|y_i = 1, a) \cdot \eta \cdot (1 + e^{-w_{a,i}}) - \\ & -p(r = 0|y_i = 1, a) \cdot \eta \cdot (1 + e^{w_{a,i}}) = 0 \quad . \end{aligned}$$

[1] Using the approximation $\log(1 + x) \approx x$ did not visibly affect the performance of the learning rule in the computer simulations in Section 6.5.

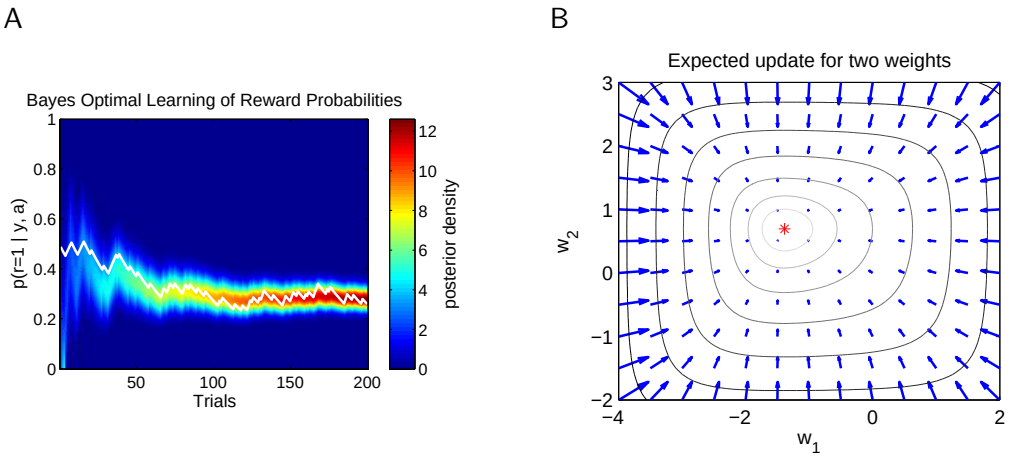


Figure 6.2: Convergence behavior of the Bayesian Hebb rule. **A)** The weights learned by the Bayesian Hebb rule approximate Bayes-optimal learning. The posterior for the reward probability $q_{a,i} = p(r = 1|y_i = 1, a)$ at every training trial was modeled by a Beta$(\alpha_{a,i} + 1, \beta_{a,i} + 1)$ distribution, with counters $\alpha_{a,i}$ and $\beta_{a,i}$ for rewarded and unrewarded trials. The color shows the estimated posterior density function for $q_{a,i}$ at every training trial. The white curve shows the approximation learned by the Bayesian Hebb rule (6.8) (with constant learning rate $\eta = 0.02$). The weight $w_{a,i}$ was transformed into an estimated reward probability by $\hat{q}_{a,i} = \frac{1}{1+\exp(-w_{a,i})}$. One can see that the approximation follows the optimal estimate closely. **B)** Attractor property of the Bayesian Hebb rule (6.8) plotted for two weights w_1 and w_2. The expected update (indicated by a blue arrow) is always in the direction of the optimal weights (marked by a red star). Gray curves connect points with the same amount of expected weight change.

As we show in the following section 6.2.3.1, the latter explicitly holds iff $w_{a,i}$ is at the target value $w^*_{a,i} = \log \frac{p(r=1|y_i=1,a)}{p(r=0|y_i=1,a)}$. If a vector of $n+1$ weights $\langle w_{a,0}, \ldots, w_{a,n} \rangle$ for an action a is learned simultaneously, the point $\langle w^*_{a,0}, \ldots, w^*_{a,n} \rangle$ is a global fixed point attractor in the weight space $\mathbb{R}^{n+1}$ with regard to expected weight changes under the Bayesian Hebb rule (see Figure 6.2B).

Another unusual feature of the Bayesian Hebb rule is that one can prove that it converges exponentially fast to $w^*_{a,i}$ (w.r.t. $E[\Delta w_{a,i}]$). In particular, weight updates move the weight in larger steps towards the attractor $w^*_{a,i}$ if they are farther off, without requiring any change of the learning rate, or knowledge of the ideal values $w^*_{a,i}$.

6.2.3.1 Convergence proofs for the Bayesian Hebb rule

We assume that $p(r|\mathbf{y}, a)$, the reward probability conditioned on the current input and action, is stationary, and $p(y_i = 1, a) > 0$ for all $a \in A$ and $i \in \{1, \ldots, n\}$. Apart

from the latter assumption, the equilibrium is independent of the exploration policy $\pi(\mathbf{x}, a)$. The constraint on $p(y_i = 1, a)$ means that all values of all input variables must have a non-zero probability in the input-distribution, and every action must have a non-zero probability of being tried out. If $p(y_i = 1, a) = 0$ for some y_i and a, then such trials are never encountered, and no meaningful weight $w_{a,i}$ can be learned.

Since updates of $w_{a,i}$ in (6.8) are only made when a is executed and $y_i = 1$, one can write

$$\begin{aligned} E[\Delta w_{a,i}] = 0 \quad &\Leftrightarrow \quad p(r = 1|y_i = 1, a) \cdot \eta \cdot (1 + e^{-w_{a,i}}) - \\ &\quad -p(r = 0|y_i = 1, a) \cdot \eta \cdot (1 + e^{w_{a,i}}) = 0 \\ &\Leftrightarrow \quad \frac{1 + e^{w_{a,i}}}{1 + e^{-w_{a,i}}} = \frac{p(r = 1|y_i = 1, a)}{p(r = 0|y_i = 1, a)} \\ &\Leftrightarrow \quad e^{w_{a,i}} = \frac{p(r = 1|y_i = 1, a)}{p(r = 0|y_i = 1, a)} \\ &\Leftrightarrow \quad w_{a,i} = \log \frac{p(r = 1|y_i = 1, a)}{p(r = 0|y_i = 1, a)}. \end{aligned}$$

The above is a chain of equivalence transformations, therefore $w^*_{a,i} = log\frac{p(r=1|y_i=1,a)}{p(r=0|y_i=1,a)}$ is the only equilibrium value of rule (6.8).

One can also show that the expected update of weights $w_{a,i}$ is always in the right direction:

$$\begin{aligned} &E[\Delta w_{a,i}]|_{w^*_{a,i}+2\varepsilon} = E[\Delta w_{a,i}]|_{w^*_{a,i}+2\varepsilon} - E[\Delta w_{a,i}]|_{w^*_{a,i}} \\ &\propto p(r{=}1|y_i = 1, a)e^{-w^*_{a,i}}(e^{-2\varepsilon} - 1) - p(r{=}0|y_i = 1, a)e^{w^*_{a,i}}(e^{2\varepsilon} - 1) \\ &= p(r{=}0|y_i = 1, a)(e^{-2\varepsilon} - 1) - p(r{=}1|y_i = 1, a)(e^{2\varepsilon} - 1) \\ &= \left[(p(r{=}0|y_i = 1, a)e^{-\varepsilon} + p(r{=}1|y_i = 1, a)e^{\varepsilon}\right] (e^{-\varepsilon} - e^{\varepsilon}). \end{aligned} \tag{6.9}$$

The first term in (6.9) is always positive, and from the last term in (6.9) one can see that whenever $w_{a,i} > w^*_{a,i}$, i.e. $\varepsilon > 0$, the expected change of $w_{a,i}$ is negative, and positive if $\varepsilon < 0$. The expected change of weights is therefore always in the direction of the optimal weight, and the initial weight values or pertubations of the weights decay exponentially fast. Furthermore, trajectories of weights that start at different initial values converge exponentially fast. Hence the resulting weight dynamics is contracting in the sense of Lohmiller and Slotine (1998).

6.3 The Linear Bayesian Hebb rule

The reward-modulated Bayesian Hebb rule (6.8) includes exponential terms $\exp(-w_{a,i})$ and $\exp(w_{a,i})$. One may argue that an exact calculation of the exponential function is beyond the capabilities of a synaptic learning process. Therefore we have also analyzed a linear approximation to the Bayesian Hebb rule. The

exponential function is defined by the Taylor series

$$\exp(x) = \sum_{i=0}^{\infty} \frac{x^i}{i!}. \tag{6.10}$$

Thus, the first order approximations for $\exp(w_{a,i})$ and $\exp(-w_{a,i})$ are

$$\exp(w) \approx 1 + w \tag{6.11}$$
$$\exp(-w) \approx 1 - w. \tag{6.12}$$

Inserting the approximations (6.11) and (6.12) into (6.8), a computationally simpler learning rule is obtained, which we call the *linear Bayesian Hebb* rule. Whenever action a is selected and $y_i = 1$, it updates weight $w_{a,i}$ by:

$$\boxed{\Delta w_{a,i} = \begin{cases} \eta \cdot (2 - w_{a,i}), & \text{if } r = 1 \\ -\eta \cdot (2 + w_{a,i}), & \text{if } r = 0. \end{cases}} \tag{6.13}$$

This new rule resembles strongly the typical Hebb rule with a regularization term. The weights are increased by a constant if the pre- and postsynaptic neurons "fire together" (i.e., $y_i = 1$ and action a is selected), and decreased by a constant if they don't. The $\pm w_{a,i}$ term prevents the weights from growing too large or too small. Actually, for $\eta \leq 1$ it always keeps the weights within the range $[-2, 2]$. This shows immediately that the linear Bayesian Hebb rule cannot learn the true reward log-odds for arbitrary distributions, but only an approximation. Figure 6.3A shows the updates by the linear Bayesian Hebb rule (dashed lines) in comparison to those of the exact rule (6.8) (solid lines). One can see that the difference between the updates grows for larger values of the target weight $w^*_{a,i}$. However, our computer experiments in Section 6.5 will demonstrate that the linear Bayesian Hebb rule performs remarkably well for many benchmark tasks.

6.3.1 Convergence of the Linear Bayesian Hebb Rule

We show in the following section 6.3.1.1 that the equilibrium value for the linear Bayesian Hebb rule (6.13), i.e. the weight value where $E[\Delta w_{a,i}] = 0$, is at

$$\begin{aligned} w^+_{a,i} &= -2 + 4 \cdot p(r = 1 | y_i = 1, a) \\ &= 2 \cdot (p(r = 1 | y_i = 1, a) - p(r = 0 | y_i = 1, a)) \quad . \end{aligned}$$

This equilibrium value is monotonically increasing with $w^*_{a,i}$, the equilibrium value of the exact Bayesian Hebb rule (6.8). They are only equal when $p(r = 1 | y_i = 1, a) = p(r = 0 | y_i = 1, a)$, i.e. $w^*_{a,i} = w^+_{a,i} = 0$.

In Figures 6.3B and C the evolution of two weights during learning for a random distribution is shown. In 6.3B, the target value is close to zero, where the target values for the exact rule (6.8) and the linear Bayesian Hebb rule (6.13) are very similar. Thus, no big difference in weight space is visible. In 6.3C, however,

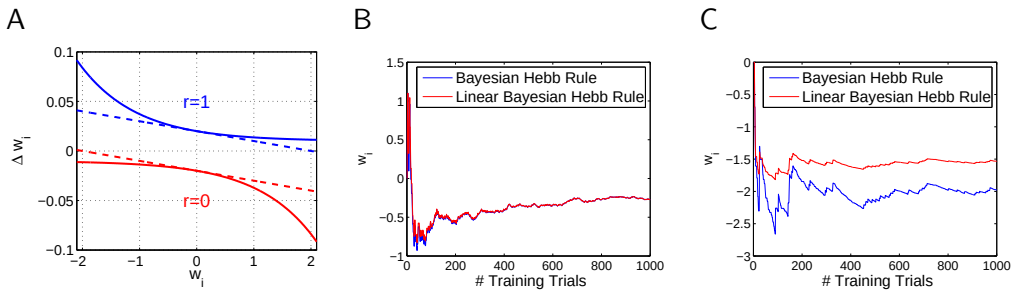


Figure 6.3: Linear approximation of the Bayesian Hebb rule. **A)** Update Δw_i of the Bayesian Hebb rule (6.8) (solid lines) and the linear Bayesian Hebb rule (6.13) (dashed lines) plotted as a function of the current weight value w_i for training trials with $r = 1$ (blue curves) and $r = 0$ (red curves). **B)** Example of the evolution of a single weight under the Bayesian Hebb rule (6.8) and the linear Bayesian Hebb rule (6.13). The target value is close to 0, where the approximation of the linear Bayesian Hebb rule is very good. **C)** Another example of the weight evolution, in which the two rules converge to different weights. The target weight is close to -2, which is the border of the weight-range that the linear Bayesian Hebb rule can cover. The approximation error is therefore large compared to B.

the target value is close to the maximum value that the linear rule can represent, therefore the two rules do not converge to the same value, indicating a larger approximation error for the linear rule. Hence the linear Bayesian Hebb rule can be expected to perform well if the target values of the weights have small absolute values.

6.3.1.1 Convergence proof for the Linear Bayesian Hebb rule

The expected update of the linear Bayesian Hebb rule (6.13) vanishes when

$$\begin{aligned}
E[\Delta w_{a,i}] = 0 \quad &\Leftrightarrow \quad p(r=1|y_i=1,a)\cdot\eta\cdot(2-w_{a,i}) - p(r=0|y_i=1,a)\cdot\eta\cdot(2+w_{a,i}) = 0 \\
&\Leftrightarrow \quad 2(p(r=1|y_i=1,a) - p(r=0|y_i=1,a)) = \\
&\qquad = w_{a,i}\cdot(p(r=1|y_i=1,a) + p(r=0|y_i=1,a)) = w_{a,i} \\
&\Leftrightarrow \quad w_{a,i} = 2(p(r=1|y_i=1,a) - 1 + p(r=1|y_i=1,a)) \\
&\Leftrightarrow \quad w_{a,i} = -2 + 4\cdot p(r=1|y_i=1,a) \quad .
\end{aligned}$$

We have used here that the reward is binary, and so

$$p(r=0|y_i=1,a) + p(r=1|y_i=1,a) = 1 \quad .$$

The above is a chain of equivalence transformations, so $w^+_{a,i} = -2 + 4\cdot p(r=1|y_i = 1,a)$ is the only equilibrium value of (6.13).

6.4 Population codes for Hebbian learning of asymptotically optimal decisions

In this section two preprocessing mechanisms are presented, which are based on different assumptions about statistical dependencies among input variables. Applied to these population encodings of the input, the WTA circuit in Figure 6.1 selects actions that maximize the probability of obtaining reward, according to the current statistical model represented by the input encoding and the reward log-odds learned with the Bayesian Hebb rule.

We have previously shown that the reward-modulated Bayesian Hebb rule (6.8) has a unique equilibrium at the reward log-odd

$$w^*_{a,i} = \log \frac{p(r=1|y_i=1,a)}{p(r=0|y_i=1,a)} \quad . \tag{6.14}$$

In order to approximate the true reward probabilities for every action as weighted sums as in (6.4), every vector of input variables $\mathbf{x} = \langle x_1, \ldots, x_m \rangle$ needs to be suitably preprocessed into a population code vector $\mathbf{y} = \langle y_1, \ldots, y_n \rangle$. If the weights $w_{a,i}$ for every y_i and every action a are learned with the Bayesian Hebb rule, our previous analysis guarantees that the resulting policy will asymptotically approach the best policy that can be inferred for the given preprocessing function.

Let the input variables $x_1, \ldots, x_m$ be some arbitrary multinomial random variables with unknown joint distribution, where each variable x_k assumes m_k different values $v^k_1, \ldots, v^k_{m_k}$. For the sake of simplicity we assume that $v^k_j = j$ for $j = 1, \ldots, m_k$ and $k = 1, \ldots, m$.

We first present a very simple population coding, which is sufficient to represent the optimal policy as a weighted sum if the Naive Bayes assumption holds for the input variables, i.e. the input variables x_k are conditionally independent of each other given the selected action a and the reward r:

$$p(x_k|r,a,x_1,\ldots,x_{k-1},x_{k+1},\ldots,x_m) = p(x_k|r,a) \text{ for all } k \in \{1,\ldots,m\}. \tag{6.15}$$

In this case it holds that

$$\frac{p(r=1|\mathbf{x},a)}{p(r=0|\mathbf{x},a)} = \frac{p(r=1|a)}{p(r=0|a)} \prod_{k=1}^{m} \frac{p(x_k|r=1,a)}{p(x_k|r=0,a)} \, . \tag{6.16}$$

Every x_k is discrete and can only take on finitely many different values. Each discrete conditional distribution $p(x_k|r,a)$ for a fixed action a and a fixed value of r is therefore fully described by m_k probability values, one for each possible value of x_k, and can be written in the form

$$p(x_k|r,a) = p(x_k=1|r,a)^{I(x_k=1)} \cdot p(x_k=2|r,a)^{I(x_k=2)} \cdot \ldots \cdot p(x_k=m_k|r,a)^{I(x_k=m_k)},$$

where the indicator function I is defined as $I(\texttt{true}) = 1$ and $I(\texttt{false}) = 0$. With this notation, and with an application of Bayes' theorem, (6.16) can be rewritten

as

$$\begin{aligned}\frac{p(r=1|\mathbf{x},a)}{p(r=0|\mathbf{x},a)} &= \frac{p(r=1|a)}{p(r=0|a)}\prod_{k=1}^{m}\prod_{j=1}^{m_k}\left(\frac{p(x_k=j|r=1,a)}{p(x_k=j|r=0,a)}\right)^{I(x_k=j)} \\ &= \frac{p(r=1|a)}{p(r=0|a)}\prod_{k=1}^{m}\prod_{j=1}^{m_k}\left(\frac{p(r=1|x_k=j,a)}{p(r=0|x_k=j,a)}\cdot\frac{p(r=0|a)}{p(r=1|a)}\right)^{I(x_k=j)} \\ &= \left(\frac{p(r=1|a)}{p(r=0|a)}\right)^{1-m}\prod_{k=1}^{m}\prod_{j=1}^{m_k}\left(\frac{p(r=1|x_k=j,a)}{p(r=0|x_k=j,a)}\right)^{I(x_k=j)} \\ &= \frac{p(r=1|a)}{p(r=0|a)}\prod_{k=1}^{m}\left(\frac{p(r=0|a)}{p(r=1|a)}\prod_{j=1}^{m_k}\left(\frac{p(r=1|x_k=j,a)}{p(r=0|x_k=j,a)}\right)^{I(x_k=j)}\right).\end{aligned} \tag{6.17}$$

This suggests to represent every x_k by a population code, which has m_k+1 binary variables, one for every possible value of x_k, and one bias variable to account for the term $\frac{p(r=0|a)}{p(r=1|a)}$. Formally we define the simple preprocessing (*SP*) $\boldsymbol{\phi}(x_k)$ for a single variable x_k as

$$\boldsymbol{\phi}(x_k) = [-1, \varphi_1, \ldots, \varphi_{m_k}]^T, \text{ where } \varphi_j = \begin{cases} 1, & \text{if } x_k = j \\ 0, & \text{otherwise.} \end{cases} \tag{6.18}$$

As an example we consider the simple reward distribution with 2 input variables $\mathbf{x} = \langle x_1, x_2\rangle$, modeled by the Bayesian network in Figure 6.4A. Under the Naive Bayes assumption the dependency of x_2 on the input variable x_1 is neglected, i.e. the arrow $x_1 \to x_2$ in the Bayesian network is ignored. For binary x_k, the population code under this assumption is illustrated in Figure 6.4C. Each input variable x_k is encoded separately by 3 variables y_i, where one is constantly -1, and only one other y_i is active, depending on the value of x_k.

The vectors $\boldsymbol{\phi}(x_k)$ for $k = 1, \ldots, m$ are concatenated into one population code vector $\mathbf{y}$ for the whole input. $\mathbf{y}$ has $n = 1 + m + \sum_{k=1}^{m} m_k$ entries, of which exactly $2 \cdot m + 1$ are non-zero, and the first entry $y_0 \equiv 1$ corresponds to the bias term $\frac{p(r=1|a)}{p(r=0|a)}$ in (6.17):

$$\mathbf{y} = \boldsymbol{\Phi}(\mathbf{x}) = \begin{bmatrix} 1 \\ \boldsymbol{\phi}(x_1) \\ \boldsymbol{\phi}(x_2) \\ \vdots \\ \boldsymbol{\phi}(x_m) \end{bmatrix}. \tag{6.19}$$

Substituting the definition of $\mathbf{y}$ from (6.18) and (6.19) into (6.17) and taking the logarithm then yields the log-odd function

$$\log\frac{p(r=1|\mathbf{y},a)}{p(r=0|\mathbf{y},a)} = \log\frac{p(r=1|a)}{p(r=0|a)} + \sum_{i=1}^{n} y_i \log\frac{p(r=1|y_i\neq 0,a)}{p(r=0|y_i\neq 0,a)}. \tag{6.20}$$

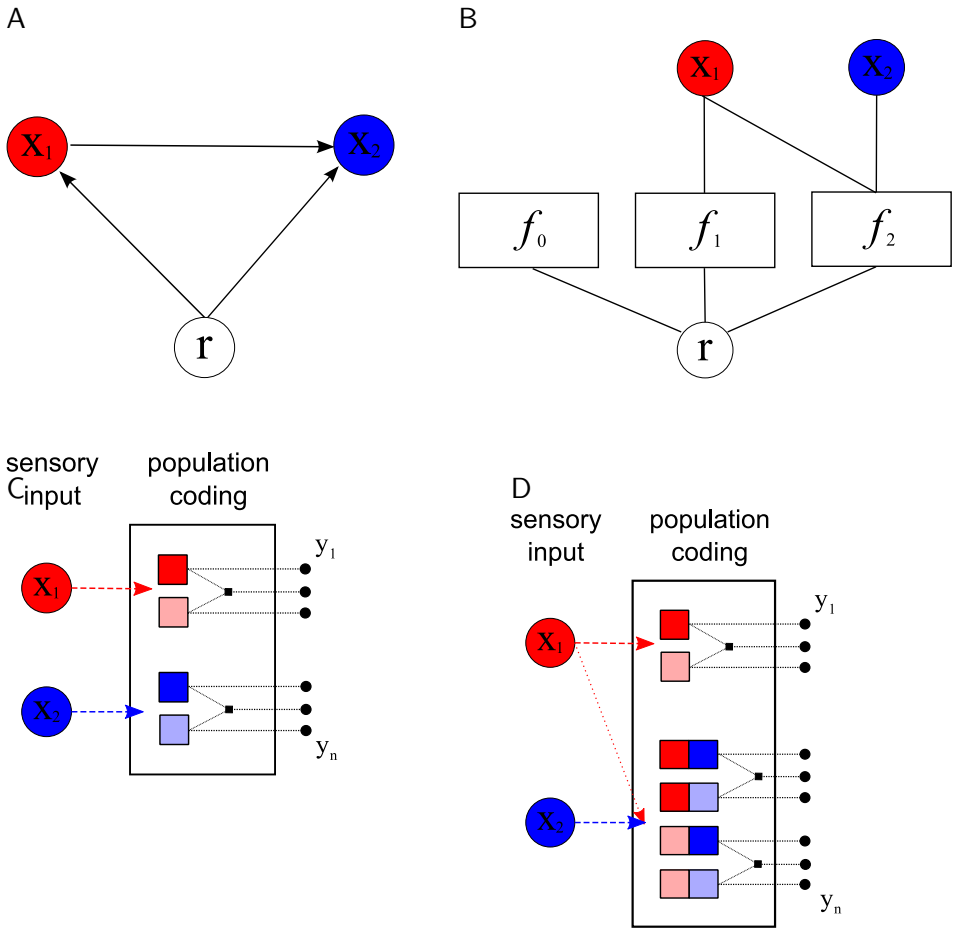


Figure 6.4: Preprocessing for tasks with arbitrary statistical dependencies. **A)** An example Bayesian network for the joint distribution of sensory inputs $\mathbf{x} = \langle x_1, x_2 \rangle$ and reward r. **B)** Factor graph representation for the prediction of r, according to the Bayesian network in panel A. Here, f_0 represents the prior $p(r)$, and the factors f_1 and f_2 represent the conditional probabilities $p(x_1|r)$ and $p(x_2|x_1, r)$, respectively. **C)** Population coding under the Naive Bayes assumption, which we refer to as simple preprocessing (SP). For every possible value of the variables x_k (here x_1, x_2 are binary), there is one variable y_i (indicated by a black circle) that outputs the value 1. Additionally there is one variable y_i for every x_k, which is constantly at -1 (black square). The constant bias term y_0 is not shown. **D)** Population coding applied to the factors in the factor graph shown in panel B. For each combination of values of the variables $\{x_k, \mathbf{x}_{\mathbf{P}_k}\}$ of a factor there is exactly one variable y_i (indicated by a black circle) associated with the factor that outputs the value 1. Other variables y_i represent OR's of these values (black squares), and yield either 0 or -1. The constant bias term y_0 is not shown. We refer to the resulting preprocessing circuit that maps sensory inputs $\mathbf{x}$ onto internal variables $\mathbf{y}$ that support Hebbian learning of optimal decisions as generalized preprocessing (GP).

If we use the population code (6.19) for $\mathbf{y}$, we can apply the reward-modulated Bayesian Hebb rule (6.8) for every y_i to learn reward log-odds conditioned on feature y_i being active[2]. For a y_i that is constantly active, such as y_0, the weight $w_{a,i}$ will converge to the prior reward probability $\log \frac{p(r=1|a)}{p(r=0|a)}$ for action a. Inserting the target values (6.14) of the weights into (6.20), we can therefore write

$$\log \frac{p(r=1|\mathbf{y},a)}{p(r=0|\mathbf{y},a)} = \sum_{i=0}^{n} w^*_{a,i}\, y_i\,. \tag{6.21}$$

During learning the current values of the weights $w_{a,0},\ldots,w_{a,n}$ are used to approximate the true reward log-odd for every action a as the weighted sums in (6.4). Actions are selected by a heuristic method according to their predicted probability of yielding reward (e.g. greedy or matching behavior). If the Naive Bayes assumption holds, the reward-modulated Bayesian Hebb rule in combination with a simple population coding for every input variable x_k is therefore sufficient to asymptotically learn the optimal action selection policy.

6.4.1 Learning decisions for arbitrary discrete distributions

We now address the more general case, where conditional independence of the input variables $x_1,\ldots,x_m$ cannot be assumed. We show that with a fixed preprocessing of the input that takes their dependencies into account, the Bayesian Hebb rule enables the resulting neural network to converge quite fast to the best performance that any action selection mechanism could possibly achieve. The dependency structure of the underlying input and reward distribution is given in terms of an arbitrary Bayesian Network BN for discrete variables (like e.g. Figure 6.4A). BN can be represented, like every Bayesian network, by a directed graph without directed cycles. We do not assume any further restrictions on the structure of the Bayesian network, so BN does not have to be a tree (as assumed in (Deneve, 2008b)), and it is not required to have no undirected cycles (as necessary for guaranteed convergence of belief propagation algorithms (Bishop, 2006)).

Without loss of generality we choose a numbering scheme such that the direct children of the reward node r in BN are $x_1,\ldots,x_{m'}$. The dependencies in BN can be described by $m+1$ parent sets $\mathtt{P}_k$, which are possibly empty, and explicitly exclude the reward node r. $\mathtt{P}_k$ is thus defined as

$$\mathtt{P}_k \;=\; \{i \mid \text{a directed edge } x_i \to x_k \text{ exists in BN} \;\text{ and }\; x_i \neq r\} \quad .$$

Additionally we define $\mathtt{P}_r$ as the set of all parents of the reward-node r. The joint probability distribution on the variables $r, x_1,\ldots,x_m$ in the Bayesian network for action a can then be factored, giving rise to a factor graph (Kschischang et al.,

[2]We consider a feature y_i active if it is non-zero, i.e. both $y_i = 1$ and $y_i = -1$ are *active* features.

2001) as indicated in Figure 6.4B:

$$p(r, \mathbf{x}|a) = p(r|\mathbf{x}_{\mathrm{P}_r}, a) \prod_{k=1}^{m'} p(x_k|\mathbf{x}_{\mathrm{P}_k}, r, a) \prod_{k=m'+1}^{m} p(x_k|\mathbf{x}_{\mathrm{P}_k}, a). \tag{6.22}$$

When calculating the log-odd of obtaining reward or not, the last terms in (6.22) cancel out, and a simple application of Bayes' theorem leads to

$$\begin{aligned}\log \frac{p(r=1|\mathbf{x}, a)}{p(r=0|\mathbf{x}, a)} = &\log \frac{p(r=1|\mathbf{x}_{\mathrm{P}_r}, a)}{p(r=0|\mathbf{x}_{\mathrm{P}_r}, a)} + \\ &+ \sum_{k=1}^{m'} \left(\log \frac{p(r=1|x_k, \mathbf{x}_{\mathrm{P}_k}, a)}{p(r=0|x_k, \mathbf{x}_{\mathrm{P}_k}, a)} - \log \frac{p(r=1|\mathbf{x}_{\mathrm{P}_k}, a)}{p(r=0|\mathbf{x}_{\mathrm{P}_k}, a)} \right). \end{aligned} \tag{6.23}$$

This is a sum of conditional reward log-odds, which can all be learned with the reward-modulated Bayesian Hebb rule. We now develop a suitable sparse encoding of $x_1, \ldots, x_m$ into binary variables $y_1, \ldots, y_n$ (with $n \gg m$), such that the reward log-odd can be written as a weighted sum

$$\log \frac{p(r=1|\mathbf{y}, a)}{p(r=0|\mathbf{y}, a)} = \sum_{i=1}^{n} w_{a,i}\, y_i,$$

and the weights $w_{a,i}$ correspond to conditional reward log-odds of y_i's. For the example Bayesian network in Figure 6.4A, the corresponding sparse code is illustrated in Figure 6.4D: One binary variable is created for every possible value assignment to a variable x_k and all its parents $\mathbf{x}_{\mathrm{P}_k}$, and additional binary variables are created for every possible value assignments to the parent nodes only. One should contrast this with the simple population code in Figure 6.4C, which assumes that the Naive Bayes condition holds, and therefore ignores that x_2 is dependent on x_1.

BN can also be viewed as a factor graph (see Figure 6.4B), in which there is for every variable x_k a factor f_k, which is connected to r, x_k and $\mathbf{x}_{\mathrm{P}_k}$, the parents of x_k in BN. The preprocessing is then computed separately for every factor f_k. We define the fixed *generalized preprocessing* (*GP*) operation for f_k with $k \geq 1$ as

$$\boldsymbol{\Phi}(x_k, \mathbf{x}_{\mathrm{P}_k}) = \begin{bmatrix} \boldsymbol{\phi}(x_k, \mathbf{x}_{\mathrm{P}_k}) \\ -\boldsymbol{\phi}(\mathbf{x}_{\mathrm{P}_k}) \end{bmatrix}. \tag{6.24}$$

The summands of the sum on the r.h.s. of (6.23) are split into two parts, and $\boldsymbol{\phi}(x_k, \mathbf{x}_{\mathrm{P}_k})$ defines the preprocessing for the first part, whereas $-\boldsymbol{\phi}(\mathbf{x}_{\mathrm{P}_k})$ defines the preprocessing for the latter part. The variables $\langle x_k, \mathbf{x}_{\mathrm{P}_k} \rangle$ are viewed as a single multinomial variable, and $\boldsymbol{\phi}(x_k, \mathbf{x}_{\mathrm{P}_k})$ is a representation of this multinomial variable through simple population coding. Thus, $\boldsymbol{\phi}(x_k, \mathbf{x}_{\mathrm{P}_k})$ has as many binary output variables $y_{k,i}$ as there are different assignments of values to all variables in $\langle x_k, \mathbf{x}_{\mathrm{P}_k} \rangle$, and exactly one variable $y_{k,i}$ has value 1 for each such assignment. Let $y_{k,i}$ be the

binary output variable that corresponds to some assignment $x_k = j,\ \mathbf{x}_{\mathtt{P}_k} = \mathbf{u}$, then the corresponding weight $w_{a,k,i}$ for action a can be learnt through the same reward-modulated Bayesian Hebb rule (6.8) as in the Naive Bayes case. The target value, to which $w_{a,k,i}$ will converge is then

$$w^*_{a,k,i} = \log \frac{p(r=1|y_{k,i}=1,a)}{p(r=0|y_{k,i}=1,a)} = \log \frac{p(r=1|x_k=j,\mathbf{x}_{\mathtt{P}_k}=\mathbf{u},a)}{p(r=0|x_k=j,\mathbf{x}_{\mathtt{P}_k}=\mathbf{u},a)} \quad . \tag{6.25}$$

Analogously, the application of the reward-modulated Bayesian Hebb rule (6.8) for every component $y_{\mathtt{P}_k,i}$ of $-\boldsymbol{\phi}(\mathbf{x}_{\mathtt{P}_k})$ leads to the target weights

$$w^*_{a,\mathtt{P}_k,i} = \log \frac{p(r=1|y_{\mathtt{P}_k,i}=-1,a)}{p(r=0|y_{\mathtt{P}_k,i}=-1,a)} = \log \frac{p(r=1|\mathbf{x}_{\mathtt{P}_k}=\mathbf{u},a)}{p(r=0|\mathbf{x}_{\mathtt{P}_k}=\mathbf{u},a)} \quad , \tag{6.26}$$

with the only formal modification to the update rule (6.8) being that updates are not only made when $y_i = 1$, but also when $y_i = -1$, which obviously does not change the behavior of the learning process. Formally, all preprocessed vectors $\boldsymbol{\Phi}(x_k, \mathbf{x}_{\mathtt{P}_k})$ are concatenated into one vector $\mathbf{y}$ with $n = \sum_{k=1}^{m'} N_k + N_{\mathtt{P}_k}$ entries

$$\mathbf{y} = \begin{bmatrix} \boldsymbol{\Phi}(\mathbf{x}_{\mathtt{P}_r}) \\ \boldsymbol{\Phi}(x_1, \mathbf{x}_{\mathtt{P}_1}) \\ \vdots \\ \boldsymbol{\Phi}(x_{m'}, \mathbf{x}_{\mathtt{P}_{m'}}) \end{bmatrix} \quad .$$

This sparse, redundant input encoding provides a weighted sum representation of the reward log-odd

$$\log \frac{p(r=1|\mathbf{y},a)}{p(r=0|\mathbf{y},a)} = \sum_{i=1}^{n} w_{a,i}\ y_i,$$

where the weights $w_{a,1}, \ldots, w_{a,n}$ can all be learnt through the reward-modulated Bayesian Hebb rule (6.8) as described above.

6.5 Results of Computer Simulations

We now evaluate the performance of the reward-modulated Bayesian Hebb rule and its linear approximation and compare it to the standard learning model for simple conditioning tasks, the non-Hebbian Rescorla-Wagner rule (Rescorla & Wagner, 1972).

The reward-modulated Bayesian Hebb rule (6.8) was tested on a variety of action selection tasks with 4 possible actions. A Bayesian network with dependency structure as in Figure 6.4A was used to model the distribution $p(r, x_1, x_2|a)$ for every action a, where r is the binary reward signal, and x_1, x_2 are the two binary input signals. We assigned a constant reward prior $p(r|a) = 0.25$ to every action a, and randomly generated the conditional probability tables for $p(x_1|r,a)$ and $p(x_2|x_1,r,a)$: for every action a, every x_k ($k \in \{1,2\}$), and every possible value

assignment to the parent nodes $\langle \mathbf{x}_{\mathbf{P}_k}, r \rangle$, a random sample $q \in [0,1]$ was drawn from a Beta-distribution, and $p(x_k = 1 | \mathbf{x}_{\mathbf{P}_k}, r, a)$ was set to q.

The Bayesian networks which model the reward distribution were also used to create the samples of input vectors $\mathbf{x} = \langle x_1, x_2 \rangle$ for every training trial. First, one of the four Bayesian networks was chosen randomly with equal probability, so the distribution of input or test samples does not depend on the action selection during learning. Inputs $\mathbf{x}$ were drawn as random samples from the selected network. The agent then received the input $\mathbf{x}$ and chose its action a. The binary reward signal r was sampled from the distribution $p(r|\mathbf{x}, a)$, and thus depends on the chosen action. The agent used the tuple $\langle \mathbf{x}, a, r \rangle$ to update its weights $w_{a,i}$. Training consisted of 2000 trials, in which the matching behavior strategy (see section 6.2.1) was used for action selection during learning. The evaluation of the performance of the resulting policy after every trial used the greedy strategy (6.3), choosing actions on 500 independent test trials and measuring the average reward. The experiment was averaged over 250 different tasks with different reward distributions.

The preprocessed binary vectors $\mathbf{y} = \mathbf{\Phi}(\mathbf{x}) \in \{0,1\}^n$ were created either by simple population coding (see (6.19) and Figure 6.4C), which is suitable for the Naive Bayes case (6.15), or generalized preprocessing (see (6.24) and Figure 6.4D). The former mechanism is referred to as *Bayesian Hebb SP* in Figure 6.5 and the remainder of this chapter, whereas the generalized preprocessing mechanism is referred to as *Bayesian Hebb GP*. The Bayesian Hebb rule with these two kinds of preprocessing mechanisms was compared to the non-Hebbian Rescorla-Wagner rule (Rescorla & Wagner, 1972). This rule predicts the value of a (multi-dimensional) stimulus as a linear sum,

$$V(\mathbf{y}) = w_0 + \sum_{i=1}^{n} w_i y_i \quad ,$$

and minimizes the prediction error with a delta learning rule

$$\Delta w_i = \eta y_i \left(r - w_0 - \sum_{i=1}^{n} w_i y_i \right) \quad . \tag{6.27}$$

It can be seen from equation (6.27), that for the update of a single weight, the complete prediction of value for the current state, which depends on all weights, is needed. In the experiments the Rescorla-Wagner rule was used to learn weights for every action separately. The classical Rescorla-Wagner rule (6.27) , which we use for comparison, is directly applied to the inputs $\mathbf{x}$. We show in section 6.5.4 that the performance and learning speed of Rescorla-Wagner can also be improved if it is applied to the preprocessed vectors $\mathbf{y} = \mathbf{\Phi}(\mathbf{x})$, using the same *SP* and *GP* preprocessing mechanisms as for the Bayesian Hebb rule.

In addition, the reward-modulated Bayesian Hebb rule was also compared to a Bayes-optimal weight learning rule. In this case the conditional probabilities in the Bayesian network in Figure 6.4A were estimated using counter variables (see

section 6.2.2), and exact inference was used to compute reward probabilities for every action.

Figure 6.5 shows that the reward-modulated Bayesian Hebb rule for both types of preprocessing learns faster than the non-Hebbian Rescorla-Wagner rule and converges to better policies. If generalized preprocessing is used, the learned policy after approximately 200 trials is almost indistinguishable from the policy of an optimal learner, and after approximately 1000 trials the performance is very close to the optimal performance level.

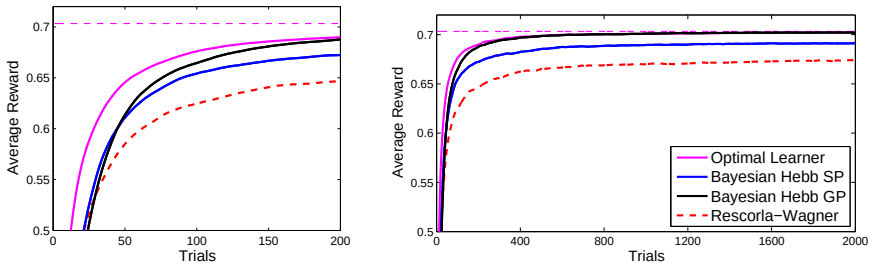


Figure 6.5: Performance of the reward-modulated Bayesian Hebb rule for action selection in a 4-action task with stochastic rewards. Each learner was trained on 2000 trials, and after every trial the performance was measured as the average reward of the greedy policy of each learner on 500 independent test trials (left: performance during the first 200 training trials). The results were averaged over 250 different problems, all having the statistical dependency structures as in Figure 6.4A, but random reward distributions (average learning and preprocessing time per problem on a dual-core 2.66 GHz, 16GB RAM PC: 0.9 s for SP, and 4.1 s for GP). The horizontal dashed line reflects the best possible performance of an optimal policy. The Bayesian Hebb rule with simple population coding (*Bayesian Hebb SP*) and generalized preprocessing (*Bayesian Hebb GP*) were compared to action-learning with the non-Hebbian Rescorla-Wagner rule. The learning rate was set to $1/N_{a,i}$, and stochastic action selection was used for exploration during training. The Bayesian Hebb rule for both preprocessing methods learned faster than the non-Hebbian Rescorla-Wagner rule and converged to better policies. With generalized preprocessing, the Bayesian Hebb rule converged to the optimal action-selection policy, as predicted by the theoretical analysis. Error bars are in the range of 10^{-3} and are omitted for clarity.

6.5.1 Approximations to the Bayesian Hebb rule

We have shown in section 6.3 that the linear Bayesian Hebb rule (6.13) can be derived as a first-order Taylor approximation of the reward-modulated Bayesian Hebb rule (6.8). There are no theoretical guarantees that the linear Bayesian Hebb rule will asymptotically converge towards weight values that allow optimal decision making. We compared the two rules on the same random Bayesian network tasks

for action selection empirically, using both the simple preprocessing (*SP*) for the Naive Bayes case, and the generalized preprocessing (*GP*) for arbitrary reward distributions. Figure 6.6 shows that this even simpler rule found good policies as quick as the exact rule. The quality of the final policy was almost indistinguishable from the policies found by the exact Bayesian Hebb rule.

6.5.2 Adaptation to changing reward distributions

In most realistic scenarios an organism experiences during its lifetime changes in the environment in which it lives. It is therefore important that a learning rule can adapt quickly to a changing reward or input distribution. It is clear that a learning rate that decays with $\frac{1}{N_i}$ (where N_i is the number of updates for a weight w_i) is not suitable for changing environments. We therefore used for this task the variance tracking mechanism for learning rate adaptation, which was first introduced by Nessler et al. (2009). This mechanism keeps track of the variance of each weight, and adapts learning rates accordingly. Learning rates are reduced for weights with small fluctuations, whereas they are increased for weights with high variance, which is an indication that those weights have not yet settled at their equilibrium values.

The learning rate adaptation mechanism uses two auxiliary variables, which can be locally estimated for every weight w_i: a running average of the weight is computed in $\bar{w}_i$, and a running average of the squared weight in $\bar{q}_i$, using the following simple update rules:

$$\begin{array}{lcll} \bar{w}_i^{new} & \leftarrow & (1-\eta_i)\,\bar{w}_i + \eta_i\, w_i & \\ \bar{q}_i^{new} & \leftarrow & (1-\eta_i)\,\bar{q}_i + \eta_i\, w_i^2 & . \end{array} \tag{6.28}$$

With these values the short-time variance of each weight can be estimated as $\bar{q}_i - \bar{w}_i^2$. Assuming that samples are drawn from stationary input distributions, it was shown in (Nessler et al., 2009) that the variance of a weight w_i can be related to the sample size N_i in the Bayes-optimal learning case (see also section 6.2.2), where exact counters for all combinations of inputs, actions and rewards are used, and conditional reward probabilities are modeled with Beta-distributions. According to this analysis, the new learning rate η_i^{new} can be set as

$$\eta_i^{new} \quad \leftarrow \quad \frac{\bar{q}_i - \bar{w}_i^2}{1+\cosh \bar{w}_i} \quad . \tag{6.29}$$

In practice this mechanism decays like $\frac{1}{N_i}$ under stationary conditions. It can also handle changing input distributions, because a new target value for w_i leads to larger updates Δw_i, thus increasing the short-time variance of the weight, and by (6.29) the learning rate η_i. Further details, and the theory behind this mechanism are described in (Nessler et al., 2009).

The variance tracking mechanism is an analytically justified rule for setting learning rates. Biological implementations of qualitatively similar processes are plausible, since all auxiliary quantities can be observed locally at the synapse. What is required is essentially a process that locally modulates potentiation or

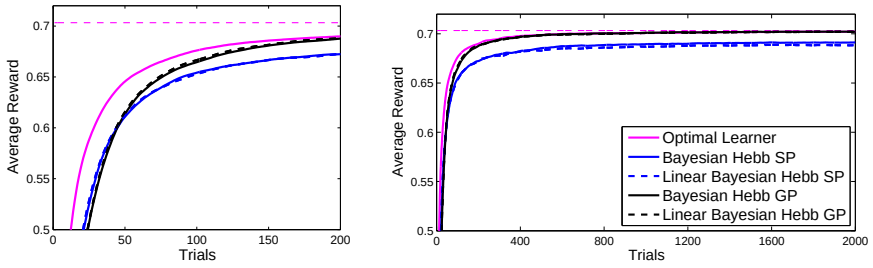


Figure 6.6: Performance of the linear approximations to the reward-modulated Bayesian Hebb rule in the same 4-action tasks as in Figure 6.5 (left: performance during the first 200 training trials). Both for simple population coding (*SP*) and generalized preprocessing (*GP*), the linear approximation to the learning rule learned as well as the exact rule. Error bars are in the range of 10^{-3} and are omitted for clarity.

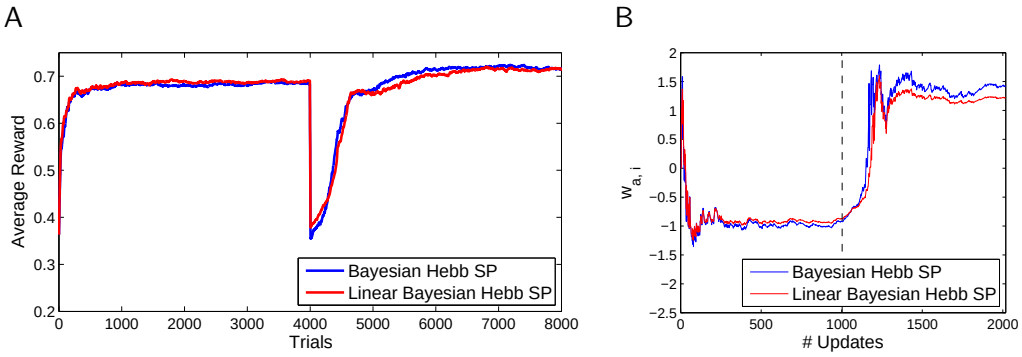


Figure 6.7: Behavior of the Bayesian Hebb rule when the reward distribution changes during training. **A)** Performance of the agent if a new reward distribution is introduced after 4000 training trials. There is an immediate drop when the distribution changes, but good performance is recovered quickly by both rules. **B)** Evolution of a single weight $w_{a,i}$ when the reward distribution changes. The weights are plotted at every trial where action a is selected, and an update for the plotted weight occurs. The weight first settles at the desired value for the first distribution, and then quickly adapts to the new target value when the distribution changes (indicated by the black dashed line).

depression of synapses, and itself is dependent on the magnitude of recent local synaptic weight changes. This could in principle be achieved by a large variety of metaplasticity mechanisms that are known to modulate synaptic plasticity (see (Abraham, 2008) for a recent review). Neuromodulators such as acetylcholine and norepinephrine could play a special role in the control of learning rates and the reduction of oscillations of weight updates (Doya, 2002; Yu & Dayan, 2003).

In the experiment shown in Figure 6.7, the weights were learned in 4000 training trials, after which the environment was changed and the learner was trained for another 4000 trials on the new input and reward distributions. Figure 6.7A shows that the performance of the learners initially improved, then dropped as soon as the distributions were switched, but quickly adapted to the new distribution, reaching almost the same performance. Figure 6.7B shows the evolution of a single weight in this scenario, for all trials in which it was updated. It can be seen that the weight first settled around the equilibrium value of the first distribution, and grew to reach the new target value after the switch.

6.5.3 Simulations for large input and action spaces

The Bayesian Hebb rule also works well for significantly larger problems. The same algorithms as in the previous sections were applied to problems with 100 binary input attributes, and 10 possible actions. The structures of the Bayesian networks that define the reward distributions for every action were generated randomly, using the algorithm described in (Ide & Cozman, 2002). Every node in the network could have a maximum of 5 parent nodes. The protocol for the generation of training samples and rewards was the same as for the previous experiments (see beginning of section 6.5). During learning actions were selected randomly, and the greedy policy was used for the evaluation on 1000 independent test trials (once every 1000 training trials).

Figure 6.8 shows that the Bayesian Hebb rule learns fast, both for simple population coding (SP), and generalized preprocessing (GP). The latter initially performs worse than SP, because the number of weights to learn is very large (about 1000 weights for every action), and approximation errors sum up. Given more training data, the Bayesian Hebb rule with generalized preprocessing approaches the performance of an optimal learner. The linear approximations to the reward-modulated Bayesian Hebb rule perform equally well on this task for both types of preprocessing.

6.5.4 Performance of the Rescorla-Wagner rule with preprocessing

The performance of the Rescorla-Wagner rule (6.27) can be improved by preprocessing input signals before the learning rule is applied. Figure 6.9 shows the average reward for the two tasks studied in Figure 6.5 (with 2 binary inputs and 4 actions), and Figure 6.8 (with 100 binary inputs and 10 actions). When the Rescorla-Wagner rule (6.27) was applied to simple population coding (SP) or to generalized preprocessing (GP), it learned faster and converged to better policies,

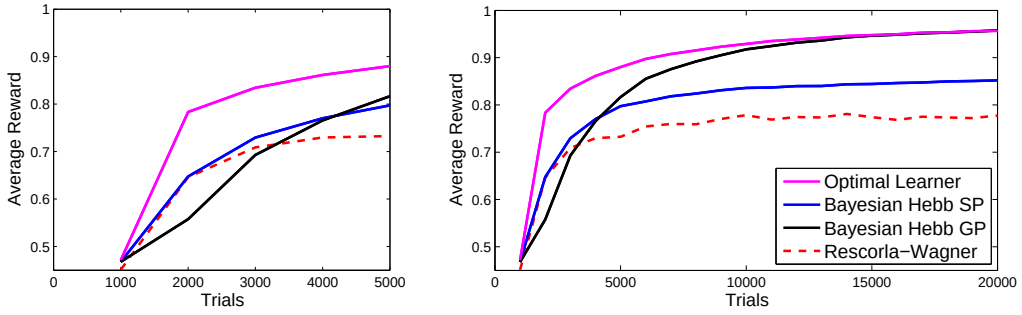


Figure 6.8: The Bayesian Hebb rule works well also for simulations with large input and action spaces. Each learner was trained on 20,000 trials of action selection problems with 10 actions, 100 binary input attributes, and stochastic rewards. Every 1000 trials the performance was measured as the average reward of the greedy policy of each learner on 1000 independent test trials (left: performance during the first 5000 training trials). The results were averaged over 40 different problems with random statistical dependency structures and random reward distributions (average learning and preprocessing time per problem on a 2-core 2.66 GHz, 16GB RAM PC: 27.8 s for SP, and 301.6 s for GP). The learning rates were set to $1/N_{a,i}$, and random action selection was used for exploration during training. With generalized preprocessing, the Bayesian Hebb rule approached the performance of an optimal learning mechanism. Error bars are in the range of 10^{-2} and are omitted for clarity.

although the performance of the Bayesian Hebb rule was mostly superior. These results suggest that the preprocessing methods presented in section 6.4, could also be beneficial for other learning mechanisms. The Augmented Rescorla-Wagner rule (Yuille, 2006) uses a preprocessing mechanism similar to GP, but it did not perform better for the experiments in this study.

6.6 Decision making with continuous inputs

The Bayesian Hebb rule can be generalized to action-selection problems defined on continuous input distributions. A rule very similar to (6.8) learns reward log-odds on a continuous input encoding, comparable to population codes with bell-shaped tuning curves that are observed in the brain.

The Bayesian Hebb rule has previously been defined only for discrete inputs x_k, which were mapped to binary variables y_i with various ways of preprocessing. We now present a learning rule to approximate distributions of a binary reward variable for continuous inputs. The preprocessing for this case is a population code, which uses radial-basis functions (*RBFs*) [3] to map continuous input variables x_k to new continuous features y_i, which may e.g. correspond to firing rates in a neural population code. Population codes with RBF- or bell-shaped tuning curves have

[3] Other mappings are also possible, but are not presented here.

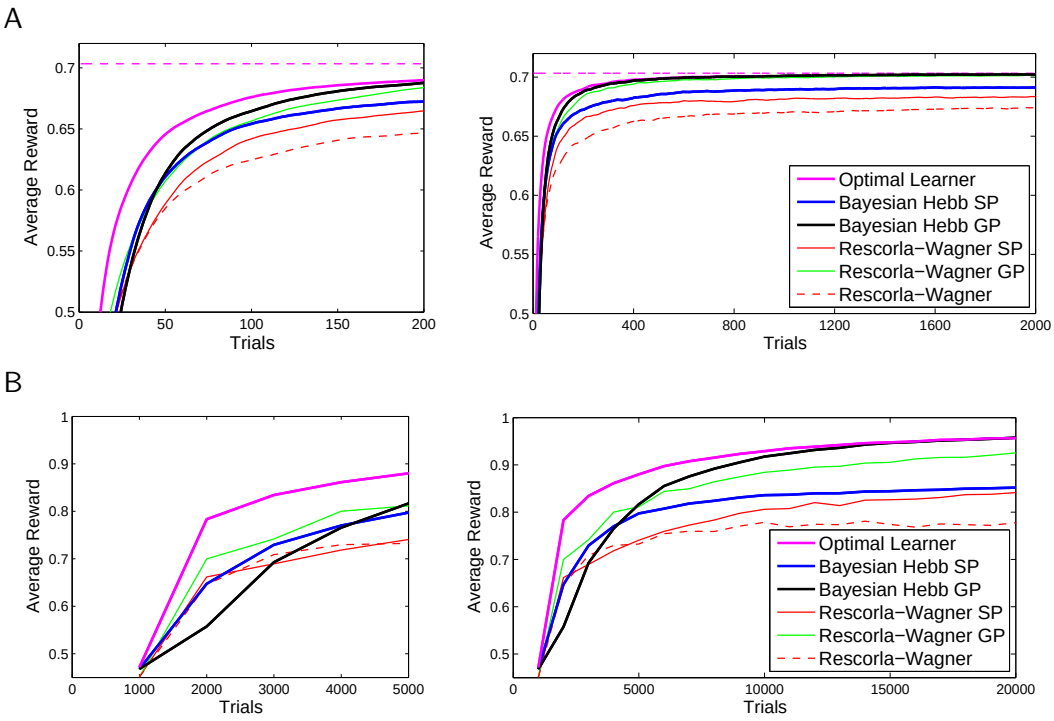


Figure 6.9: The performance of the Rescorla-Wagner rule can be improved by preprocessing input signals. The Rescorla-Wagner rule was applied to preprocessed inputs using simple population coding (*Rescorla-Wagner SP*), or generalized preprocessing (*Rescorla-Wagner GP*). The Rescorla-Wagner rule with preprocessing generally learned faster, and converged to better policies than the classical Rescorla-Wagner rule. **A)** Performance for the same 4-action tasks with 2 binary input variables as in Figure 6.5. **B)** Performance in the same 10-action tasks with 100 binary input variables as in Figure 6.8.

been observed, for example, in area MT of the visual system for direction sensitive cells (see (Pouget & Latham, 2002) for a review), place cells in rat hippocampus (O'Keefe et al., 1998), or for the encoding of movement directions in primate motor cortex (Georgopoulos et al., 1986). Networks of RBF units are also commonly used for models of visual object recognition (Riesenhuber & Poggio, 1999).

Consider the input variables $\mathbf{x} = \langle x_1, \ldots, x_m \rangle \in X \subseteq \mathbb{R}^m$, and a binary reward variable $r \in \{0, 1\}$. The continuous input $\mathbf{x}$ is mapped to a new set of n continuous non-negative features y_i. The activation of feature y_i is proportional to the activation of a RBF-kernel $\phi_i(\mathbf{x})$:

$$\phi_i(\mathbf{x}) = \exp\left(-\sum_{k=1}^{m} \frac{|x_k - c_{i,k}|^2}{s_{i,k}^2}\right). \tag{6.30}$$

The centers of the RBF kernels are located at $\mathbf{c}_i = \langle c_{i,1}, \ldots, c_{i,m} \rangle$, and the widths

of the kernels are given by $s_{i,k}$ (different widths may be used for different input dimensions). The preprocessed vector $\mathbf{y} = \langle y_1, \ldots, y_n \rangle$ is obtained by calculating the activations of all n different RBFs and normalizing the vector:

$$y_i(\mathbf{x}) = \frac{\phi_i(\mathbf{x})}{\sum_{j=1}^{n} \phi_j(\mathbf{x})}. \tag{6.31}$$

Notice that this kind of preprocessing can take combinations of variables into account, such as RBF kernels on $\mathbb{R}^m$, not only single variables. Figure 6.10 illustrates a simple continuous population code for 5 RBF kernels in one input dimension.

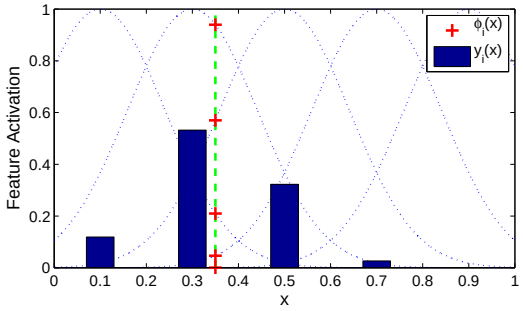


Figure 6.10: Example of a continuous population code with 5 equally spaced RBF kernels (width $s = 0.2$) for a 1-dimensional input x. The activations of the RBF-kernels $\phi_i(x)$ depend on the distance between x and the center c_i of the kernel. The normalized features $y_i(x)$ are obtained by dividing every $\phi_i(x)$ by the total sum of activations. The RBF-kernel activations $\phi_i(x)$ (red crosses mark the intersection of the vertical line at $x = 0.35$ with the 5 RBF-kernels indicated by blue dotted lines), and the normalized feature activations $y_i(x)$ (blue bars) are here shown for an example input at $x = 0.35$ (green dashed line).

A rule for learning reward log-odds conditioned on a single feature $y_i = y_i(\mathbf{x})$ can be defined by generalizing the reward-modulated Bayesian Hebb rule (6.8). Whenever action a is selected, every weight $w_{a,i}$ is updated by:

$$\boxed{\Delta w_{a,i} = \begin{cases} \eta \cdot y_i(\mathbf{x}) \cdot (1 + e^{-w_{a,i}}), & \text{if } r = 1 \\ -\eta \cdot y_i(\mathbf{x}) \cdot (1 + e^{w_{a,i}}), & \text{if } r = 0 \end{cases}.} \tag{6.32}$$

This rule is a generalization of rule (6.8), in which the updates are weighted by the activation of feature y_i. For the previously described discrete population codes, where y_i is either 0 or 1, the rule (6.32) is equivalent to (6.8).

For the analysis of the equilibrium of rule (6.32), we use an alternative population code of virtual binary features $\tilde{y}_1, \ldots, \tilde{y}_n$. We interpret $y_1(\mathbf{x}), \ldots, y_n(\mathbf{x})$ as (non-normalized) probabilities for randomly selecting one $i \in \{1, \ldots, n\}$, for which one sets $\tilde{y}_i = 1$ (while setting $\tilde{y}_j = 0$ for $j \neq i$). This gives a new interpretation

to the continuous population code features $y_i(\mathbf{x})$, because they are proportional to the probability that $\tilde{y}_i = 1$ (we then say that "feature $\tilde{y}_i$ is active").

To find the equilibrium of the rule (6.32) for the weight $w_{a,i}$, we set the expected update $E[\Delta w_{a,i}]$ to zero, and rewrite it as

$$\begin{aligned} E[\Delta w_{a,i}] = 0 \quad \Leftrightarrow \quad & (1 + e^{-w_{a,i}}) \int_X y_i(\mathbf{x}) \; p(r = 1, \mathbf{x}|a) \; d\mathbf{x} \\ & -(1 + e^{w_{a,i}}) \int_X y_i(\mathbf{x}) \; p(r = 0, \mathbf{x}|a) \; d\mathbf{x} = 0 \quad . \end{aligned}$$

We now show that this condition is fulfilled if and only if $w_{a,i}$ is at the target value

$$w^*_{a,i} = \log \frac{p(r = 1|\tilde{y}_i = 1, a)}{p(r = 0|\tilde{y}_i = 1, a)} \quad ,$$

using the interpretation of $y_i(\mathbf{x})$ as $p(\tilde{y}_i = 1|\mathbf{x})$. Since the virtual population code feature $\tilde{y}_i$ depends only on $\mathbf{x}$, but not on r, one can assume that r and $\tilde{y}_i$ are conditionally independent given $\mathbf{x}$, i.e.

$$p(r, \tilde{y}_i|\mathbf{x}, a) = p(r|\mathbf{x}, a) \cdot p(\tilde{y}_i|\mathbf{x}) \quad .$$

This assumption, and simple transformations using basic laws of probability lead to

$$\begin{aligned} E[\Delta w_{a,i}] = 0 \quad & \Leftrightarrow \quad \frac{1 + e^{w_{a,i}}}{1 + e^{-w_{a,i}}} = \frac{\int_X p(\tilde{y}_i = 1|\mathbf{x}) \; p(r = 1|\mathbf{x}, a) \; p(\mathbf{x}|a) \; d\mathbf{x}}{\int_X p(\tilde{y}_i = 1|\mathbf{x}) \; p(r = 0|\mathbf{x}, a) \; p(\mathbf{x}|a) \; d\mathbf{x}} \\ & \Leftrightarrow \quad e^{w_{a,i}} = \frac{\int_X p(\tilde{y}_i = 1, r = 1|\mathbf{x}, a) \; p(\mathbf{x}|a) \; d\mathbf{x}}{\int_X p(\tilde{y}_i = 1, r = 0|\mathbf{x}, a) \; p(\mathbf{x}|a) \; d\mathbf{x}} \\ & \Leftrightarrow \quad e^{w_{a,i}} = \frac{\int_X p(\tilde{y}_i = 1, r = 1, \mathbf{x}|a) d\mathbf{x}}{\int_X p(\tilde{y}_i = 1, r = 0, \mathbf{x}|a) d\mathbf{x}} \\ & \Leftrightarrow \quad e^{w_{a,i}} = \frac{p(\tilde{y}_i = 1, r = 1|a)}{p(\tilde{y}_i = 1, r = 0|a)} \\ & \Leftrightarrow \quad e^{w_{a,i}} = \frac{p(r = 1|\tilde{y}_i = 1, a)}{p(r = 0|\tilde{y}_i = 1, a)} \\ & \Leftrightarrow \quad w_{a,i} = \log \frac{p(r = 1|\tilde{y}_i = 1, a)}{p(r = 0|\tilde{y}_i = 1, a)} \quad . \end{aligned}$$

If the active (virtual) feature $\tilde{y}_i$ was known, the corresponding weight $w_{a,i}$ would directly indicate the log-odd of obtaining reward with action a. In this scenario, however, only the continuous features $y_i(\mathbf{x}), i = 1, \ldots, n$ are known. Due to the normalization, the feature values sum up to 1, and one can therefore weight every $w_{a,i}$ by $y_i(\mathbf{x})$, yielding

$$L_a(\mathbf{x}) = \sum_{i=1}^{n} w_{a,i} \; y_i(\mathbf{x}) \quad , \tag{6.33}$$

which is an interpolation between the reward log-odds $w_{a,i}$ for different features $\tilde{y}_i$. The interpolation weights are in this case the factors $y_i(\mathbf{x})$, which means that

those features $\tilde{y}_i$ which are more likely to be active contribute more to the weighted sum, since $y_i(\mathbf{x})$ is proportional to $p(\tilde{y}_i = 1|\mathbf{x})$. $L_a(\mathbf{x})$ thus approximates the reward log-odd $\log \frac{p(r=1|\mathbf{x},a)}{p(r=0|\mathbf{x},a)}$, and the reward probability $p(r = 1|\mathbf{x}, a)$ can be approximated by

$$p(r = 1|\mathbf{x}, a) \approx \sigma(L_a(\mathbf{x})) = \frac{1}{1 + e^{-L_a(\mathbf{x})}} \quad , \tag{6.34}$$

where $\sigma(.)$ is the log-sigmoidal transfer function.

6.6.1 Computer Experiments with continuous input

For the following experiment reward distributions were defined on single continuous input variables $x \in [0, 1]$. For every action a different reward distribution was modeled, and the learner's task was to approximate the true reward distributions with the continuous Bayesian Hebb rule (6.32), and to choose the action with the highest reward probability. 2000 training trials with inputs drawn from a uniform distribution on $[0, 1]$ were used, and the performance after every update was measured on 500 independent test trials. 20 RBFs with constant widths $s = 0.05$ were used for the input preprocessing. The centers of the RBFs were equally distributed in the interval $[0, 1]$.
Figure 6.11 shows the performance at every training trial, and the approximations of the reward distributions that were obtained after 2000 training trials. The average reward obtained after training is close to the best possible performance, and the reward distributions are learned accurately.

6.7 Discussion

6.7.1 Summary and open problems

We have proposed in this chapter a simple neural network architecture for learning and decision making, which makes use of two learning processes that operate on two different time scales. We assume that generic dependencies among sensory input variables or features, or in other words, the factors of the underlying Bayesian network, are detected on a larger time scale, and that combinations of conditionally dependent input features are presented to the decision stage through sparse population coding. We have shown that on the basis of such preprocessing, the optimal policy can be represented as a WTA operation applied to weighted sums, and the corresponding weights can be learnt very fast. In fact, we have shown that a very simple Hebbian learning rule (the reward-modulated Bayesian Hebb rule) can integrate information from past experience in a close to optimal way. The models that we presented and analyzed are biologically plausible and arguably minimal with regard to their complexity, but nevertheless can be shown to asymptotically approximate theoretically optimal performance. All information from past experience is stored in synaptic weights of simple linear neuron models, and can therefore immediately be used for online decision making. In contrast to other learning

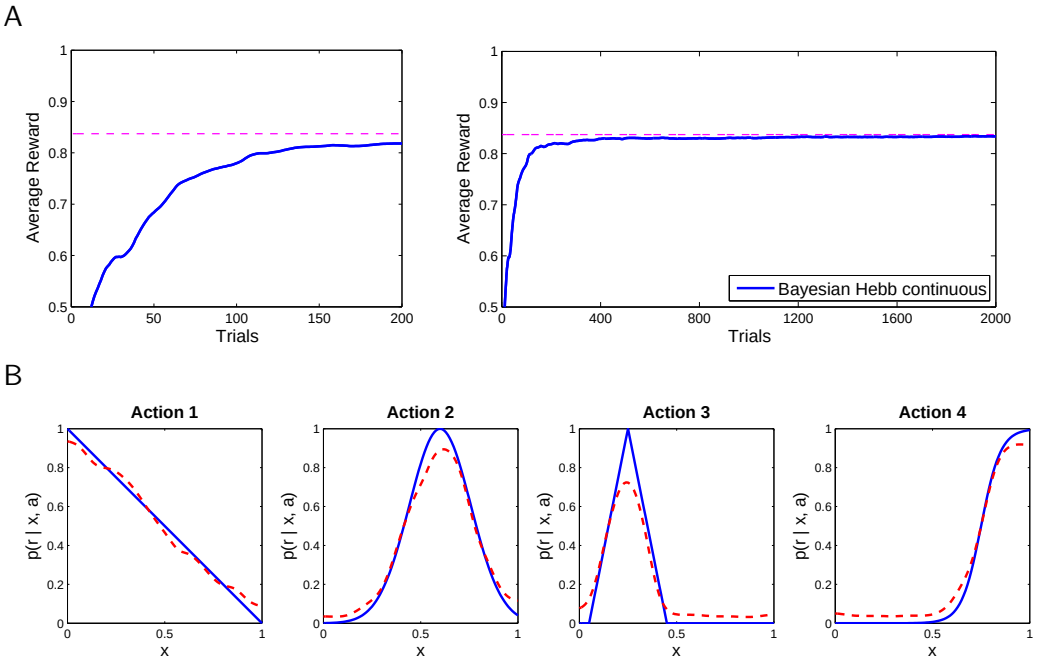


Figure 6.11: Performance of the Bayesian Hebb rule for continuous inputs. The input preprocessing consists of 20 RBF kernels that yield a population code $\mathbf{y}$ for the continuous inputs $\mathbf{x}$. **A)** Average reward of the learner obtained on 500 independent test trials during training on 2000 trials (left: performance during the first 200 training trials). The performance level rises quickly and in the end is close to the best possible performance of an optimal action selector (horizontal dashed line). Error bars are in the range of 10^{-3} and are omitted for clarity. Results are averaged over 32 runs. **B)** Approximation of the reward probabilities learned by the continuous Bayesian Hebb rule after 2000 training trials. The learned approximation (red, dashed line) is very close to the true reward distribution (blue, solid line).

rules that have previously been proposed for modeling animal learning — such as the Rescorla-Wagner rule (Rescorla & Wagner, 1972; Yuille, 2006), the perceptron learning rule, or learning rules based on the Kalman-filter model (Sutton, 1992; Dayan & Kakade, 2001) — this new learning rule is a truly Hebbian learning rule. Its weight updates depend on the current pre- and postsynaptic activity, as well as on a third signal (Bailey et al., 2000) that contains information about success or failure of the currently selected decision, but not on the current values of the other weights (or the resulting weighted sums of input variables). All information required for the weight update is therefore available locally at the synapse.

A major advantage of the local nature of purely Hebbian learning rules is that synapses can be removed or added to a neuron, without changing the target weights of the other synapses. One can therefore view the reward-modulated Bayesian Hebb rule as a candidate for learning in self-organizing organisms with developing neural structure. Assume, for example, that an input variable x_{new} is added, and the population code is appropriately modified. Then all weights belonging to factors in the factor graph that are not connected to x_{new} are unaffected, and can still be used for decision making. Removal or addition of single weights does however affect the decision making process, if the resulting population code does not match either the SP or GP encoding.

The Bayesian Hebb rule is one of very few online learning rules that admit a rigorous theoretical analysis of their convergence properties. We have shown that the theoretically optimal values of the weights are fixed point attractors for expected weight changes (see Figure 6.2B). This implies in particular, that learning cannot get stuck in local minima of some loss function. In fact, one can easily show that the expected weight updates give rise to an exponentially fast contracting dynamical system in weight space. Hence, this learning process falls into the theoretical framework of *contracting systems*, proposed by Lohmiller and Slotine (1998). According to this theory, this learning process can therefore be combined with other adaptive processes that also exhibit a contracting dynamics of adaptive parameters. Their theory guarantees that the resulting hybrid learning system will also converge.

We have also considered in section 6.3 a computationally simpler linear version of the Bayesian Hebb rule. Although this rule is only an approximation to the Bayesian Hebb rule, and theoretical convergence results are weaker (see the discussion in Section 6.3.1), we have shown that it performs almost equally well in a large number of complex decision making tasks (see Figures 6.6, 6.7, 6.12). The linear Bayesian Hebb rule is similar to well-known mathematical models for Hebbian learning, and may therefore provide a new interpretation of these learning rules as approximations to more complex plasticity mechanisms

In this chapter we have studied the scenario of online reward-based learning of decision making with multiple alternatives from stochastic rewards and input signals, which is important for fields like operant conditioning or reinforcement learning. In section 6.2.2 we have shown analytically and empirically (Figure 6.2A) that the Bayesian Hebb rule achieves near optimal learning in terms of learning speed, and asymptotically approaches the optimal policy for the given preprocess-

ing mechanism. We have supported this theoretical prediction through a variety of computer simulations of decision tasks (see Figures 6.5, 6.8, 6.11). The resulting higher learning speed is particularly interesting in our context of reward-based learning, where most learning algorithms are too slow to be applicable to real-world problems. Hence the contribution of this study can be seen as another step in the program to speed up reinforcement learning by making near optimal use of previous experience. We have shown in section 6.5.2 that this approach can also be applied to non-stationary distributions of inputs and rewards.

The question, how the brain forms decisions that involve more than two alternatives, is one of the most important open research problems (Gold & Shadlen, 2007). For binary decisions, Wald's sequential probability ratio test (Wald & Wolfowitz, 1948) provides a theoretically optimal tool for learning and decision making from limited evidence. In this case it is sufficient to update a single decision variable, and compare it to a threshold value. For problems with more than 2 alternatives it is unclear whether an optimal test exists, and tests that guarantee asymptotic optimality, such as the method developed by Dragalin et al. (1999) become much more complex (see (Gurney & Bogacz, 2006) for a possible neural implementation). Here we have studied a simpler network model, which does not select actions optimally in the sense of sequential analysis. It converges asymptotically to an optimal policy, and uses heuristic strategies for choosing actions during learning. We have analyzed a model that is based on the Winner-Take-All (WTA) operation, and directly uses the learned weights for the evaluation of actions. We have shown that if WTA is applied to several linear neurons, each of which learns via the Bayesian Hebb rule to approximate the log-odd of receiving a reward for an associated action (see Figure 6.1), our simple model can handle the case of more than two decision alternatives without any extra effort (see sections 6.2 and 6.3 for the theoretical analysis and Figures 6.5, 6.6, 6.8, 6.11 for empirical tasks).

WTA-circuits are of interest in the context of neural network models for action selection, since it has been suggested that generic cortical microcircuits implement a soft version of WTA-circuits (where $z_a > 0$ also for the runner-ups in the competition among the L_a), see (Douglas & Martin, 2004). This view is supported by the anatomical observation that the output cells (pyramidal neurons) of cortical microcircuits are subject to lateral inhibition (each pyramidal neuron excites inhibitory interneurons that target other pyramidal neurons). It is also supported by the physiological observation that simultaneous activation of very large numbers of sensory neurons (for example in the retina) is transformed through cortical processing into sparse activity of neurons in higher sensory areas (e.g., area IT). Consequently, WTA-circuits have become a primary target for the design of neurally inspired electronic hardware (Hahnloser et al., 2000; Neftci et al., 2008).

The components of our neural network model (Figure 6.1) have substantial experimental and theoretical support. Hebbian learning, and the use of weighted sums for decision making (Roth, 1999b) is clearly feasible for biological neurons. The other essential ingredient of our model for reward-based learning of decision making is a suitable preprocessing of variables $\mathbf{x}$ (typically representing sensory

inputs) that form the evidence on which a decision has to be based in a single trial. Our model requires a sparse population coding of the values of these variables (both for variables with discrete and for variables with continuous values, see section 6.6). Sparse encodings (Olshausen & Field, 1996), or population codes are common models for coding strategies of the brain, and experimental evidence for the existence of such codes has been found in various brain areas of different species (see e.g. Pouget & Latham, 2002; O'Keefe et al., 1998; Georgopoulos et al., 1986).

Furthermore, in the case of conditioned dependencies among variables our model assumes that there exists a population coding for "complex features" (reminiscent of neural codes reported for example for visual areas V2 and IT), i.e. for combinations of variables (see Figure 6.4 for an example). Hence, our simple neural network model for learning decision making entails concrete predictions for the computational strategies, neural codes, and learning mechanisms in those cortical areas that provide information about sensory inputs in a highly processed form to other cortical areas where decisions are made. It proposes that those subgroups of sensory variables (from the same or different sensory modalities) that have statistical dependencies, such as those represented by a factor graph (Kschischang et al., 2001), are brought together in some cortical microcircuits, and that projection neurons from these cortical microcircuits each assume a high firing rate for a particular combination of values of these variables (thereby mimicking the output variables y_i of our general preprocessing, see section 6.4.1).

This link of factor graph theory and experimentally observed population codes provides a novel view on the potential role of sensory areas that provide input to higher decision making stages in the brain. The proposed preprocessing has the advantage of relieving the subsequent decision stage from complex computations (such as belief propagation via message passing) and nonlinear learning devices. In fact, it enables the decision stage to use only linear operations in conjunction with WTA. It also enables the decision stage to accumulate evidence from history through the very simple and robust Hebbian learning processes that were discussed in this chapter.

In this study we assume that the graph structure of the factor graph is known, which is a very common assumption for parameter learning algorithms in graphical models (see e.g. Neapolitan, 2004; Jensen & Nielsen, 2007). The evolution of preprocessing circuits is obviously a complex process, and the design of learning algorithms that generate such preprocessing of sensory inputs is an interesting open problem. Testing variables for (conditional) dependence is perhaps a less formidable problem for a neural network than it may appear on first sight, provided one assumes that numerous autonomous learning processes try to predict each variable in terms of others. Dependencies among the variables exist, and can in principle be found autonomously by this process, whenever such prediction learning turns out to be successful. As mentioned above, such relationships between input signals may be learned on much longer time scales than decision strategies, which require very fast adaptation.

Other obvious open problems that arise from our model are whether it can be

implemented with spiking neurons, and whether there exist relationships between the theoretically optimal reward-modulated Bayesian Hebb rule and concrete heterosynaptic learning mechanisms of biological synapses such as those discussed in (Bailey et al., 2000). Another open problem concerns a possible extension of our model to rewards signals with more than two values, to third signals that represent predictions of rewards, and to reward based learning in continuous time.

Altogether our simple neural network model for learning decision making has shown that this problem is in some aspects less difficult than it may appear on first sight. It remains to be explored whether biological neural systems have adopted related implementation strategies, or have found even simpler solutions to this problem.

6.7.2 Related Work

6.7.2.1 Models for Decision Making

The study of decision making in biological systems dates back to the classical experiments by Pavlov, in which dogs learned associations between cues and rewards. On the other hand, operant or instrumental conditioning is concerned with associations between actions and rewards, and how behavior is modified through reward and punishment. The goal is to learn a policy, i.e. a way to select actions near-optimally in response to environmental stimuli. According to Sugrue et al. (2005), biological organisms first transform sensory input into decision related variables, e.g. value representations in area LIP for visual discrimination tasks in monkeys (Yang & Shadlen, 2007). An unknown computational mechanism maps the values of these variables to the probability of reward for executing various actions, which then leads to a motor response. An actor-critic model is assumed, in which the actor and the critic are two modules that operate with a common reward currency. The critic adapts the value of every action to the perceived reward probabilities, thereby altering the decision transformation, which the actor uses to choose actions. An example for models of instrumental conditioning is the experiment of Montague et al. (1995), in which the behavior of a foraging bee is simulated with a neural network model and a suitable learning rule (a variation of the Rescorla-Wagner rule). X. J. Wang (2002) has described a recurrent cortical network model, which uses feedback and winner-take-all mechanisms to integrate information in visual discrimination tasks with two possible outcomes. Gurney and Bogacz (2006) have presented a model for optimal decision making with multiple actions, which models the functionality of the basal ganglia. Further neural network models for decision making have been reviewed in (Sakai, Okamoto, & Fukai, 2006).

6.7.2.2 Learning Rules for Decision Making

The classical model for learning associations of stimuli, actions, and rewards is the Rescorla-Wagner rule (Rescorla & Wagner, 1972). It was the first mathematical model for learning that could explain most of the effects observed in animal behavior

studies. In particular it was able to explain reactions based on combinations of stimuli. Reward associations for many conditioning paradigms, such as e.g. partial reinforcement, inhibitory conditioning, or extinction can be learned by the Rescorla-Wagner rule (and also by the Bayesian Hebb rule). The associative model of the Rescorla-Wagner rule represents the predicted amount of reward as a weighted sum of stimuli, and weights are updated using the difference between the predicted and the actually received reward (see (6.27)). The Rescorla-Wagner rule is therefore not a strictly Hebbian learning rule, because this error signal, rather than the activation of the post-synaptic neuron is required for the update. Studies by Schultz et al. have however indicated that such an error signal may be available in the form of the neuromodulator dopamine (Schultz, Dayan, & Montague, 1997).

Learning rules that minimize prediction errors were also useful to explain blocking phenomena in conditioning (Dayan & Abbott, 2001). However, some observed effects like backward blocking — an established reward association is unlearned, because another stimulus sufficiently explains the occurrence of rewards — can neither be sufficiently captured by the Rescorla-Wagner rule, nor by the Bayesian Hebb rule. The reason for this is that weights in these models can only be reduced, if unrewarded trials are observed (which is not the case in the backward blocking paradigm). Algorithms that specifically address learning of reward associations in the backward blocking scenario are based on Kalman filter models for conditioning (Sutton, 1992). Dayan and Yu (2003) argue that in addition to error correction, it is necessary to model the uncertainty in the parameter estimates during learning, and neuromodulators like acetylcholine or norepinephrine could signal such uncertainty in biological systems (Yu & Dayan, 2003). An artificial recurrent neural network model, which approximates the Kalman filter estimates of reward associations for backward blocking was presented by Dayan and Kakade (2001). A different learning mechanism is suggested by Griffiths and Tenenbaum (2005), who argue that phenomena like backward blocking could also be modeled by learning changes in the causal structure of the problem, rather than by learning new reward associations.

The mathematical problem of learning optimal action selection is also well-studied in the field of reinforcement learning (*RL*) (Sutton & Barto, 1998). Typical RL algorithms learn value- or Q-functions, which estimate the expected reward resulting from the execution of action a in state $\mathbf{x}$. The goal of RL is to converge to optimal policies, which select for every state those actions that maximize the expected reward (typically a discounted long-term reward for sequential decision problems). Classical RL algorithms do not directly aim at maximizing the online performance, i.e. the amount of reward obtained during learning, but typically employ some heuristics to tackle the exploration-exploitation dilemma. This dilemma concerns the trade-off of online performance (exploitation) and exploration of unseen parts of the state- and action space in order to improve the final policy. More recently the problem of optimizing online performance has attracted more attention in the RL literature (e.g. Kearns & Singh, 1998; Audibert et al., 2007; Auer et al., 2009). Asymptotic convergence of RL algorithms to the optimal policy can only be guaranteed for discrete environments, if action values are stored in look-up tables

with one entry for every combination of state and action. Such tabular representations are biologically not realistic, and for computers the memory requirements are too large for most real-world applications. Value functions are therefore approximated, but convergence results exist only for a limited number of approximation schemes (Bertsekas & Tsitsiklis, 1996).

Using Bayesian inference for action selection in uncertain environments was e.g. studied by Attias (2003), and Verma and Rao (2006). They consider the problem of planning action sequences of fixed length for partially observable Markov decision processes with one or more fixed goal states. The dynamics of the environment are initially unknown. The learning part uses frequency counters to update conditional probabilities for transition and reward models. Planning is reduced to Bayesian inference in graphical models based on the learned parameters, which is computed with standard algorithms, like e.g. belief propagation or the junction tree algorithm (Bishop, 2006). The posterior over actions, given that start and goal state are fixed, is computed and the maximally likely sequence of actions (and intermediate states in (Verma & Rao, 2006)) is selected. This approach is conceptually quite different from our approach, since our approach does not learn sequences of actions, and does not require a defined goal-state. The learned parameters in our model (the weights $w_{a,i}$) are not auxiliary variables, but are directly used in the decision making process. Furthermore, our approach only requires very basic and apparently biologically feasible mechanisms like Hebbian learning, weighted summations, and winner-take-all. Implementing full Bayesian inference is a much more difficult process, for which it is not clear how the brain can achieve it efficiently, although some models have been proposed (e.g. (Rao, 2007; Deneve, 2008b)). Lansner and Ekeberg (1998); Kononenko (1998); Lansner and Holst (1996) and Sandberg et al. (2002) have studied various learning rules (although not in a reinforcement learning context) that approximate optimal Bayesian inference. The learning rules differ from the Bayesian Hebb rule that was introduced in this chapter primarily by fact that they require auxiliary counters for storing evidence from past experience.

6.7.2.3 Analogies to recent experimental studies of decision making in primates

Recent experimental results by Yang and Shadlen (2007) have shown that the previous experience of macaque monkeys in probabilistic decision tasks is represented by the firing rates of neurons in area LIP in the form of the log-likelihood ratio of receiving a reward for a particular action a in response to a stimulus $\mathbf{x}$, like in equation (6.1) of our framework. In their experiment a monkey had to choose at each trial between two possible actions. It could choose to move the eyes either towards a red target R ($a = R$) or a green target G ($a = G$). The probability that a reward was received at either choice depended on four visual input stimuli $\mathbf{x} = (x_1, x_2, x_3, x_4)$ that had been shown at the beginning of the trial. Every stimulus $x_k, k = 1, \ldots, 4$, was one shape s_j out of a set of ten possibilities $\{s_1, \ldots, s_{10}\}$ and had an associated weight $\omega_k = \omega(s_j)$, which had been defined by the experimenter. The log-odd of

obtaining a reward was equal to the sum of $\omega_1, \ldots, \omega_4$:

$$\log \frac{p(r=1|\mathbf{x}, a=R)}{p(r=1|\mathbf{x}, a=G)} = \sum_{k=1}^{4} \omega_k \quad . \tag{6.35}$$

The monkey thus had to combine the evidence from four visual stimuli to optimize its action selection behavior. It also had to find out that reward probabilities only depended on the presented shapes, but not on the order or location in which they were presented. A reward was assigned before the trial to one of the targets according to the distribution (6.35).

One can easily model this task in our framework, using a simple population code $\mathbf{y} = \mathbf{\Phi}(\mathbf{x})$ as in (6.19), where the stimulus $\mathbf{x}$ was encoded by a 40-dimensional binary vector $\mathbf{y}$ with exactly $m = 4$ inputs being 1. The positions of the 1's corresponded to the four visual shapes that were shown during a trial. The log-odd of obtaining reward with action $a = R$ can then be written as a weighted sum

$$\log \frac{p(r=1|\mathbf{y}, a=R)}{p(r=0|\mathbf{y}, a=R)} = \sum_{i=1}^{40} w_i^* y_i \quad , \tag{6.36}$$

with

$$w_i^* \quad = \quad \log \frac{p(r=1|y_i=1, a=R)}{p(r=0|y_i=1, a=R)} \quad . \tag{6.37}$$

Due to the symmetry of the task (reward is either at R or G), the log-odds in (6.35) and (6.36) are equivalent. The weights w_i can be learned with an efficient version of the reward-modulated Bayesian Hebb rule (6.8), which takes this symmetry into account. The equilibrium w_i^* of weight w_i under this slightly modified rule is then exactly at the desired value (6.37). We simulated this task, using a learner with the reward-modulated Bayesian Hebb rule and a $1/N_i$ learning rate for every weight. Figure 6.12A shows that this task can be successfully learned both by the exact reward-modulated Bayesian Hebb rule (6.8) and the linear approximation (6.13). The learning rules learn as fast as the non-Hebbian Rescorla-Wagner rule (6.27), and their performance is close to the theoretical optimum after 1000 training trials. Furthermore Figures 6.12B and C show that the intermediate and final policies resemble the behavior that was reported for two monkeys in (Yang & Shadlen, 2007).

The experimental data of Yang and Shadlen (2007) are consistent with the assumption that monkeys apply a WTA-operation to the log-likelihood ratios

$$L_a = \log \frac{p(r=1|\mathbf{x}, a)}{p(r=0|\mathbf{x}, a)} \quad ,$$

which are, according to their model, represented through firing rates of neurons in area LIP. It is not known, which values are represented by the firing rates y_i of the presynaptic neurons of these neurons. In our simple model we model the neurons

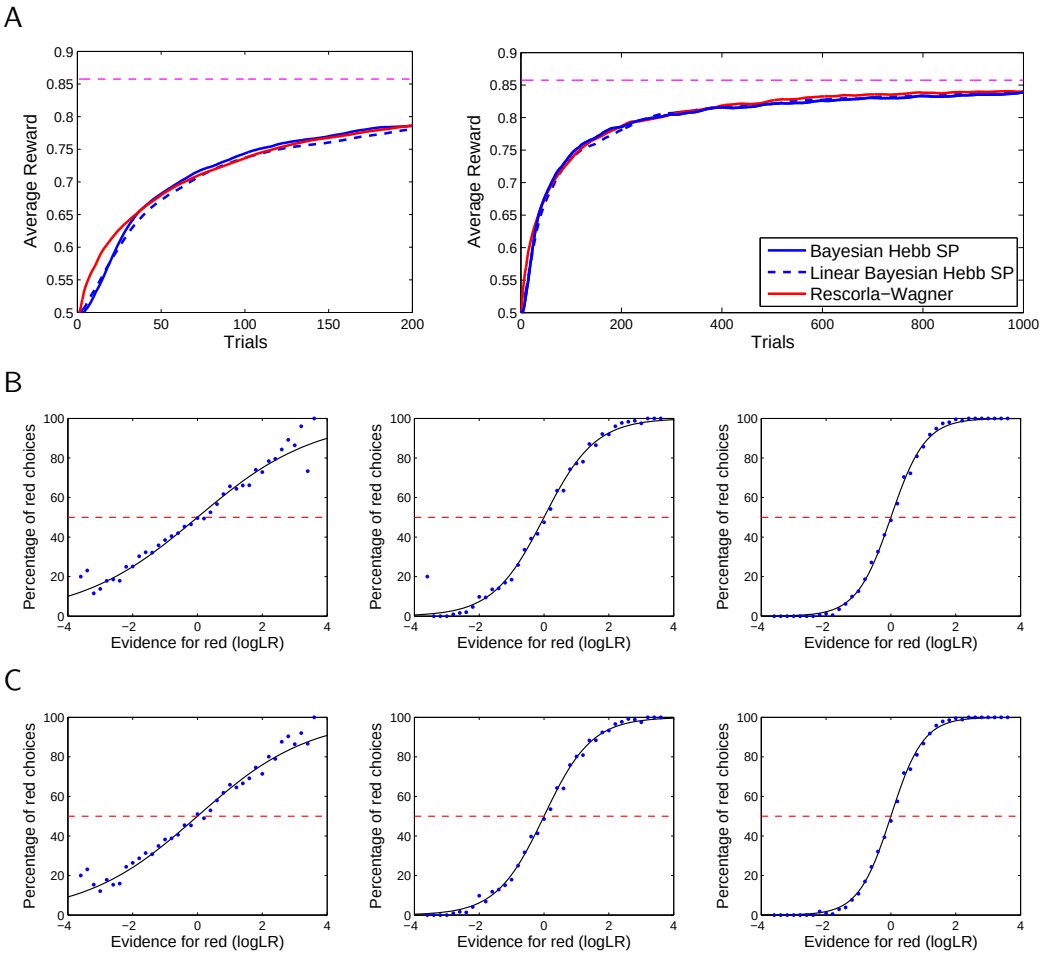


Figure 6.12: Performance of the reward-modulated Bayesian Hebb rule in the model for the conditioning task by Yang and Shadlen (see (Yang & Shadlen, 2007) for details). **A)** The reward-modulated Bayesian Hebb rule learns as fast as the non-Hebbian Rescorla-Wagner rule (curves result from averaging over 32 repetitions of the experiment, where the average reward was measured on 500 independent test trials). The horizontal dashed line reflects the theoretically best possible performance. Error bars are in the range of 10^{-2} and are omitted for clarity. **B, C)** Action selection policies (greedy policy according to (6.3)) resulting from the model using the exact Bayesian Hebb rule (6.8) (**B**) or the linear Bayesian Hebb rule (6.13) (**C**) after 100 (left), 500 (middle), and 1000 (right) trials, fitted by sigmoidal curves (results are from 32 repetitions of the experiment, where the behavior was measured on 1000 independent test trials). The policies represented by the left and right panels are qualitatively similar to the policies adopted by monkeys H and J in the experiments by Yang and Shadlen (2007) after learning (see Figure 1b in (Yang & Shadlen, 2007)).

within the WTA circuit as linear neurons, and assume that their output L_a can be written as a linear sum $L_a = \sum_{i=0}^{n} w_{a,i}\, y_i$ of variables y_i that represent a population coding of the sensory input $\mathbf{x}$. As we have shown in section 6.4, if this population coding is chosen in a suitable way, the true reward log-odd $\log \frac{p(r=1|\mathbf{x},a)}{p(r=0|\mathbf{x},a)}$ can in fact be written as such weighted sum. Hence our theoretical framework makes concrete predictions about the nature of the transformation of raw sensory inputs $\mathbf{x}$ to inputs $\mathbf{y}$ for higher brain areas that select suitable responses. The required weights $w_{a,i}$ can be learnt by the reward-modulated Bayesian Hebb rule, and a linear Poisson neuron whose weights are updated according to this rule will adapt for each trial a firing rate proportional to the log-likelihood ratio $\log \frac{p(a=R|\mathbf{x})}{p(a=G|\mathbf{x})}$. This response matches that of the neurons in area LIP shown in Figure 2c and 3b of (Yang & Shadlen, 2007).[4]

The Bayesian Hebb rule provides an arguably minimal model for the biological data of Yang and Shadlen (2007). One difference between their results and our model is that learning is much faster in our model. This could be explained by the fact that many aspects of the probabilistic decision task of Yang and Shadlen (2007) - e.g. the fact that the reward policy was stationary, the fact that the reward probabilities did not depend on the order of appearance, or the spatial location of the shown icons, and the fact that reward probabilities did not depend on any other aspects that the monkeys had perceived before or during a session - also had to be learned by the monkeys, whereas they were assumed as given in our model. Learning of these invariances and symmetries was actually quite hard in the set-up of Yang and Shadlen (2007) since rewards were given stochastically, rather than by deterministic laws (note that even many humans believe to "learn" various misleading reward-predictors while gambling for a long time in the lottery or casinos). An interesting open question is whether reward-based learning of decision making by humans or animals can approach the learning speed of the Bayesian Hebb rule when such differences between the learning tasks of the living organisms and the mathematical model have been removed.

6.7.3 Conclusion

We have demonstrated the functionality of a simple neural network model for learning of asymptotically optimal action selection, which uses only biologically plausible mechanisms such as reward-modulated Hebbian learning, sparse population coding, and winner-take-all computations. Furthermore we have shown that on the basis of a suitable preprocessing that takes dependencies among salient variables into account, a very simple Hebbian learning rule can converge towards optimal policies extremely fast. On the side, our approach offers concrete processing goals for brain areas that integrate multi-modal sensory input, in order to facilitate learning and decision making in higher brain areas. Empirical results have confirmed that the new reward-modulated Bayesian Hebb rule, and an even simpler linear

[4] Note that the optimal weights w_i^* are equal to the weights $\omega_k = \omega(s_j)$ that were assigned to the different visual shapes s_j.

approximation to it, compare favorably to well-known non-Hebbian learning rules for action-selection tasks. Our results suggest that learning and decision making under uncertainty can be implemented very efficiently in biological neural systems.

6.8 Acknowledgments

This chapter is based on the paper *Reward-modulated Hebbian Learning of Decision Making*, which was written by myself (MP), Bernhard Nessler (BN), Rodney J. Douglas (RD), and Wolfgang Maass (WM). The theory of the reward-modulated Bayesian Hebb rule was developed by MP, based on previous results by BN. All experiments were performed by MP, and the paper was written by MP, WM, and RD with additional input from BN.

Chapter 7

STDP enables spiking neurons to detect hidden causes of their inputs

Contents

The principles by which spiking neurons contribute to the astounding computational power of generic cortical microcircuits, and how spike-timing-dependent plasticity (STDP) of synaptic weights could generate and maintain this computational function, are unknown. We show here that STDP, in conjunction with a stochastic soft winner-take-all (WTA) circuit, induces spiking neurons to generate through their synaptic weights implicit internal models for subclasses (or "causes") of the high-dimensional spike patterns of hundreds of pre-synaptic neurons. Hence these neurons will fire after learning whenever the current input best matches their internal model. The resulting computational function of soft WTA circuits, a common network motif of cortical microcircuits, could therefore be a drastic dimensionality reduction of information streams, together with the autonomous creation of internal models for the probability distributions of their input patterns. We show that the autonomous generation and maintenance of this computational function can be explained on the basis of rigorous mathematical principles. In particular, we show that STDP is able to approximate a stochastic online Expectation-Maximization (EM) algorithm for modeling the input data. A corresponding result is shown for Hebbian learning in artificial neural networks.

7.1 Introduction

It is well-known that synapses change their synaptic efficacy ("weight") w in dependence of the difference $t_{post} - t_{pre}$ of the firing times of the post- and presynaptic neuron according to variations of a generic STDP rule (see (Dan & Poo, 2004) for a recent review). However, the computational benefit of this learning rule is largely unknown (Abbott & Nelson, 2000; Morrison et al., 2007). It has also been observed that local WTA-circuits form a common network-motif in cortical microcircuits (Douglas & Martin, 2004). However, it is not clear how this network-motif contributes to the computational power and adaptive capabilities of laminar cortical microcircuits, out of which the cortex is composed. Finally, it has been conjectured for quite some while, on the basis of theoretical considerations, that the discovery and representation of hidden causes of their high-dimensional afferent spike inputs is a generic computational operation of cortical networks of neurons (Hinton & Ghahramani, 1997). One reason for this belief is that the underlying mathematical framework, Expectation-Maximization (EM), arguably provides the most powerful approach to unsupervised learning that we know of. But one has so far not been able to combine these three potential pieces (STDP, WTA-circuits, EM) of the puzzle into a theory that could help us to unravel the organization of computation and learning in cortical networks of neurons.

We show in this chapter that STDP in WTA-circuits approximates EM for discovering hidden causes of large numbers of input spike trains. We first demonstrate this in section 7.2 in an application to a standard benchmark dataset for the discovery of hidden causes. In section 7.3 we show that the functioning of this demonstration can be explained on the basis of EM for simpler non-spiking approximations to the spiking network considered in section 7.2.

7.2 Discovery of hidden causes for a benchmark dataset

We applied the network architecture shown in Fig. 7.1A to handwritten digits from the MNIST dataset (LeCun et al., 1998).[1] This dataset consists of $70,000$ 28×28-pixel images of handwritten digits[2], from which we picked the subset of $20,868$ images containing only the digits $0, 3$ and 4. Training examples were randomly sampled from this subset with a uniform distribution of digit classes.

Pixel values x_j were encoded through population coding by binary variables y_i (spikes were produced for each variable y_i by a Poisson process with a rate of 40

[1] A similar network of spiking neurons had been applied successfully in (Gupta & Long, 2007) to learn with STDP the classification of symbolic (i.e., not handwritten) characters. Possibly our theoretical analysis could also be used to explain their simulation result.

[2] Pixels were binarized to black/white. All pixels that were black in less than 5% of the training examples were removed, leaving $m = 429$ external variables x_j, that were encoded by $n = 858$ spiking neurons y_i. Our approach works just as well for external variables x_j that assume any finite number of values, provided that they are presented to the network through population coding with one variable y_i for every possible value of x_j. In fact, the approach appears to work also for the commonly considered population coding of continuous external variables.

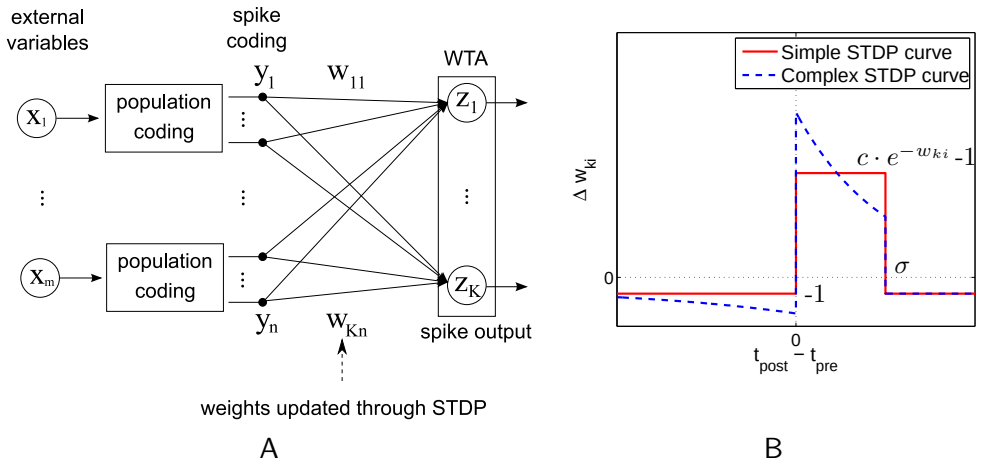


Figure 7.1: **A)** Architecture for learning with STDP in a WTA-network of spiking neurons. **B)** Learning curve for the two STDP rules that were used (with $\sigma = 10$ms). The synaptic weight w_{ki} is changed in dependence of the firing times t_{pre} of the presynaptic neuron y_i and t_{post} of the postsynaptic neuron z_k. If z_k fires at time t without a firing of y_i in the interval $[t-\sigma, t+2\sigma]$, w_{ki} is reduced by 1. The resulting weight change is in any case multiplied with the current learning rate η, which was chosen in the simulations according to the variance tracking rule (see footnote 7).

Hz for $y_i = 1$, and 0 Hz for $y_i = 0$, at a simulation time step of 1ms, see Fig. 7.2A). Every training example $\mathbf{x}$ was presented for 50ms. Every neuron y_i was connected to all $K = 10$ output neurons $z_1, \ldots, z_{10}$. A Poisson process caused firing of one of the neurons z_k on average every 5ms. The WTA-mechanism ensured that only one of the output neurons could fire at any time step. The winning neuron at time step t was chosen from the soft-max distribution

$$p(z_k \text{ fires at time } t|\mathbf{y}) = \frac{e^{u_k(t)}}{\sum_{l=1}^{K} e^{u_l(t)}}, \tag{7.1}$$

where $u_k(t) = \sum_{i=1}^{n} w_{ki}\tilde{y}_i(t) + w_{k0}$ represents the current membrane potential of neuron z_k (with $\tilde{y}_i(t) = 1$ if y_i fired within the time interval $[t - 10\text{ms}, t]$, else $\tilde{y}_i(t) = 0$).[3]

STDP with the learning curves shown in Fig. 7.1B was applied to all synapses w_{ki} for an input consisting of a continuous sequence of spike encodings of hand-written digits, each presented for 50ms (see Fig. 7.2A).[4] The learning rate η was

[3] This amounts to a representation of the EPSP caused by a firing of neuron y_i by a step function, which facilitates the theoretical analysis in section 7.3. Learning with the spiking network works just as well for biologically realistic EPSP forms.

[4] Whereas the weights in the theoretical analysis of section 7.3 will approximate logs of probabilities (see (7.6)), one can easily make all weights non-negative by restricting the range of these *log*-probabilities to $[-5, 0]$, and then adding a constant 5 to all weight values. This transformation gives rise to the factor $c = e^5$ in Fig. 7.1B.

chosen locally according to the variance tracking rule (see footnote 7).Fig. 7.2C shows that for subsequent representations of new handwritten samples of the same digits only one neuron responds during each of the 50ms while a handwritten digit is shown. The implicit internal models which the output neurons $z_1, \ldots, z_{10}$ had created in their weights after applying STDP are made explicit in Fig. 7.3B and C. Since there were more output neurons than digits, several output neurons created internal models for different ways of writing the same digit. When after applying STDP to 2000 random examples of handwritten digits 0 and 3 also examples of handwritten digit 4 were included in the next 2000 examples, the internal models of the 10 output neurons reorganized autonomously, to include now also two internal models for different ways of writing the digit 4. In order to efficiently observe the effect of the ongoing learning process on the synaptic weights in the spiking network we calculated the firing probabilities at the presentation of the test examples without the influence of the stochastic spike generation, by setting $\tilde{y}_i(t) = y_i$ in (7.1). This leads to one static firing probability $p(z_k = 1|\mathbf{y})$ for each example $\mathbf{y}$. The adaptation of the synaptic weights is measured in Fig. 7.3.A by the normalized conditional entropy $H(L|Z)/H(L,Z)$ over all examples y, where L denotes the correct classification of each handwritten digit $\mathbf{y}$, and Z is a random variable with $p(Z = k|\mathbf{y}) = p(z_k = 1|\mathbf{y})$.

Since after training by STDP each of the output neurons fire preferentially for one digit, we can measure the emergent classification capability of the network. After 2000 training steps we obtain a classification error of 2.13% on the training set with 2 classes of digits. After 4000 training steps we obtain a classification error of 3.92% on the 3-class training set.

7.3 Underlying theoretical principles

We show in this section that one can analyze the learning dynamics of the spiking network considered in the preceding section (with the simple STDP curve of Fig. 7.1B with the help of Hebbian learning (using rule (7.12)) in a corresponding non-spiking neural network $\mathcal{N}_{\mathbf{w}}$. $\mathcal{N}_{\mathbf{w}}$ is a stochastic artificial neural network with the architecture shown in Fig. 7.1A, and with a parameter vector $\mathbf{w}$ consisting of thresholds w_{k0} $(k = 1, \ldots, K)$ for the K output units $z_1, \ldots, z_K$ and weights w_{ki} for the connection from the i^{th} input node y_i $(i = 1, \ldots, n)$ to the k^{th} output unit z_k. We assume that this network receives at each discrete time step a binary input vector $\mathbf{y} \in \{0,1\}^n$ and outputs a binary vector $\mathbf{z} \in \{0,1\}^K$ with $\sum_{k=1}^K z_k = 1$, where the k such that $z_k = 1$ is drawn from the distribution over $\{1, \ldots, K\}$ defined by

$$p(z_k = 1|\mathbf{y}, \mathbf{w}) = \frac{e^{u_k}}{\sum\limits_{l=1}^{K} e^{u_l}} \qquad \text{with} \quad u_k = \sum_{i=1}^{n} w_{ki}\ y_i + w_{k0} \quad . \tag{7.2}$$

We consider the case where there are arbitrary discrete external variables $x_1, \ldots, x_m$, each ranging over $\{1, \ldots, M\}$ (we had $M = 2$ in section 7.2), and

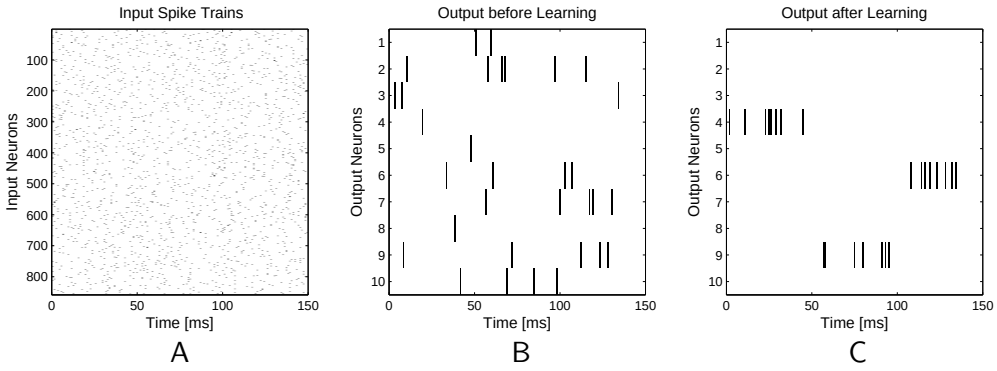


Figure 7.2: Unsupervised classification learning and sparsification of firing of output neurons after training. For testing we presented three examples from an independent test set of handwritten digits $0, 3, 4$ from the MNIST dataset, and compared the firing of the output-neurons before and after learning. **A)** Representation of the three handwritten digits $0, 3, 4$ for 50ms each by 858 spiking neurons y_i. **B)** Response of the output neurons before training. **C)** Response of the output neurons after STDP (according to Fig. 7.1B) was applied to their weights w_{ki} for a continuous sequence of spike encodings of 4000 randomly drawn examples of handwritten digits $0, 3, 4$, each represented for 50ms (like in panel A). The three output neurons z_4, z_9, z_6 that respond have generated internal models for the three shown handwritten digits according to Fig. 7.3C.

assume that these are encoded through binary variables $y_1, \ldots, y_n$ for $n = m \cdot M$ with $\sum_{i=1}^{n} y_i = m$ according to the rule

$$y_{(j-1)\cdot M+r} = 1 \quad \Longleftrightarrow \quad x_j = r \ , \qquad \text{for } j = 1, \ldots, m \text{ and } r = 1, \ldots, M. \tag{7.3}$$

In other words: the group G_j of variables $y_{(j-1)\cdot M+1}, \ldots, y_{(j-1)\cdot M+M}$ provides a population coding for the discrete variable x_j.

We now consider a class of probability distributions that is particularly relevant for our analysis: mixtures of multinomial distributions (Meilă & Heckerman, 2001), a generalization of mixtures of Bernoulli distributions (see section 9.3.3 of (Bishop, 2006)). This is a standard model for latent class analysis (McLachlan & Peel, 2000) in the case of discrete variables. Mixtures of multinomial distributions are arbitrary mixtures of K distributions $p_1(\mathbf{x}), \ldots, p_K(\mathbf{x})$ that factorize, i.e.,

$$p_k(\mathbf{x}) = \prod_{j=1}^{m} p_{kj}(x_j)$$

for arbitrary distributions $p_{kj}(x_j)$ over the range $\{1, \ldots, M\}$ of possible values for x_j. In other words: there exists some distribution over hidden binary variables

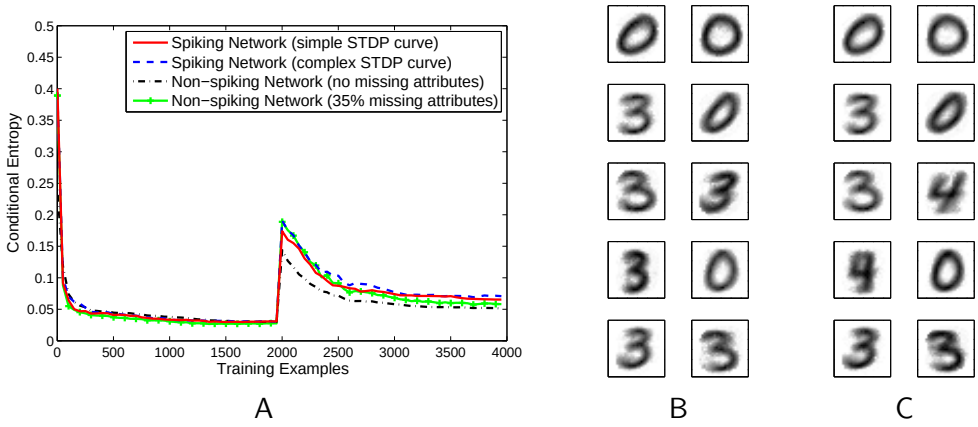


Figure 7.3: Analysis of the learning progress of the spiking network for the MNIST dataset. **A)** Normalized conditional entropy (see text) for the spiking network with the two variants of STDP learning rules illustrated in Fig. 7.1B (red solid and blue dashed lines), as well as two non-spiking approximations of the network with learning rule (7.12) that are analyzed in section 7.3. According to this analysis the non-spiking network with 35% missing values (green line) is expected to have a very similar learning behavior to the spiking network. 2000 random examples of handwritten digits 0 and 3 were presented (for 50ms each) to the spiking network as the first 2000 examples. Then for the next 2000 examples also samples of handwritten digit 4 were included. **B)** The implicit internal models created by the neurons after 2000 training examples are made explicit by drawing for each pixel the difference $w_{ki} - w_{k(i+1)}$ of the weights for input y_i and y_{i+1} that encode the two possible values (black/white) of the variable x_j that encodes this pixel value. One can clearly see that neurons created separate internal models for different ways of writing the two digits 0 and 3. **C)** Re-organized internal models after 2000 further training examples that included digit 4. Two output neurons had created internal models for the newly introduced digit 4.

z_k with $\sum_{k=1}^{K} z_k = 1$, where the k with $z_k = 1$ is usually referred to as a hidden "cause" in the generation of $\mathbf{x}$, such that

$$p(\mathbf{x}) = \sum_{k=1}^{K} p(z_k = 1) \cdot p_k(\mathbf{x}). \tag{7.4}$$

We first observe that any such distribution $p(\mathbf{x})$ can be represented with some suitable weight vector $\mathbf{w}$ by the neural network $\mathcal{N}_{\mathbf{w}}$, after recoding of the multinomial variables x_j by binary variables y_i as defined before:

$$p(\mathbf{y}|\mathbf{w}) = \sum_{k=1}^{K} e^{u_k^*} \qquad \text{with} \qquad u_k^* := \sum_{i=1}^{n} w_{ki}^* \, y_i + w_{k0}^* \quad , \tag{7.5}$$

for

$$w^*_{ki} := \log p(y_i = 1 | z_k = 1) \qquad \text{and} \qquad w^*_{k0} := \log p(z_k = 1) \quad . \tag{7.6}$$

In addition, $\mathcal{N}_{\mathbf{w}}$ defines for any weight vector $\mathbf{w}$ whose components are normalized, i.e.

$$\sum_{k=1}^{K} e^{w_{k0}} = 1 \quad \text{and} \quad \sum_{i \in G_j} e^{w_{ki}} = 1 \; , \qquad \text{for } j = 1, \ldots, m;\; k = 1, \ldots, K, \tag{7.7}$$

a mixture of multinomials of the type (7.4).

The problem of learning a generative model for some arbitrarily given input distribution $p^*(\mathbf{x})$ (or $p^*(\mathbf{y})$ after recoding according to (7.3)), by the neural network $\mathcal{N}_{\mathbf{w}}$ is to find a weight vector $\mathbf{w}$ such that $p(\mathbf{y}|\mathbf{w})$ defined by (7.5) models $p^*(\mathbf{y})$ as accurately as possible. As usual, we quantify this goal by demanding that

$$\mathrm{E}_{p^*}[\log p(\mathbf{y}|\mathbf{w})] \tag{7.8}$$

is maximized.

Note that the architecture $\mathcal{N}_{\mathbf{w}}$ is very useful from a functional point of view, because if (7.7) holds, then the weighted sum u_k at its unit z_k has according to (7.2) the value $\log p(z_k = 1|\mathbf{y}, \mathbf{w})$, and the stochastic WTA rule of $\mathcal{N}_{\mathbf{w}}$ picks the "winner" k with $z_k = 1$ from this internally generated model $p(z_k = 1|\mathbf{y}, \mathbf{w})$ for the actual distribution $p^*(z_k = 1|\mathbf{y})$ of hidden causes. We will not enforce the normalization (7.7) explicitly during the subsequently considered learning process, but rather use a learning rule (7.12) that turns out to automatically approximate such normalization in the limit.

Expectation Maximization (EM) is the standard method for maximizing $\mathrm{E}_{p^*}[\log p(\mathbf{y}|\mathbf{w})]$. We will show that the simple STDP-rule of Fig. 7.1B for the spiking network of section 7.2 can be viewed as an approximation to an online version of this EM method. We will first consider in section 7.3.1 the standard EM-approach, and show that the Hebbian learning rule (7.12) provides a stochastic approximation to the maximization step.

7.3.1 Reduction to EM

The standard method for maximizing the expected log-likelihood $\mathrm{E}_{p^*}[\log p(\mathbf{y}|\mathbf{w})]$ with a distribution p of the form $p(\mathbf{y}|\mathbf{w}) = \sum_{\mathbf{z}} p(\mathbf{y}, \mathbf{z}|\mathbf{w})$ with hidden variables $\mathbf{z}$, is to observe that $\mathrm{E}_{p^*}[\log p(\mathbf{y}|\mathbf{w})]$ can be written for arbitrary distributions $q(\mathbf{z})$ in the form

$$\mathrm{E}_{p^*}[\log p(\mathbf{y}|\mathbf{w})] = \mathcal{L}(q, \mathbf{w}) + \mathrm{E}_{p^*}[\mathrm{KL}(q(\mathbf{z})||p(\mathbf{z}|\mathbf{y}, \mathbf{w}))] \tag{7.9}$$

$$\mathcal{L}(q, \mathbf{w}) = \mathrm{E}_{p^*}\left[\sum_{\mathbf{z}} q(\mathbf{z}) \log \frac{p(\mathbf{y}, \mathbf{z}|\mathbf{w})}{q(\mathbf{z})}\right] \quad , \tag{7.10}$$

where KL(.) denotes the Kullback-Leibler divergence.

In the E-step one sets $q(\mathbf{z}) = p(\mathbf{z}|\mathbf{y}, \mathbf{w}^{old})$ for the current parameter values $\mathbf{w} = \mathbf{w}^{old}$, thereby achieving $\mathrm{E}_{p^*}[\mathrm{KL}(q(\mathbf{x})||p(\mathbf{z}|\mathbf{y}, \mathbf{w}^{old}))] = 0$. In the M-step one replaces $\mathbf{w}^{old}$ by new parameters $\mathbf{w}$ that maximize $\mathcal{L}(q, \mathbf{w})$ for this distribution $q(\mathbf{z})$. One can easily show that this is achieved by setting

$$w^*_{ki} = \log p^*(y_i = 1|z_k = 1), \qquad \text{and} \qquad w^*_{k0} = \log p^*(z_k = 1), \tag{7.11}$$

with values for the variables z_k generated by $q(\mathbf{z}) = p(\mathbf{z}|\mathbf{y}, \mathbf{w}^{old})$, while the values for the variables $\mathbf{y}$ are generated by the external distribution p^*. Note that this distribution of $\mathbf{z}$ is exactly the distribution (7.2) of the output of the neural network $\mathcal{N}_{\mathbf{w}}$ for inputs $\mathbf{y}$ generated by p^*.[5] In the following section we will show that this M-step can be approximated by applying iteratively a simple Hebbian learning rule to the weights $\mathbf{w}$ of the neural network $\mathcal{N}_{\mathbf{w}}$.

7.3.2 A Hebbian learning rule for the M-step

We show here that the target weight values (7.11) are the only equilibrium points of the following Hebbian learning rule:

$$\boxed{\Delta w_{ki} = \begin{cases} \eta\,(e^{-w_{ki}} - 1), & \text{if } y_i{=}1 \text{ and } z_k{=}1 \\ -\eta, & \text{if } y_i{=}0 \text{ and } z_k{=}1 \\ 0, & \text{if } z_k = 0, \end{cases}} \tag{7.12}$$

$$\boxed{\Delta w_{k0} = \begin{cases} \eta\,(e^{-w_{k0}} - 1), & \text{if } z_k{=}1 \\ -\eta, & \text{if } z_k{=}0 \end{cases}} \tag{7.13}$$

It is obvious (using for the second equivalence the fact that y_i is a binary variable) that

$$\begin{aligned} \mathrm{E}[\Delta w_{ki}] = 0 \;&\Leftrightarrow\; p^*(y_i{=}1|z_k{=}1)\eta(e^{-w_{ki}} - 1) - p^*(y_i{=}0|z_k{=}1)\eta = 0 \\ &\Leftrightarrow\; p^*(y_i{=}1|z_k{=}1)(e^{-w_{ki}} - 1) + p^*(y_i{=}1|z_k{=}1) - 1 = 0 \\ &\Leftrightarrow\; p^*(y_i{=}1|z_k{=}1)e^{-w_{ki}} = 1 \\ &\Leftrightarrow\; w_{ki} = \log p^*(y_i{=}1|z_k{=}1) \quad . \end{aligned} \tag{7.14}$$

Analogously one can show that $\mathrm{E}[\Delta w_{k0}] = 0 \Leftrightarrow w_{k0} = \log p^*(z_k{=}1)$. With similar elementary calculations one can show that $\mathrm{E}[\Delta w_{ki}]$ has for any $\mathbf{w}$ a value that moves w_{ki} in the direction of w^*_{ki} (in fact, exponentially fast).

One can actually show that one single step of (7.12) is a linear approximation of the ideal incremental update of $w_{ki} = \log \frac{a_{ki}}{N_k}$, with a_{ki} and N_k representing the values of the corresponding sufficient statistics, as $\log \frac{a_{ki}+1}{N_k+1} = w_{ki} + \log(1 +$

[5] Hence one can extend $p^*(\mathbf{y})$ for each fixed $\mathbf{w}$ to a joint distribution $p^*(\mathbf{y}, \mathbf{z})$, where the $\mathbf{z}$ are generated for each $\mathbf{y}$ by $\mathcal{N}_{\mathbf{w}}$.

$\eta e^{-w_k i}) - \log(1+\eta)$ for $\eta = \frac{1}{N_k}$. This also reveals the role of the learning rate η as the reciprocal of the equivalent sample size[6].

In order to guarantee the stochastic convergence (see (Kushner & Yin, 1997)) of the learning rule one has to use a decaying learning rate $\eta^{(t)}$ such that $\sum_{t=1}^{\infty} \eta^{(t)} = \infty$ and $\sum_{t=1}^{\infty} (\eta^{(t)})^2 = 0$.[7]

The learning rule (7.12) is similar to a rule that had been introduced in (Nessler et al., 2009) in the context of supervised learning and reinforcement learning. That rule had satisfied an equilibrium condition similar to (7.14). But to the best of our knowledge, such type of rule has so far not been considered in the context of unsupervised learning.

One can easily see the correspondence between the update of w_{ki} in (7.12) and in the simple STDP rule of Fig. 7.1B. In fact, if each time where neuron z_k fires in the spiking network, each presynaptic neuron y_i that currently has a high firing rate has fired within the last $\sigma = 10$ms before the firing of z_k, the two learning rules become equivalent. However since the latter condition could only be achieved with biologically unrealistic high firing rates, we need to consider in section 7.3.4 the case for the non-spiking network where some input values are missing (i.e., $y_i = 0$ for all $i \in G_j$; for some group G_j that encodes an external variable x_j via population coding).

We first show that the Hebbian learning rule (7.12) is also meaningful in the case of online learning of $\mathcal{N}_{\mathbf{w}}$, which better matches the online learning process for the spiking network.

7.3.3 Stochastic online EM

The preceding arguments justify an application of learning rule (7.12) for a number of steps within each M-step of a batch EM approach for maximizing $\mathrm{E}_p^*[\log p(\mathbf{y}|\mathbf{w})]$. We now show that it is also meaningful to apply the same rule (7.12) in an online stochastic EM approach (similarly as in (Sato, 1999)), where at each combined EM-step only one example $\mathbf{y}$ is generated by p^*, and the learning rule (7.12) is applied just once (for z_k resulting from $p(\mathbf{z}|\mathbf{y}, \mathbf{w})$ for the current weights $\mathbf{w}$, or simpler: for the z_k that is output by $\mathcal{N}_{\mathbf{w}}$ for the current input $\mathbf{y}$).

Our strategy for showing that a single application of learning rule (7.12) is expected to provide progress in an online EM-setting is the following. We consider

[6] The equilibrium condition (7.14) only sets a necessary constraint for the the quotient of the two directions of the update in (7.12). The actual formulation of (7.12) is motivated by the goal of updating a sufficient statistics.

[7] In our experiments we used an adaptation of the variance tracking heuristic from (Nessler et al., 2009). If we assume that the consecutive values of the weights represent independent samples of their true stochastic distribution at the current learning rate, then this observed distribution is the log of a beta-distribution of the above mentioned parameters of the sufficient statistics. Analytically this distribution has the first and second moments $\mathrm{E}[w_{ki}] \approx \log \frac{a_{ki}}{N_i}$ and $\mathrm{E}[w_{ki}^2] \approx \mathrm{E}[w_{ki}]^2 + \frac{1}{a_{ki}} + \frac{1}{N_i}$, leading to the estimate $\eta_{ki}^{new} = \frac{1}{N_i} = \frac{\mathrm{E}[w_{ki}^2] - E[w_{ki}]^2}{e^{-\mathrm{E}[w_{ki}]} + 1}$. The empirical estimates of these first two moments can be gathered online by exponentially decaying averages using the same learning rate η_{ki}.

the Lagrangian F for maximizing $\mathrm{E}_{p^*}[\log p(\mathbf{y}|\mathbf{w})]$ under the constraints (7.7), and show that an application of rule (7.12) is expected to increase the value of F. We set

$$F(\mathbf{w},\boldsymbol{\lambda}) = \mathrm{E}_{p^*}[\log p(\mathbf{y}|\mathbf{w})] - \lambda_0 \left(1 - \sum_{k=1}^{K} e^{w_{k0}}\right) - \sum_{k=1}^{K}\sum_{j=1}^{m} \lambda_{kj} \left(1 - \sum_{i\in G_j} e^{w_{ki}}\right). \tag{7.15}$$

According to (7.5) one can write $p(\mathbf{y}|\mathbf{w}) = \sum_{k=1}^{K} e^{u_k}$ for $u_k = \sum_{i=1}^{K} w_{ki}\, y_i + w_{k0}$. Hence one arrives at the following conditions for the Lagrange multipliers $\boldsymbol{\lambda}$:

$$\sum_{k=1}^{K} \frac{\partial F}{\partial w_{k0}} = \sum_{k=1}^{K} \left(\mathrm{E}_{p^*}[\frac{e^{u_k}}{\sum_{l=1}^{K} e^{u_l}}] - \lambda_0 e^{w_{k0}}\right) = 0 \tag{7.16}$$

$$\sum_{i\in G_j} \frac{\partial F}{\partial w_{ki}} = \sum_{i\in G_j} \left(\mathrm{E}_{p^*}[y_i\, \frac{e^{u_k}}{\sum_{l=1}^{K} e^{u_l}}] - \lambda_{kj} e^{w_{ki}}\right) = 0, \tag{7.17}$$

which yield $\lambda_0 = 1$ and $\lambda_{kj} = \mathrm{E}_{p^*}[\frac{e^{u_k}}{\sum_{l=1}^{K} e^{u_l}}]$.

Plugging these values for $\boldsymbol{\lambda}$ into $\nabla_{\mathbf{w}} F \cdot \mathrm{E}^*_p[\Delta\mathbf{w}]$ with $\Delta\mathbf{w}$ defined by (7.12) shows that this vector product is always positive. Hence even a single application of learning rule (7.12) to a single new example $\mathbf{y}$, drawn according to p^*, is expected to increase $\mathrm{E}_{p^*}[\log p(\mathbf{y}|\mathbf{w})]$ under the constraints (7.7).

7.3.4 Impact of missing input values

We had shown at the end of 7.3.2 that learning in the spiking network corresponds to learning in the non-spiking network $\mathcal{N}_{\mathbf{w}}$ with missing values. A profound analysis of the correct handling of missing data in EM can be found in (Ghahramani & Jordan, 1997). Their analysis implies that the correct learning action is then not to change the weights w_{ki} for $i \in G_j$. However the STDP rule of Fig. 7.1B, as well as (7.12), reduce also these weights by η if z_k fires. This yields a modification of the equilibrium analysis (7.14):

$$\begin{aligned} \mathrm{E}[\Delta w_{ki}] = 0 \;&\Leftrightarrow\; (1-r)\left(p^*(y_i{=}1|z_k{=}1)\eta(e^{-w_{ki}}-1) - p^*(y_i{=}0|z_k{=}1)\eta\right) - r\eta = 0 \\ &\Leftrightarrow\qquad w_{ki} = \log p^*(y_i{=}1|z_k{=}1) + \log(1-r) \quad , \end{aligned} \tag{7.18}$$

where r is the probability that i belongs to a group G_j where the value of x_j is missing. Since this probability r is independent of the neuron z_k and also independent of the current value of the external variable x_i, this offset of $\log(1-r)$ is expected to be the same for all weights. It can easily be verified, that such an offset does not change the resulting probabilities of the competition in the E-step according to (7.2).

7.3.5 Relationship between the spiking and the non-spiking network

As indicated at the end of section 7.3.2, the learning process for the spiking network from section 7.2 with the simple STDP curve from Fig. 7.1B (and external variables x_j encoded by input spike trains from neurons y_i) is equivalent to a somewhat modified learning process of the non-spiking network $\mathcal{N}_{\mathbf{w}}$ with the Hebbian learning rule (7.12) and external variables x_j encoded by binary variables y_i. Each firing of a neuron z_k at some time t corresponds to a discrete time step in $\mathcal{N}_{\mathbf{w}}$ with an application of the Hebbian learning rule (7.12). Each neuron y_i that had fired during the time interval $[t - 10\text{ms}, t]$ contributes a value $\tilde{y}_i(t) = 1$ to the membrane potential $u_k(t)$ of the neuron z_k at time t, and a value $\tilde{y}_i(0) = 0$ if it did not fire during $[t - 10\text{ms}, t]$. Hence the weight updates at time t according to the simple STDP curve are exactly equal to those of (7.12) in the non-spiking network. However (7.12) will in general be applied to a corresponding input $\mathbf{y}$ where it may occur that for some $j \in \{1, \ldots, m\}$ one has $y_i = 0$ for all $i \in G_j$ (since none of the neurons y_i with $i \in G_j$ fired in the spiking network during $[t - 10\text{ms}, t]$). Hence we arrive at an application of (7.12) to an input $\mathbf{y}$ with missing values, as discussed in section 7.3.4.

Since several neurons z_k are likely to fire during the presentation of an external input $\mathbf{x}$ (each handwritten digit was presented for 50ms in section 7.2; but a much shorter presentation time of 10ms also works quite well), this external input $\mathbf{x}$ gives in general rise to several applications of the STDP rule. This corresponds to several applications of rule (7.12) to the same input (but with different choices of missing values) in the non-spiking network. In the experiments in section 7.2, every example in the non-spiking network with missing values was therefore presented for 10 steps, such that the average number of learning steps is the same as in the spiking case. The learning process of the spiking network corresponds to a slight variation of the stochastic online EM algorithm that is implemented through (7.12) according to the analysis of section 7.3.3.

7.4 Discussion

The model for discovering hidden causes of inputs that is proposed in this study presents an interesting shortcut for implementing and learning generative models for input data in networks of neurons. Rather than building and adapting an explicit model for re-generating internally the distribution of input data, our approach creates an implicit model of the input distribution (see Fig. 7.3B) that is encoded in the weights of neurons in a simple WTA-circuit. One might call it a Vapnik-style (Vapnik, 2005) approach towards generative modelling, since it focuses directly on the task to represent the most likely hidden causes of the inputs through neuronal firing. As the theoretical analysis via non-spiking networks in section 7.3 has shown, this approach also offers a new perspective for generating self-adapting networks on the basis of traditional artificial neural networks. One just needs to add the stochas-

tic and non-feedforward parts required for implementing stochastic WTA circuits to a 1-layer feedforward network, and apply the Hebbian learning rule (7.12) to the feedforward weights. One interesting aspect of the "implicit generative learning" approach that we consider in this study is that it retains important advantages of the generative learning approach, faster learning and better generalization (Ng & Jordan, 2002), while retaining the algorithmic simplicity of the discriminative learning approach.

Our approach also provides a new method for analyzing details of STDP learning rules. The simulation results of section 7.2 show that a simplified STDP rule that can be understood clearly from the perspective of stochastic online EM with a suitable Hebbian learning rule, provides good performance in discovering hidden causes for a standard benchmark dataset. A more complex STDP rule, whose learning curve better matches experimentally recorded average changes of synaptic weights, provides almost the same performance. Hence, the experimentally observed STDP curve could be the result of a suboptimal approximation to a theoretically optimal (but biologically hard to implement) simpler STDP curve. The large variance of experimentally observed synaptic weight changes in single trials (see (Dan & Poo, 2004)) could be viewed as circumstantial evidence for the difficulty of a biological implementation of a precise STDP update. Alternatively, the experimentally observed STDP learning curves could provide additional functional properties for learning in networks of spiking neurons. Additional further work will be needed to determine whether also more realistic shapes of EPSPs, or interactions between triplets of pre- and postsynaptic spikes, have a significant influence on the optimal STDP curve for discovering hidden causes of spike-inputs.

7.5 Acknowledgments

This chapter is based on the paper *STDP enables spiking neurons to detect hidden causes of their inputs*, which was written by Bernhard Nessler (BN), myself (MP), and Wolfgang Maass (WM). The theory and the non-spiking version of the algorithm was developed by BN. MP conducted all experiments and developed the extension of the algorithm to spiking neurons. The paper was written by WM, BN, and MP.

Appendix A

List of Publications

---

1. M. Pfeiffer. *Machine learning applications in computer games*. Master's thesis, Institute for Theoretical Computer Science, Graz University of Technology. 2003.

2. M. Pfeiffer. *Machine learning applications in computer games*. ÖGAI-Journal 23 (3). 2004.

3. M. Pfeiffer. *Reinforcement Learning of Strategies for Settlers of Catan*. Proceedings of the International Conference on Computer Games: Artificial Intelligence, Design and Education, Reading, UK. 2004.

4. M. Pfeiffer, A. Saffari A.A., and A. Juffinger. *Predicting Text Relevance from Sequential Reading Behavior*. Proceedings of the NIPS Workshop on Implicit Feedback and User Modeling, Whistler, Canada. 2005. (Winner of PASCAL Challenge "Inferring Relevance from Eye Movements", Competition One)

5. G. Neumann, M. Pfeiffer, and W. Maass. *Efficient Continuous-Time Reinforcement Learning with Adaptive State Graphs*. Proceedings of the 18th European Conference on Machine Learning (ECML), Warszaw, Poland. 2007.

6. B. Nessler, M. Pfeiffer, and W. Maass. *Hebbian learning of Bayes optimal decisions*. Proceedings of Advances in Neural Information Processing Systems (NIPS) 21, Vancouver, Canada. 2008.

7. M. Pfeiffer, B. Nessler, and W. Maass. *A Hebbian learning rule for optimal decision making*. Poster at NIPS Workshop on Machine Learning meets Human learning, Whistler, Canada. 2008.

8. B. Nessler, M. Pfeiffer, and W. Maass. *STDP enables spiking neurons to detect hidden causes of their inputs*. Proceedings of Advances in Neural Information Processing Systems (NIPS) 22, Vancouver, Canada. 2009.

9. M. Pfeiffer, B. Nessler, R.J. Douglas, and W. Maass. *Reward-modulated Hebbian Learning of Decision Making*. Neural Computation, in press. 2009.

10. M. Pfeiffer, M. Hartbauer, W. Maass, and H. Römer. *Probing real sensory worlds of receivers with unsupervised clustering*. In preparation. 2009.

A.1 Comments and Contributions to Publications

The first three publications resulted from my master's thesis on *Machine learning applications in computer games*, which won the ÖGAI-Award of the Austrian Society for Artificial Intelligence as the best master's thesis in the field of Artificial Intelligence in the years 2002-2003. The results from these papers are not contained in this thesis.

The paper *Predicting Text Relevance from Sequential Reading Behavior* was a joint paper with Amir Saffari and Andreas Juffinger, who contributed ideas for the design of features and helped in preparing the manuscript. I presented this paper at the NIPS workshop on "Implicit Feedback and User Modeling" in Whistler, Canada, and the solution presented in the paper won the first place in the international competition on "Inferring Relevance from Eye Movements", organized by the PASCAL network of excellence. This paper is the basis for Chapter 3 of this thesis.

The paper *Efficient Continuous-Time Reinforcement Learning with Adaptive State Graphs* was a joint paper with Gerhard Neumann (GN) and my supervisor Wolfgang Maass (WM). The presented algorithm was jointly developed by GN and myself, the experiments were performed by GN and myself, and the paper was written by myself and GN with input from WM. This paper was accepted for oral presentation at the 18th European Conference on Machine Learning in Warsaw, Poland. This paper is the basis for Chapter 4 of this thesis.

The paper *Hebbian learning of Bayes optimal decisions* was a joint paper with Bernhard Nessler (BN) and Wolfgang Maass (WM). The theory of the Bayesian Hebb rule was developed by BN with input from WM and myself, the experiments were performed by myself, and the paper was written by myself and BN with input from WM. This paper was presented at the 21st conference on Advances in Neural Information Processing Systems (NIPS) in Vancouver, Canada. This paper is the basis for Chapter 5 of this thesis.

The paper *Reward-modulated Hebbian Learning of Decision Making* was a joint paper with Bernhard Nessler (BN), Rodney J. Douglas (RD) from ETH Zürich, and Wolfgang Maass (WM). I developed the theory of the reward-modulated Bayesian Hebb rule based on results by BN in the previous paper. The experiments were performed by myself, and the paper was written by myself, WM, and RD with input from BN. The paper was accepted for publication in the journal Neural Computation (2008 impact factor 2.378) and is currently in press. This paper is the basis for Chapter 6 of this thesis. The poster *A Hebbian learning rule for optimal decision making* is based on earlier results from that paper and was presented at the NIPS workshop on "Machine Learning meets Human learning" in Whistler, Canada.

The paper *STDP enables spiking neurons to detect hidden causes of their inputs* was a joint paper with Bernhard Nessler (BN) and Wolfgang Maass (WM). The theory and the non-spiking version of the algorithm was developed by BN, the experiments and the extension of the algorithm to spiking neurons was done by myself, and the paper was written by WM, BN, and myself. This paper was accepted

for spotlight presentation at the 22nd conference on Advances in Neural Information Processing Systems (NIPS) in Vancouver, Canada. This paper is the basis for Chapter 7 of this thesis.

The paper *Probing real sensory worlds of receivers with unsupervised clustering* is a joint paper with Manfred Hartbauer (MH) and Heinrich Römer (HR) of Karl-Franzens University Graz, and Wolfgang Maass (WM). The data was recorded by MH and HR, the data analysis and the experiments were designed and performed by myself, and the paper was written by myself, HR, MH and WM. This paper is in preparation and will be submitted in 2009. This paper is the basis for Chapter 2 of this thesis.

References

Abbott, L. F., & Nelson, S. B. (2000). Synaptic plasticity: taming the beast. *Nature Neuroscience*, *3*, 1178-1183. 67, 106

Abraham, W. C. (2008). Metaplasiticity: tuning synapses and networks for plasticity. *Nature Reviews Neuroscience*, *9*, 387–399. 88

Alitto, H., Weyand, T., & Usrey, W. (2005). Distinct properties of stimulus-evoked bursts in the lateral geniculate nucleus. *Journal of Neuroscience*, *25*(2), 514–523. 25

Attias, H. (2003). Planning by probabilistic inference. In *Proc. of the 9th Int. Workshop on Artificial Intelligence and Statistics.* 100

Attneave, F. (1954). Some informational aspects of visual perception. *Psychological Review*, *61*, 183–193. 26

Audibert, J.-Y., Munos, R., & Szepesvari, C. (2007). Tuning bandit problems in stochastic environments. In *Proc. of the 18th International Conference on Algorithmic Learning Theory* (pp. 150–165). 99

Auer, P., Cesa-Bianchi, N., & Fischer, P. (2002). Finite time analysis of the multiarmed bandit problem. *Machine Learning*, *47*(2/3), 235–256. 71

Auer, P., Jaksch, T., & Ortner, R. (2009). Near-optimal regret bounds for reinforcement learning. In *Advances in Neural Information Processing Systems 21* (pp. 89–96). Cambridge, MA: MIT Press. 99

Bailey, C. H., Giustetto, M., Huang, Y.-Y., Hawkins, R. D., & Kandel, E. R. (2000). Is heterosynaptic modulation essential for stabilizing Hebbian plasticity and memory? *Nature Reviews Neuroscience*, *1*, 11–20. 68, 95, 98

Barlow, H. (1961). Possible principles underlying the transformation of sensory messages. In W. Rosenblith (Ed.), *Sensory communication* (pp. 217–234). Cambridge, MA: MIT Press. 26

Bauer, E., & Kohavi, R. (1999). An empirical comparison of voting classification algorithms: bagging, boosting, and variants. *Machine Learning*, *36*, 105–142. 39

Bertsekas, D. P., & Tsitsiklis, J. (1996). *Neuro-Dynamic Programming.* Belmont, MA: Athena Scientific. 58, 100

Bi, G., & Poo, M. (1998). Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type. *J Neuroscience*, *18*(24), 10464-10472. 2

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning.* New York: Springer. 55, 56, 57, 61, 69, 81, 100, 109

Blum, A., & Langley, P. (1997). Selection of relevant features and examples in machine learning. *Artificial Intelligence*, *97*(1–2), 245–271. 36

Boyan, J. A., & Moore, A. W. (1995). Generalization in reinforcement learning: Safely approximating the value function. In *Nips 7* (pp. 369–376). 42

Brumm, H., & Slabbekoorn, H. (2005). Acoustic communication in noise. *Advances in the Study of Behavior*, *35*, 151–209. 26

Caporale, N., & Dan, Y. (2008). Spike timing-dependent plasticity: a Hebbian learning rule. *Annu Rev Neuroscience*, *31*, 25–46. 67

Capranica, R., & Moffat, J. (1983). Neurobehavioral correlates of sound communication in anurans. In J.-P. Ewert, R. Capranica, & D. Ingle (Eds.), *Advances in vertebrate neuroethololgy* (pp. 701–730). New York: Plenum Press. 26

Chiappalone, M., Novellino, A., Vajda, I., Vato, A., Martinoia, S., & Pelt, J. van. (2005). Burst detection algorithms for the analysis of spatio-temporal patterns in cortical networks of neurons. *Neurocomputing*, *65–66*, 653–662. 11

Chittka, L., Skorupski, P., & Raine, N. (2009). Speed-accuracy tradeoffs in animal decision making. *Trends in Ecology and Evolution*, *24*(7), 400–407. 29

Christen, M., Kohn, A., Ott, T., & Stoop, R. (2006). Measuring spike pattern reliability with the Lempel-Ziv-distance. *Journal of Neuroscience Methods*, *156*, 342–350. 7, 12

Cocatre-Zilgien, J., & Delcomyn, F. (1992). Identification of bursts in spike trains. *Journal of Neuroscience Methods*, *41*, 19–30. 11

Dan, Y., & Poo, M. (2004). Spike timing-dependent plasticity of neural circuits. *Neuron*, *44*, 23-30. 106, 116

Dayan, P., & Abbott, L. F. (2001). *Theoretical neuroscience: Computational and mathematical modeling of neural systems*. Cambridge, MA: MIT Press. 68, 99

Dayan, P., & Daw, N. (2008). Decision theory, reinforcement learning, and the brain. *Cognitive, Affective, & Behavioral Neuroscience*, *8*, 429–453. 71

Dayan, P., & Kakade, S. (2001). Explaining away in weight space. In *Advances in Neural Information Processing Systems 13* (pp. 451–457). Cambridge, MA: MIT Press. 95, 99

Dayan, P., & Yu, A. (2003). Uncertainty and learning. *IETE Journal of Research*, *49*, 171–182. 99

Deneve, S. (2008a). Bayesian spiking neurons I, II. *Neural Computation*, *20*(1), 91-145. 56

Deneve, S. (2008b). Bayesian spiking neurons I: Inference. *Neural Computation*, *20*(1), 91-117. 81, 100

Domingos, P., & Pazzani, M. (1997). On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning*, *275*(29), 103–130. 69

Douglas, R. J., & Martin, K. A. (2004). Neuronal circuits of the neocortex. *Annu Rev Neurosci*, *27*, 419–451. 2, 4, 96, 106

Doya, K. (2002). Metalearning and neuromodulation. *Neural Networks*, *15*, 495–506. 88

Dragalin, V., Tartakovsky, A., & Veeravalli, V. (1999). Multihypothesis sequential probability ratio tests — Part I: Asymptotic optimality. *IEEE Transactions on Information Theory*, *45*(7), 2448–2461. 96

Eggermont, J. J. (1998). Is there a neural code? *Neuroscience and Biobehavioral Reviews*, *22*, 355–370. 6, 28

Eggermont, J. J., & Smith, G. M. (1996). Burst-firing sharpens frequency tuning in primary auditory cortex. *NeuroReport*, *7*, 753–757. 6

Eichhorn, J., Tolias, A. S., Zien, A., Kuss, M., Rasmussen, C. E., Weston, J., et al. (2004, 06). Prediction on spike data using kernel algorithms. In *17th annual conference on neural information processing systems* (Vol. 16, p. 1367-1374). MIT Press. 7

Eyherabide, H., Rokem, A., Herz, A., & Samengo, I. (2008). Burst firing is a neural code in an insect auditory system. *Frontiers in Computational Neuroscience*, *2*(3). 29

Eyherabide, H., Rokem, A., Herz, A., & Samengo, I. (2009). Bursts generate a non-reducible spike-pattern code. *Frontiers in Neuroscience*, *3*(1), 8–14. 29

Farries, M. A., & Fairhall, A. L. (2007). Reinforcement learning with modulated spike timing-dependent synaptic plasticity. *Journal of Neurophysiology*, *98*, 3648–3665. 68

Fellous, J.-M., Tiesinga, P., Thomas, P., & Sejnowski, T. (2004). Discovering spike patterns in neuronal responses. *Journal of Neuroscience*, *24*(12), 2989–3001. 27

Fregnac, Y. (2003). Hebbian synaptic plasticity. In M. A. Arbib (Ed.), *The Handbook of Brain Theory and Neural Networks* (pp. 515–522). Cambridge, MA: MIT Press. 67, 68

Frey, B. J., & Dueck, D. (2007). Clustering by passing messages between data points. *Science*, *315*(5814), 972–976. 7, 13, 27, 28

Fullard, J. (1998). The sensory co-evolution of moths and bats. In R. Hoy, A. Popper, & R. Fay (Eds.), *Comparative hearing: Insects* (pp. 279–326). New York, Berlin, Heidelberg: Springer. 30

Gabbiani, F., Metzner, W., Wessel, R., & C., K. (1996). Coding of time-varying signals in spike trains. *Nature*, *384*, 564-7. 25

Georgopoulos, A. P., Schwartz, A. P., & Ketner, R. E. (1986). Neuronal population coding of movement direction. *Science*, *233*, 1416–1419. 90, 97

Gerhardt, H., & Huber, F. (2002). *Acoustic communication in insects and frogs: common problems and diverse solutions*. Chicago: University of Chicago Press. 29

Ghahramani, Z., & Jordan, M. (1997). Mixture models for learning from incomplete data. *Computational Learning Theory and Natural Learning Systems*, *4*, 67–85. Available from `http://books.google.at/books?id=TzNao7YzOTYC&printsec=frontcover` 114

Gittins, J. (1979). Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society*, *41*, 148–177. 71

Gold, J. I., & Shadlen, M. N. (2007). The neural basis of decision making. *Annu Rev Neuroscience*, *30*, 535–574. 96

Gourevitch, B., & Eggermont, J. J. (2007). A nonparametric approach for detection of bursts in spike trains. *Journal of Neuroscience Methods*, *160*, 349–358. 11

Green, D., & Swets, J. (1966). *Signal detection theory and psychophysics*. New York: Wiley. 14

Griffiths, T., & Tenenbaum, J. (2005). Structure and strength in causal induction. *Cognitive Psychology*, *51*, 334–384. 99

Guestrin, C. E., & Ormoneit, D. (2001). Robust combination of local controllers. In *Proc. uai* (pp. 178–185). 42, 45

Gupta, A., & Long, L. N. (2007). Character recognition using spiking neural networks. *IJCNN*, 53–58. 106

Gurney, K., & Bogacz, R. (2006). The basal ganglia and cortex implement optimal decision making between alternative actions. *Neural Computation*, *19*, 442–477. 96, 98

Hahnloser, R. H. R., Sarpeshkar, R., Mahowald, M. A., Douglas, R. J., & Seung, H. S. (2000). Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *Nature*, *405*, 947–951. 96

Hart, P., Nilsson, N., & Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. SSC*, *4*, 100–107. 46, 48

Hastie, T., Tibshirani, R., & Friedman, J. (2001). *The elements of statistical learning*. Springer (New York). 12, 13

Hauser, H., Neumann, G., Ijspeert, A. J., & Maass, W. (2007). Biologically inspired kinematic synergies provide a new paradigm for balance control of humanoid robots. In *Proceedings of the IEEE-RAS 7th International Conference on Humanoid Robots (Humanoids 2007)*. 52

Hebb, D. O. (1949). *The organization of behavior*. New York: Wiley. 3, 56, 67

Hinton, G. E., & Ghahramani, Z. (1997). Generative models for discovering sparse distributed representations. *Philos Trans R Soc Lond B Biol Sci.*, *352*(1358), 1177–1190. 106

Hoy, R. (1992). The evolution of hearing in insects as an adaptation to predation from bats. In D. Webster, A. Popper, & R. Fay (Eds.), *The evolutionary biology of hearing* (pp. 115–130). New York, Berlin, Heidelberg: Springer. 30

Ide, J. S., & Cozman, F. G. (2002). Random generation of Bayesian networks. In *Proc. of the 16th Brazilian Symposium on Artificial Intelligence: Advances in Artificial Intelligence* (Vol. 2507, pp. 366–375). London: Springer. 61, 88

Izhikevich, E., Desai, N., Walcott, E., & Hoppenstaedt, F. (2003). Bursts as a unit of neural information: selective communication via resonance. *Trends in Neurosciences*, *26*(3), 161–167. 25

Jensen, F. V., & Nielsen, T. D. (2007). *Bayesian networks and decision graphs (2nd edition)*. New York: Springer. 97

Jong, N., & Stone, P. (2006). Kernel-based models for reinforcement learning. In *ICML Workshop on kernel machines and reinforcement learning*. 42

Kavraki, L., Svestka, P., Latombe, J., & Overmars, M. (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE T-RA*, *12*(4). 42, 45

Kearns, M., & Singh, S. (1998). Near-optimal performance for reinforcement learning in polynomial time. In *Proc. of the 15th International Conference on Machine Learning (ICML)* (pp. 260–268). 99

Kepecs, A., & Lisman, J. (2003). Information encoding and computation with spikes and bursts. *Network: Computation in neural systems*, *14*, 103–118. 25

Kononenko, I. (1998). Bayesian neural networks. *Biol. Cybernetics*, *61*, 361–370. 100

Krahe, R., & Gabbiani, F. (2004). Burst firing in sensory systems. *Nature Review Neuroscience*, *24*, 10731–10740. 6, 25

Kschischang, F. R., Frey, B. J., & Loeliger, H. A. (2001). Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, *47*(2), 498–519. 13, 69, 81, 82, 97

Kushner, J., & Yin, G. (1997). *Stochastic approximation algorithms and applications*. Springer. 113

Lai, T., & Robbins, H. (1985). Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, *6*, 4–22. 71

Lang, A., Teppner, I., Hartbauer, M., & Römer, H. (2005). Predation and noise in communication networks of neotropical katydids. In P. McGregor (Ed.), *Animal communication networks* (pp. 152–169). Cambridge University Press. 8, 10, 11

Lansner, A., & Ekeberg, O. (1998). A one-layer feedback artificial neural network with a Bayesian learning rule. *International Journal of Neural Systems*, *1*, 77–87. 100

Lansner, A., & Holst, A. (1996). A higher order Bayesian neural network with spiking units. *International Journal of Neural Systems*, *7*(2), 115–128. 100

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324. 106

Legenstein, R., Pecevski, D., & Maass, W. (2008). A learning theory for reward-modulated spike-timing-dependent plasticity with application to biofeedback. *PLoS Computational Biology*, *4*(10), 1–27. 68

Lestienne, R. (2001). Spike timing, synchronisation and information processing on the sensory side of the nervous system. *Progress in Neurobiology*, *65*, 545–591. 6

Lisman, J. (1997). Bursts as a unit of neural information: making unreliable synapses reliable. *TINS*, *20*, 38-43. 6, 25

Lohmiller, W., & Slotine, J. J. (1998). Contraction analysis for nonlinear systems. *Automatica*, *34*(6), 683–696. 75, 95

Maass, W. (2000). On the computational power of winner-take-all. *Neural Computation*, *12*(11), 2519–2536. 67

Maass, W., Natschlaeger, T., & Markram, H. (2002). Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Computation*, *14*(11), 2531-2560. 2

Machens, C., Gollisch, T., Kolesnikova, O., & Herz, A. (2005). Testing the efficiency of sensory coding with optimal stimulus ensembles. *Neuron*, *47*, 447–456. 26

Machens, C., Schütze, H., Franz, A., Kolesnikova, O., Stemmler, M., Ronacher,

B., et al. (2003). Single auditory neurons rapidly discriminate conspecific communication signals. *Nature Neuroscience*, *6*(4), 341–342. 27, 30

Machens, C., Stemmler, M., Prinz, P., Krahe, R., Ronacher, B., & Herz, A. (2001). Representation of acoustic communication signals by insect auditory receptor neurons. *Journal of Neuroscience*, *21*(9), 3215–3227. 26

Mainen, Z., & Sejnowski, T. (1995). Reliability of spike timing in neocortical neurons. *Science*, *268*, 1503-1505. 29

Markram, H. (2006). The Blue Brain project. *Nature Reviews Neuroscience*, *7*, 153–160. 2

Marsat, G., & Pollack, G. (2006). A behavioral role for feature detection by sensory bursts. *Journal of Neuroscience*, *26*, 10542–10547. 6

Marsat, G., & Pollack, G. (2007). Efficient inhibition of bursts by bursts in the auditory system of crickets. *Journal of Comparative Physiology A*, *193*, 625–633. 25

McGregor, P., & Krebs, J. (1984). Sound degradation as a distance cue in great tit (*parus major*) song. *Behavioral Ecology and Sociobiology*, *16*, 49–56. 29

McLachlan, G., & Peel, D. (2000). *Finite mixture models*. Wiley. 109

Meilă, M., & Heckerman, D. (2001). An experimental comparison of model-based clustering methods. *Machine Learning*, *42*(1), 9–29. 109

Metzner, W., Koch, C., Wessel, R., & Gabbiani, F. (1998). Feature extraction by burst-like spike patterns in multiple sensory maps. *Journal of Neuroscience*, *18*, 2283–2300. 6

Molina, J., & Stumpner, A. (2005). Effects of pharmacological treatment and photoinactivation on the directional responses of an insect neuron. *Journal of Experimental Zoology*, *303A*(12), 1085–1103. 8, 26

Montague, P., Dayan, P., Person, C., & Sejnowski, T. (1995). Bee foraging in uncertain environments using predictive Hebbian learning. *Nature*, *377*, 725–728. 98

Moore, A. W., & Atkeson, C. G. (1993). Prioritized sweeping: Reinforcement learning with less data and less real time. *Machine Learning*, *13*, 103–130. 42, 48, 49, 50

Morrison, A., Aertsen, A., & Diesmann, M. (2007). Spike-timing-dependent plasticity in balanced random networks. *Neural Computation*, *19*, 1437–1467. 106

Morton, E. (1975). Ecological sources of selection on avian sounds. *American Naturalist*, *108*, 17–34. 6

Mussa-Ivaldi, F., & Bizzi, E. (2000). Motor learning through the combination of primitives. *Phil. Trans. R. Soc. London B*, *355*, 1755–1769. 3

Narayan, R., Graña, G., & Sen, K. (2006). Distinct time scales in cortical discrimination of natural sounds in songbirds. *Journal of Neurophysiology*, *96*, 252–258. 7, 27

Neapolitan, R. (2004). *Learning Bayesian networks*. Upper Saddle River, NJ: Prentice Hall. 73, 97

Neftci, E., Chicca, E., Indiveri, G., Slotine, J., & Douglas, R. (2008). Contrac-

tion properties of VLSI cooperative competitive neural networks of spiking neurons. In *Advances in Neural Information Processing Systems.* Cambridge, MA: MIT Press. 96

Nessler, B., Pfeiffer, M., & Maass, W. (2009). Hebbian learning of Bayes optimal decisions. *In Proc. of NIPS 2008: Advances in Neural Information Processing Systems*, *21*. (MIT Press) 86, 113

Neuhofer, D., Wohlgemuth, A., Stumpner, A., & Ronacher, B. (2008). Evolutionarily conserved coding properties of auditory neurons across grasshopper species. *Proceedings of the Royal Society of London, Series B: Biological Sciences*, *275*(1646), 1965–1974. 26

Ng, A. Y., & Jordan, M. I. (2002). On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes. *Advances in Neural Information Processing Systems (NIPS)*, *14*, 841–848. 64, 116

Ng, A. Y., Jordan, M. I., & Weiss, Y. (2001). On spectral clustering: Analysis and an algorithm. In *Advances in neural information processing systems* (Vol. 14, pp. 849–856). 13

Nicolelis, M., Ghazanfar, A., Stambaugh, C., Oliveira, L., Laubach, M., Chapin, J., et al. (1998). Simultaneous encoding of tactile information by three primate cortical areas. *Nature Neuroscience*, *1*(7), 621–630. 29

O'Keefe, J., Burgess, N., Donnett, J., Jeffery, K., & Maguire, E. (1998). Place cells, navigational accuracy, and the human hippocampus. *Philosophical Transactions of the Royal Society of London*, *353*(1373), 1333-1340. 90, 97

Olshausen, B. A., & Field, D. J. (1996). Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, *381*, 607–609. 97

Opitz, D., & Maclin, R. (1999). Popular ensemble methods: an empirical study. *Journal of Artificial Intelligence Research*, *11*, 169–198. 39

Ormoneit, D., & Sen, S. (2002). Kernel-based reinforcement learning. *Machine Learning*, *49*(2-3), 161–178. 42

Oswald, A.-M., Chacron, M., Doiron, B., Bastian, J., & Maler, L. (2004). Parallel processing of sensory input by bursts and isolated spikes. *Journal of Neuroscience*, *24*(18), 4351–4362. 25

Pollack, G. (1988). Selective attention in an insect auditory neuron. *Journal of Neuroscience*, *8*(7), 2635–2639. 28

Pouget, A., & Latham, P. (2002). Population codes. In M. A. Arbib (Ed.), *The Handbook of Brain Theory and Neural Networks, 2nd ed.* (pp. 893–897). Cambridge, MA: MIT Press. 2, 90, 97

Rao, R. P. N. (2007). Neural models of Bayesian belief propagation. In K. Doya, S. Ishii, A. Pouget, & R. P. N. Rao (Eds.), *Bayesian brain.* (pp. 239–267). Cambridge, MA: MIT-Press. 55, 56, 100

Rasmussen, C., & Williams, C. (2006). *Gaussian processes for machine learning.* MIT Press. 47

Rayner, K. (1998). Eye movements in reading and information processing: 20 years of research. *Psychological Bulletin*, *124*, 372–422. 34

Rescorla, R. A., & Wagner, A. R. (1972). Classical conditioning II. In A. H. Black & W. F. Prokasy (Eds.), *A theory of Pavlovian conditioning: Variations in the effectiveness of reinforcement and nonreinforcement* (pp. 64–99). Appleton–Century–Crofts. 68, 83, 84, 95, 98

Reynolds, J. N., Hyland, B. I., & Wickens, J. R. (2001). A cellular mechanism of reward-related learning. *Nature*, *413*, 67–70. 68

Rheinlaender, J., & Römer, H. (1986). Insect hearing in the field: I. the use of identified nerve cells as "biological microphones". *Journal of Comparative Physiology*, *158*, 647–651. 8

Richards, D., & Wiley, R. (1980). Reverberations and amplitude fluctuations in the propagation of sound in a forest: implications for animal communication. *The American Naturalist*, *115*(3), 381–399. 29

Rieke, F., Bodnar, D., & Bialek, W. (1995). Naturalistic stimuli increase the rate and efficiency of information transmission by primary auditory afferents. *Proc. of the Royal Society of London: Biological Sciences*, *262*, 259–265. 26

Rieke, F., Warland, D., Steveninck, R. R. D. van, & Bialek, W. (1997). *SPIKES: Exploring the neural code*. MIT Press, Cambridge, MA. 6

Riesenhuber, M., & Poggio, T. (1999). Models of object recognition. *Nature Neuroscience*, *2*, 1019–1025. 90

Rokem, A., Watzl, S., Gollisch, T., Stemmler, M., Herz, A., & Samengo, I. (2006). Spike-timing precision underlies the coding efficiency of auditory receptor neurons. *Journal of Neurophysiology*, *95*, 2541-2552. 26

Römer, H. (1985). Anatomical representation of frequency and intensity in the auditory system of orthoptera. In K. Kalmring & N. Elsner (Eds.), *Acoustic and vibrational communication in insects* (pp. 25–32). Paul Parey. 8

Römer, H. (1998). The sensory ecology of acoustic communication in insects. In R. Hoy, A. Popper, & R. Fay (Eds.), *Comparative hearing: Insects* (pp. 63–96). New York, Berlin, Heidelberg: Springer. 6

Römer, H., & Bailey, W. J. (1986). Insect hearing in the field: Ii. spacing behaviour and related acoustic cues for the male mygalopsis marki (tettigoniidae). *Journal of Comparative Physiology*, *159*, 627–638. 8

Römer, H., Hedwig, B., & Ott, S. R. (2002). Contralateral inhibition as a sensory bias: The neural basis for a female preference in a synchronously calling bushcricket, mecopoda elongata. *European Journal of Neuroscience*, *15*(10), 1655–1662. 9

Römer, H., & Krusch, M. (2000). A gain-control mechanism for processing of chorus sounds in the afferent auditory pathway of the bushcricket *tettigonia viridissima* (orthoptera; tettigoniidae). *Journal of Comparative Physiology A*, *186*, 181–191. 28

Römer, H., & Lewald, J. (1992). High-frequency sound transmission in natural habitats: implications for the evolution of insect acoustic communication. *Behavioral Ecology and Sociobiology*, *29*, 437–444. 6, 8, 29

Römer, H., Marquart, V., & Hardt, M. (1988). The organization of a sensory neuropil in the auditory pathway of grasshoppers and bushcrickets. *Journal*

of Comparative Neurology, *275*, 201–215. 8, 27

Ronacher, B., Franz, A., Wohlgemuth, S., & Hennig, R. (2004). Variability of spike trains and the processing of temporal patterns of acoustic signals - problems, constraints, and solutions. *Journal of Comparative Physiology*, *190*, 257–277. 6, 26

Ronacher, B., & Stumpner, A. (1988). Filtering of behaviourally relevant temporal parameters of a grasshopper song by an auditory interneuron. *Journal of Comparative Physiology A*, *163*, 517–523. 26

Rosenblatt, J. F. (1962). *Principles of neurodynamics*. New York: Spartan Books. 2

Roth, D. (1999a). Learning in natural language. In *Proc. of ijcai* (pp. 898–904). 56

Roth, D. (1999b). Learning in natural language. In *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)* (pp. 898–904). 69, 96

Sakai, Y., Okamoto, H., & Fukai, T. (2006). Computational algorithms and neuronal network models underlying decision processes. *Neural Networks*, *19*(8), 1091–1105. 98

Salojärvi, J., Puolamäki, K., Simola, J., Kovanen, L., Kojo, I., & Kaski, S. (2005). *Inferring relevance from eye movements: Feature extraction.* (Tech. Rep. No. Report A82). Helsinki University of Technology, Publications in Computer and Information Science. 3, 33

Sandberg, A., Lansner, A., Petersson, K. M., & Ekeberg, O. (2002). A Bayesian attractor network with incremental learning. *Network: Computation in Neural Systems*, *13*, 179–194. 56, 100

Sato, M. (1999). Fast learning of on-line EM algorithm. *Rapport Technique, ATR Human Information Processing Research Laboratories*. 113

Schrauwen, B., & Campenhout, J. V. (2007). Linking non-binned spike train kernels to several existing spike train metrics. *Neurocomputing*, *70*, 1247–1253. 7, 12

Schultz, W., Dayan, P., & Montague, P. (1997). A neural substrate of prediction and reward. *Science*, *275*, 1593-9. 99

Shadlen, M., & Newsome, W. (1994). Noise, neural codes and cortical organization. *Current Opinion in Neurobiology*, *4*, 569-579. 28

Shpigelman, L., Singer, Y., Paz, R., & Vaadia, E. (2003). Spikernels: Embedding spiking neurons in inner-product spaces. In *Advances in neural information processing systems* (Vol. 15). 7

Simoncelli, E. P., & Olshausen, B. A. (2001). Natural image statistics and neural representation. *Annu Rev Neurosci*, *24*, 1193–1216. 26

Simsek, Ö., & Barto, A. (2006). An intrinsic reward mechanism for efficient exploration. In *Icml* (pp. 833–840). 42

Steimer, A., Maass, W., & Douglas, R. (2009). Belief-propagation in networks of spiking neurons. *Neural Computation*, *21*, 2502–2523. 69

Sugrue, L. P., Corrado, G. S., & Newsome, W. T. (2004). Matching behavior and the representation of value in the parietal cortex. *Science*, *304*, 1782–1787. 66, 71

Sugrue, L. P., Corrado, G. S., & Newsome, W. T. (2005). Choosing the greater of two goods: Neural currencies for valuation and decision making. *Nature Reviews Neuroscience*, *6*(5), 363–375. 66, 72, 98

Sutton, R. S. (1992). Gain adaptation beats least squares. In *Proceedings of the 7th Yale Workshop on Adaptive and Learning Systems* (pp. 161–166). New Haven, CT. 95, 99

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction*. Cambridge, MA: MIT Press. 42, 43, 44, 50, 68, 71, 99

Toups, J., & Tiesinga, P. (2006). Methods for finding and validating neural spike patterns. *Neurocomputing*, *69*, 1362–1365. 27

Turnbull, L., Dian, E., & Gross, G. (2005). The string method of burst identification in neuronal spike trains. *Journal of Neuroscience Methods*, *145*, 23–35. 11

van Rossum, M. (2001). A novel spike distance. *Neural Computation*, *13*, 751–763. 7, 12, 27

Vapnik, V. (2005). Universal learning technology: Support vector machines. *NEC Journal of Advanced Technology*, *2*, 137–144. 115

Verma, D., & Rao, R. (2006). Goal-based imitation as probabilistic inference over graphical models. In *Advances in Neural Information Processing Systems 18* (pp. 1393–1400). Cambridge, MA: MIT Press. 100

Victor, J. (2005). Spike train metrics. *Current Opinion in Neurobiology*, *15*(5), 585–592. 7, 12, 27

Victor, J., & Purpura, K. (1997). Metric-space analysis of spike trains: Theory, algorithms and applications. *Network: Computation in neural systems*, *8*, 127–164. 27

Wald, A., & Wolfowitz, J. (1948). Optimal character of the sequential probability ratio test. *Ann. Math. Statist.*, *19*, 326–339. 96

Wang, L., Narayan, R., a, G. G., Shamir, M., & Sen, K. (2007). Cortical discrimination of complex natural stimuli: Can single neurons match behavior? *Journal of Neuroscience*, *27*(3), 582–589. 7, 27

Wang, X. J. (2002). Probabilistic decision making by slow reverberation in cortical circuits. *Neuron*, *36*, 955–968. 98

Waters, D. (1996). The peripheral auditory characteristics of noctuid moths: information encoding and endogenous noise. *Journal of Experimental Biology*, *199*, 857–868. 30

Wehner, R. (1987). 'matched filters' — neural models of the external world. *Journal of Comparative Physiology A*, *161*(4), 511–531. 26

Wiley, R., & Richards, D. (1978). Physical constraints on acoustic communication in the atmosphere: Implications for the evolution of animal vocalizations. *Behavioral Ecology and Sociobiology*, *3*, 69–94. 6

Wiley, R., & Richards, D. (1982). Adaptations for acoustic communication in birds: Sound transmission and signal detection. In D. Kroodsma, E. Miller, & H. Quellet (Eds.), *Acoustic communication in birds* (pp. 131–181). New York: Academic Press. 6

Witten, I., & Frank, E. (2005). *Data mining: Practical machine learning tools and techniques* (2nd ed.). San Francisco, CA: Morgan Kaufmann. 38

Wyttenbach, R., May, M., & Hoy, R. (1996). Categorical perception of sound frequency by crickets. *Science*, *273*(5281), 1542–1544. 30

Yang, T., & Shadlen, M. N. (2007). Probabilistic reasoning by neurons. *Nature*, *447*, 1075–1080. 56, 64, 66, 70, 98, 100, 101, 102, 103

Yu, A., & Dayan, P. (2003). Expected and unexpected uncertainty: ACh and NE in the neocortex. In *Advances in Neural Information Processing Systems 15* (pp. 157–164). Cambridge, MA: MIT Press. 88, 99

Yuille, A. L. (2006). Augmented Rescorla-Wagner and maximum likelihood estimation. In *Advances in Neural Information Processing Systems 18* (pp. 1561–1568). Cambridge, MA: MIT Press. 89, 95

Yuille, A. L., & Geiger, D. (2003). Winner-take-all networks. In M. A. Arbib (Ed.), *The Handbook of Brain Theory and Neural Networks* (pp. 1228–1231). MIT Press. 66